STEP 7 V14 SP1 ヘルプファイル
S7-1200/S7-1500専用命令

www.siemens.com/jp/tia-portal

法律上の注意

警告事項

本書には、ユーザーの安全を確保し製品の損傷を防止する上で守るべき注意事項が記載されています。ユーザーの安全に関する注意事項は、安全警告サインで強調表示されています。このサインは、物的損傷に関する注意事項には表示されません。注意事項は、危険度によって以下の等級に分類されています。

⚠ 危険
回避しなければ、直接的な死亡または重傷に至る危険な状態を示します。

⚠ 警告
回避しなければ、死亡または重傷に至るおそれのある危険な状況を示します。

⚠ 注意
回避しなければ、軽度または中度の人身傷害を引き起こすおそれがある危険な状況を示します(安全警告サイン付き)。

注意
回避しなければ、物的損傷を引き起こすおそれのある危険な状況を示します(安全警告サインなし)。

複数の危険レベルに相当する場合は、通常、最も危険度の高い(番号の低い)事項が表示されることになっています。安全警告サイン付きの人身傷害に関する注意事項があれば、物的損傷に関する警告が付加されます。

有資格者

装置/システムのセットアップおよび使用にあたっては必ず本書を参照してください。機器のインストールおよび操作は有資格者のみが行うものとします。有資格者とは、法的な安全規制/規格に準拠してアースの取り付け、電気回路、設備およびシステムの設定に携わることを承認されている技術者のことをいいます。

シーメンス製品の適切な使用

以下の事項に注意してください。

⚠ 警告
シーメンス製品は、カタログおよび付属の技術説明書の指示に従ってお使いください。他社の製品または部品との併用は、弊社の推奨もしくは許可がある場合に限ります。シーメンス製品を正しく安全にご使用いただくには、適切な運搬、保管、取り付け、組み立て、据え付け、配線、始動、操作、保守を行ってください。ご使用になる場所は、許容された範囲を必ず守ってください。付属の技術説明書に記述されている指示を順守してください。

商標

本書において®で識別されるすべての名称は、Siemens AG の登録商標です。本書に記載するその他の称号は商標であり、第三者が自己の目的において使用した場合、所有者の権利を侵害することになります。

免責事項

本書のハードウェアおよびソフトウェアに関する記述と、実際の製品内容との一致については検証済みです。しかしながら、本書の記述が実際の製品内容と異なる可能性もあり、完全な一致が保証されているわけではありません。記述内容については定期的に検証し、訂正が必要な場合は次の版で更新いたします。
本マニュアルは、英語版を原本として参照のみを目的として作成されるものであり、当社は、当該翻訳の不足や正確性に関して責任を負わないものとします。

英語マニュアル：SIMATIC STEP 7 Basic/Professional V15.1 and SIMATIC WinCC V15.1
(https://support.industry.siemens.com/cs/jp/ja/view/109755202/en)

命令

この章には下記に関する情報が記載されています：

命令の一般パラメータ

この章には下記に関する情報が記載されています：

- 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味 (S7-1200, S7-1500)
- 出力パラメータ RET_VAL を使用したエラーの評価 (S7-1200, S7-1500)

非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味

非同期命令

非同期で機能する命令については、ファンクションは複数の呼び出しを使用して実行されます。

ジョブの識別

非同期命令を使用して、上記の SFC のいずれかによってプロセス割り込みのトリガ、DP スレーブへの制御コマンドの出力、データ転送の開始、または設定されていない接続の中断を実行した後、現在のジョブが完了する前に再び同じ SFC を呼び出す場合、その SFC の応答は、2 回目の呼び出しに同じジョブが含まれているかどうかによって異なります。

パラメータ REQ

入力パラメータ REQ(request)は、ジョブの開始にのみ使用されます。

- 現在有効になっていないジョブについての命令を呼び出す場合、そのジョブは REQ = 1 で開始されます(ケース 1)。
- 特定のジョブが開始されてまだ完了していないときに、同じジョブを実行するために再び命令を呼び出す場合(たとえば周期割り込み OB で)、その命令では REQ は評価されません(ケース 2)。

パラメータ RET_VAL および BUSY

出力パラメータ RET_VAL および BUSY はジョブのステータスを示します。

次のセクションの注記に留意してください。 出力パラメータ RET_VAL を使用したエラーの評価

- ケース 1(REQ=1 による最初の呼び出し)では、システムリソースが使用可能で供給が適切であれば、入力パラメータが RET_VALW#16#7001 に入力されます。BUSY がセットされます。

  必要なシステムリソースが現在使用されているか、入力パラメータにエラーがある場合は、対応するエラーコードが RET_VAL に入力され、BUSY の値が 0 になります。
- ケース 2(中間呼び出し)では、W#16#7002 が RET_VAL に入力され(これは警告:「ジョブはまだ処理中です!」に対応しています)、BUSY がセットされます。
- ジョブの最後の呼び出しには、以下が適用されます。
  - 命令「DPNRM_DG」については、データ伝送にエラーがなければ、データ数(バイト単位)が整数として RET_VAL に入力されます。この場合、BUSY の値は「0」になります。

    エラーがある場合は、エラー情報が RET_VAL に入力されます。この場合は、BUSY を評価してはなりません。
  - それ以外のすべての命令については、ジョブがエラーなく実行されると、「0」が RET_VAL に入力されます。この場合、BUSY の値は「0」になります。 エラーがある場合は、エラーコードが RET_VAL に入力されます。この場合、BUSY の値は「0」になります。

    注記
    最初と最後の呼び出しが同じであれば、RET_VAL と BUSY の応答は最後の呼び出しについての説明どおりになります。

概要

以下の表に、上記で説明した関係の概要を示します。 特に、命令の呼び出しが完了したのにジョブの実行が完了していない場合の出力パラメータの可能な値を示します。

注記

各呼び出しの後に、プログラムで関連する出力パラメータを評価する必要があります。

「実行中の」ジョブが実行されている間の呼び出し、REQ、RET_VAL および BUSY 間の関係。

呼び出しの番号	呼び出しのタイプ	REQ	RET_VAL	BUSY
1	最初の呼び出し	1	W#16#7001	1
			エラーコード	0
2 ~ (n - 1)	中間呼び出し	対象外	W#16#7002	1
n	最後の呼び出し	対象外	エラーが発生していない場合は W#16#0000。	0
			エラーが発生した場合はエラーコード	0

出力パラメータ RET_VAL を使用したエラーの評価

エラー情報のタイプ

実行された命令は、ユーザープログラムで CPU が命令のファンクションを正常に実行できたかどうかを示します。

発生したエラーに関する情報は、次の 2 つの方法で取得できます。

- ステータスワードの BR ビットで取得
- 出力パラメータ RET_VAL(return value)で取得。

> **注記**
> 命令固有の出力パラメータを評価する前に、常に以下の手順を実行する必要があります。
> - 最初に、ステータスワードの BR ビットを評価します。
> - 次に、出力パラメータ RET_VAL をチェックします。
>
> BR ビットがエラーが発生したことを示しているか、RET_VAL に一般エラーコードが含まれている場合は、命令固有の出力パラメータを評価してはなりません。

戻り値のエラー情報

命令は、ステータスワードのバイナリの結果ビット(BR)に値「0」を入力することによって、その命令の実行中にエラーが発生したことを示します。 命令の中には、戻り値(RET_VAL)として知られている出力パラメータに追加のエラーコードを提供するものもあります。 出力パラメータ RET_VAL(説明については下記を参照)に一般エラーが入力される場合、このことはステータスワードの BR ビットの値「0」でのみ示されます。

戻り値は整数データタイプ(INT)の値です。 戻り値と値「0」との関係によって、ファンクションの実行中にエラーが発生したかどうかが示されます。

命令の CPU 実行	BR	戻り値	整数の符号
エラーあり	0	「0」未満	負 (符号ビットは「1」)
エラーなし	1	「0」以上	正 (符号ビットは「0」)

エラー情報に対する応答

RET_VAL には、次の 2 タイプのエラーコードが含まれます。

- すべての命令が出力できる一般エラーコードおよび
- ある特定の命令が出力でき、その命令固有のファンクションに関連している固有のエラーコード

命令の実行中に発生したエラーに応答するようにプログラムを記述できます。 このようにして、最初のエラーが原因となってさらにエラーが発生するのを防止します。

一般エラー情報および固有のエラー情報

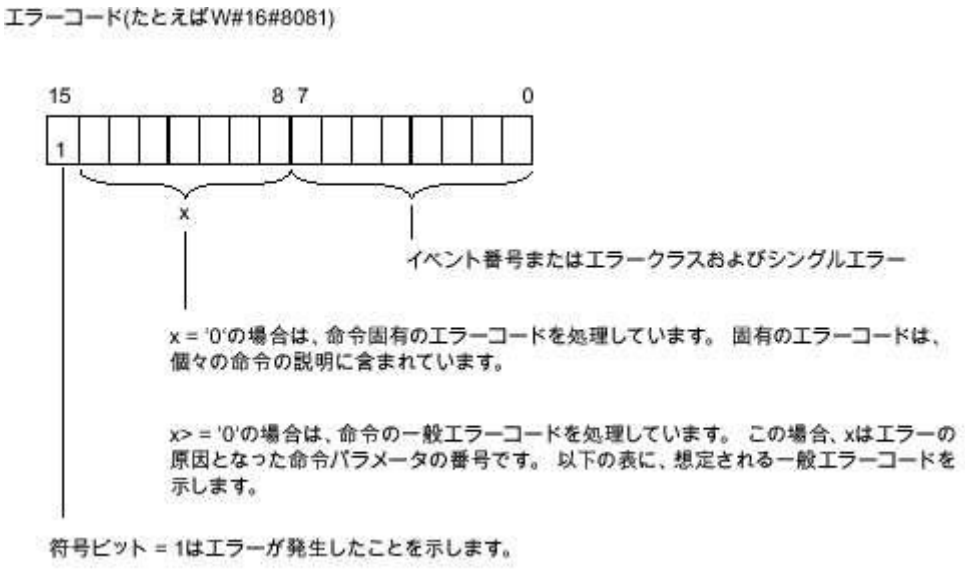
命令の戻り値(RET_VAL)は、次の 2 タイプのエラーコードのいずれかを提供します。

- どの命令でも発生する可能性のあるエラーに関連する一般エラーコード。

- 特定の命令のみに関連する固有のエラーコード。

出力パラメータ RET_VAL のデータタイプが整数(INT)であっても、命令のエラーコードは 16 進数値に基づいてグループ化されます。 戻り値を調べて、その値をこの説明書にリストされたエラーコードと比較する場合は、エラーコードを 16 進数形式で表示します。

下の図に、16 進数形式のシステムファンクションのエラーコードの構造を示します。



一般エラー情報

一般エラーコードは、すべての命令で発生する可能性のあるエラーを示します。 一般エラーコードは、次の 2 つの番号で構成されています。

- 1 ~ 111 のパラメータ番号。ここで、1 は呼び出された命令の 1 つ目のパラメータ、2 は 2 つ目のパラメータを示します(3 以降も同様)。
- 0 ~ 127 のイベント番号。イベント番号は、同期エラーが発生したことを示します。

以下の表に、一般エラーのコードと各エラーの説明を示します。

注記

一般エラーコードが RET_VAL に入力された場合は、以下の状況が考えられます。

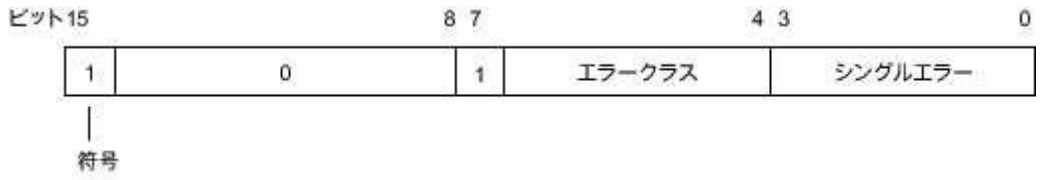
- 命令に関連する操作が開始されているか、すでに完了している可能性があります。
- 操作が実行されたときに、固有の命令エラーが発生した可能性があります。 ただし、その後一般エラーが発生したために、固有のエラーを示すことができなくなった可能性があります。

固有のエラー情報

命令の中には、その命令固有のエラーコードを提供する戻り値を持つものもあります。 固有のエラーコードは、特定の命令でのみ発生する可能性のあるエラーを示します。

固有のエラーコードは、次の 2 つの番号で構成されています。

- 0 ~ 7 のエラークラス。
- 0 ~ 15 のエラー番号。



一般エラーコード

以下の表に、戻り値の一般エラーコードの説明を示します。 エラーコードは 16 進数形式で表示されます。 各コード番号に含まれる文字 x は単なるプレースホルダーであり、エラーの原因となったシステムファンクションパラメータの番号を表すものです。

一般エラーコード

エラーコード (W#16#...)	説明
8x7F	内部エラー このエラーコードは、パラメータ x の内部エラーを示しています。
8x01	VARIANT パラメータの構文 ID が無効です
8x22	パラメータの読み出し時に、範囲長エラーが発生しました。
8x23	パラメータの書き込み時に、範囲長エラーが発生しました。 このエラーコードは、パラメータ x の全体または一部がアドレスの範囲外に配置されていることを示しているか、VARIANT パラメータでビット範囲の長さが 8 の倍数になっていないことを示しています。
8x24	パラメータの読み出し時に、範囲エラーが発生しました。
8x25	パラメータの書き込み時に、範囲エラーが発生しました。 このエラーコードは、パラメータ x がシステムファンクションに対して無効な範囲内に配置されていることを示しています。 無効な範囲については、個々のファンクションの説明を参照してください。
8x26	パラメータに含まれているタイマのセル番号が大きすぎます。 このエラーコードは、パラメータ x で指定されたタイマのセルが存在していないことを示しています。
8x27	パラメータに含まれているカウンタのセル番号が大きすぎます(カウンタ番号エラー)。 このエラーコードは、パラメータ x で指定されたカウンタのセルが存在していないことを示しています。
8x28	パラメータの読み出し時に、整列エラーが発生しました。
8x29	パラメータの書き込み時に、整列エラーが発生しました。 このエラーコードは、パラメータ x への参照が 0 以外のビットアドレスを持つオペランドであることを示しています。

8x30	パラメータが読み取り専用のグローバル DB に配置されています。
8x31	パラメータが読み取り専用のインスタンス DB に配置されています。 このエラーコードは、パラメータ x が読み取り専用のデータブロックに配置されていることを示しています。 データブロックがシステムファンクション自体によって開かれた場合、システムファンクションは常に値 W#16#8x30 を返します。
8x32	パラメータに含まれている DB 番号が大きすぎます(DB 番号エラー)。
8x34	パラメータに含まれている FC 番号が大きすぎます(FC 番号エラー)。
8x35	パラメータに含まれている FB 番号が大きすぎます(FB 番号エラー)。 このエラーコードは、パラメータ x に含まれているブロック番号が許容されている最大の番号よりも大きいことを示しています。
8x3A	パラメータに、ロードされていない DB の番号が含まれています。
8x3C	パラメータに、ロードされていない FC の番号が含まれています。
8x3E	パラメータに、ロードされていない FB の番号が含まれています。
8x42	システムが周辺機器の入力領域からパラメータを読み出そうとしたときに、アクセスエラーが発生しました。
8x43	システムが周辺機器の出力領域へパラメータを書き込もうとしたときに、アクセスエラーが発生しました。
8x44	エラー発生後の n 番目の(n > 1)読み出しアクセスでのエラーです。
8x45	エラー発生後の n 番目の(n > 1)書き込みアクセスでのエラーです。 このエラーコードは、必要なパラメータへのアクセスが拒否されていることを示しています。

基本命令

この章には下記に関する情報が記載されています：

- LAD (S7-1200, S7-1500)
- FBD (S7-1200, S7-1500)
- STL (S7-1500)
- SCL (S7-1200, S7-1500)
- GRAPH (S7-1500)

LAD

この章には下記に関する情報が記載されています：

ビット論理演算

この章には下記に関する情報が記載されています：

---| |---: ノーマルオープン

説明

ノーマルオープンの有効/無効は、関連するオペランドの信号状態によって変わります。 オペランドの信号状態が「1」の場合、ノーマルオープンが閉じ、出力の信号状態は入力の信号状態に設定されます。

オペランドの信号状態が「0」の場合、ノーマルオープンが有効にならず、命令の出力が信号状態「0」にリセットされます。

直列接続の場合、2 つ以上のノーマルオープンはビット単位で AND によって連結されます。 直列接続では、すべての接点を閉じる時に電力潮流が発生します。

並列接続の場合、ノーマルオープンは OR によって連結されます。 並列接続では、いずれかの接点が閉じる時に電力潮流が発生します。

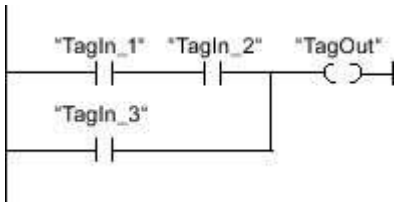
パラメータ

次の表に、ブロックパラメータのタイプを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<オペランド>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	信号状態を問い合せるオペランド。

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「1」であること。

---| / |---: ノーマルクローズ

説明

ノーマルクローズの有効/無効は、関連するオペランドの信号状態によって変わります。オペランドの信号状態が「1」の場合、ノーマルクローズが開き、命令の出力が信号状態「0」にリセットされます。

オペランドの信号状態が「0」の場合、ノーマルクローズが有効にならず、入力の信号状態が出力に転送されます。

直列接続の場合、2 つ以上のノーマルクローズはビット単位で AND によって連結されます。 直列接続では、すべての接点を閉じる時に電力潮流が発生します。

並列接続の場合、ノーマルクローズは OR によって連結されます。 並列接続では、いずれかの接点が閉じる時に電力潮流が発生します。

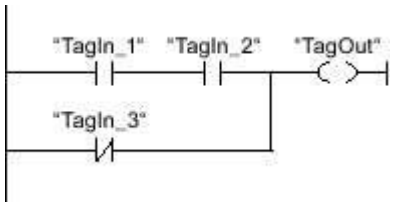
パラメータ

次の表に、ブロックパラメータのタイプを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<オペランド>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	信号状態を問い合せるオペランド。

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「0」であること。

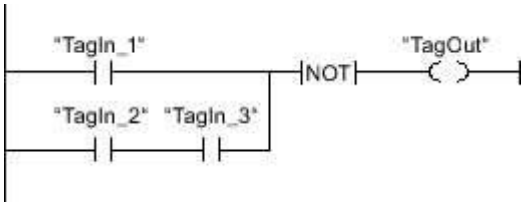
--|NOT|--: RLO の反転

説明

「RLO の反転」命令を使用し、論理演算の結果(RLO)の信号状態を反転します。 命令の入力時の信号状態が「1」の場合、命令の出力の信号状態は「0」になります。 命令の入力時の信号状態が「0」の場合、命令の出力の信号状態は「1」です。

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がリセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- オペランド「TagIn_2」および「TagIn_3」の信号状態が「1」であること。

---( )---: 割り当て

説明

「割り当て」命令を使って、指定したオペランドのビットを設定できます。 コイルの入力の論理演算(RLO)の結果が信号状態「1」の場合、指定したオペランドが信号状態「1」にセットされます。 コイルの入力における信号状態が「0」の場合、指定されたオペランドのビットが「0」にリセットされます。

この命令は、RLO に影響を与えません。 コイルの入力における RLO は、直接出力に送信されます。

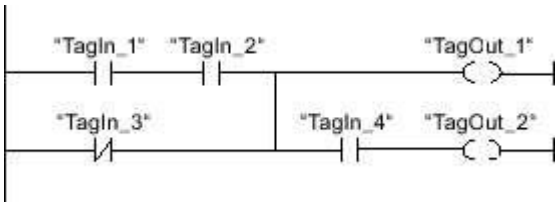
パラメータ

次の表に、「割り当て」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Output	BOOL	I、Q、M、D、L	RLO が割り当てられるオペランド。

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut_1」がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「0」であること。

次のいずれかの条件が満たされている場合、オペランド「TagOut_2」がセットされます。

- オペランド「TagIn_1」、「TagIn_2」および「TagIn_4」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「0」であり、オペランド「TagIn_4」の信号状態が「1」であること。

--(/)--: 否定割り当て

説明

「否定割り当て」命令は、論理演算の結果(RLO)を反転し、指定したオペランドに割り当てます。 コイルの入力における RLO が「1」の場合、オペランドはリセットされます。 コイルの入力における RLO が「0」の場合、オペランドは「1」にセットされます。

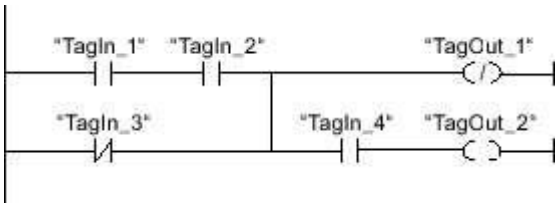
パラメータ

次の表に、「割り当てを無効にする」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Output	BOOL	I、Q、M、D、L	RLO が割り当てられるオペランド。

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut_1」がリセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「0」であること。

---(R)---: 出力のリセット

説明

「出力のリセット」命令を使って、指定したオペランドのシグナル状態を「0」にリセットできます。

この命令は、コイルの入力における論理演算の結果(RLO)が「1」の場合にのみ実行されます。 コイルへの電力潮流が発生した場合(RLO が「1」)、指定したオペランドが「0」にリセットされます。 コイルの入力における RLO が「0」の場合 (コイルへの信号流れが発生しない場合)、指定したオペランドのシグナル状態は変更されません。

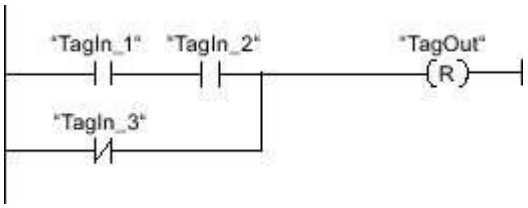
パラメータ

次の表に、「出力のリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	RLO = 「1」の場合にリセットされるオペランド。

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がリセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- オペランド「TagIn_3」のシグナル状態が「0」であること。

---(S)---: 出力のセット

説明

「出力のセット」命令を使用すると、指定したオペランドの信号状態を「1」にセットすることができます。

この命令は、コイルの入力における論理演算の結果(RLO)が「1」の場合にのみ実行されます。 コイルへの電力潮流が発生した場合(RLO が「1」)、指定したオペランドは「1」にセットされます。 コイルの入力における RLO が「0」の場合 (コイルへの信号流れが発生しない場合)、指定したオペランドの信号状態は変更されません。

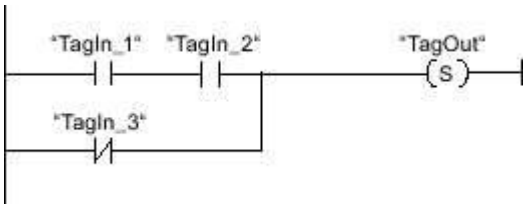
パラメータ

次の表に、「出力のセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Output	BOOL	I、Q、M、D、L	RLO =「1」を使用してセットされるオペランド

例

次の例で、命令がどのように機能するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「0」であること。

SET_BF: ビットフィールドのセット

説明

「ビットフィールドのセット」命令を使用し、特定のアドレスから開始する複数ビットを設定します。

<Operand1>の値を使用して、セット対象となるビット数を決めます。 セットする最初のビットのアドレスは、<Operand2>によって定義されます。 <Operand1>の値が、選択したバイトのビット数よりも大きい場合、次のバイトのビットがセットされます。 これらのビットは、別の命令などで明示的にリセットされるまで、セットされた状態が継続します。

この命令は、コイルの入力における論理演算の結果(RLO)が「1」の場合にのみ実行されます。 コイルの入力における RLO が「0」の場合、命令は実行されません。

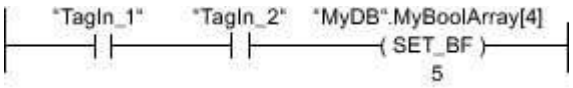
パラメータ

次の表に、「ビットフィールドのセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand2>	Output	BOOL	I、Q、M DB または IDB の場合、BOOL の ARRAY[..]のエレメント	セットする最初のビットへのポインタ
<Operand1>	Input	UINT	定数	セットするビットの数。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、オペランド "MyDB".MyBoolArray[4] のアドレスで始まる 5 つのビットがセットされます。

RESET_BF: ビットフィールドのリセット

説明

「ビットフィールドのリセット」命令を使用し、特定のアドレスから開始する複数のビットをリセットします。

<Operand1>の値を使用して、リセット対象となるビット数を決めます。 リセットする最初のビットのアドレスは、<Operand2>によって指定されます。 <Operand1>の値が、選択したバイトのビット数よりも大きい場合、次のバイトのビットがリセットされます。 これらのビットは、別の命令などで明示的にセットされるまで、リセットされた状態が継続します。

この命令は、コイルの入力における論理演算の結果(RLO)が「1」の場合にのみ実行されます。 コイルの入力における RLO が「0」の場合、命令は実行されません。

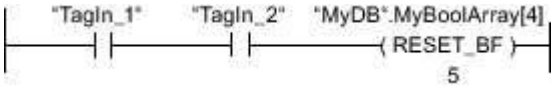
パラメータ

次の表に、「ビットフィールドのリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand2>	Output	BOOL	I、Q、M DB または IDB の場合、BOOL の ARRAY[..]のエレメント	リセットする最初のビットへのポインタ
<Operand1>	Input	UINT	定数	リセットするビットの数。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、オペランド "MyDB".MyBoolArray[4] のアドレスで始まる 5 つのビットがリセットされます。

SR: フリップフロップのセット/リセット

説明

「フリップフロップのリセット/セット」命令を使用し、入力 S および R1 の信号状態に応じて、指定されたオペランドのビットをセットまたはリセットします。 入力 S の信号状態が「1」で、入力 R1 の信号状態が「0」の場合、指定したオペランドは「1」にセットされます。 入力 S の信号状態が「0」で、入力 R1 の信号状態が「1」の場合、指定したオペランドが「0」にリセットされます。

入力 R1 が入力 S よりも優先します。入力 S および R1 の両方で信号状態が「1」の場合、指定したオペランドが信号状態「0」にリセットされます。

2 つの入力 S と R1 の信号状態が「0」の場合、この命令は実行されません。 この場合、オペランドの信号状態は変更されません。

オペランドの現在の信号状態は出力 Q に転送されるため、そこで照会できます。

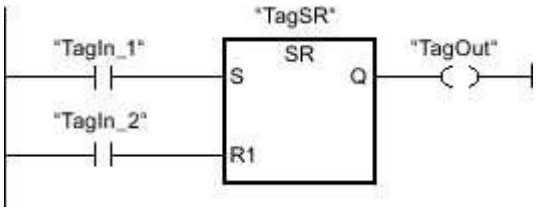
パラメータ

次の表に、「フリップフロップのセット/リセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
S	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L	セットの有効化
R1	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	リセットの有効化
<オペランド>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	セットまたはリセットされるオペランド。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	オペランドの信号状態

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、オペランド「TagSR」および「TagOut」がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- オペランド「TagIn_2」の信号状態が「0」であること。

次のいずれかの条件が満たされている場合、オペランド「TagSR」および「TagOut」がリセットされます。

- オペランド「TagIn_1」の信号状態が「0」であり、オペランド「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。

RS: フリップフロップのリセット/セット

説明

「フリップフロップのリセット/セット」命令を使って、入力 R および S1 の信号状態に基づいて指定されたオペランドのビットをセットまたはリセットします。 入力 R の信号状態が「1」で、入力 S1 の信号状態が「0」の場合、指定したオペランドが「0」にリセットされます。 入力 R の信号状態が「0」で、入力 S1 の信号状態が「1」の場合、指定したオペランドは「1」にセットされます。

入力 S1 が入力 R よりも優先します。 入力 R と S1 の信号状態がどちらも「1」の場合、指定したオペランドの信号状態が「1」にセットされます。

2 つの入力 R と S1 の信号状態が「0」の場合、この命令は実行されません。 この場合、オペランドの信号状態は変更されません。

オペランドの現在の信号状態は出力 Q に転送されるため、そこで照会できます。

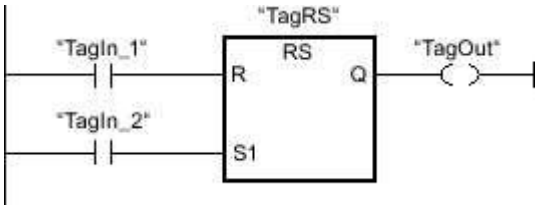
パラメータ

次の表に、「フリップフロップのリセット/セット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
R	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L	リセットの有効化
S1	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	セットの有効化
<オペランド>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	リセットまたはセットされるオペランド。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	オペランドの信号状態

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、オペランド「TagRS」および「TagOut」がリセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- オペランド「TagIn_2」の信号状態が「0」であること。

次のいずれかの条件が満たされている場合、オペランド「TagRS」および「TagOut」がセットされます。

- オペランド「TagIn_1」の信号状態が「0」であり、オペランド「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。

--|P|--:立ち上がりエッジのオペランドスキャン

説明

「立ち上がりエッジのオペランドスキャン」命令を使用すると、指定したオペランド(<Operand1>)で「0」から「1」への信号状態の切り替えが発生したかどうかを判断することができます。 この命令は、<Operand1>の現在の信号状態と、エッジメモリビット(<Operand2>)に保存された前のスキャンの信号状態を比較します。 この命令が論理演算(RLO)の結果の「0」から「1」への切り替えを検出した場合、正の信号立ち上がりエッジがあります。

立ち上がりエッジを検出した場合、命令の出力の信号状態は「1」です。 それ以外のすべての場合、この命令の出力の信号状態は「0」です。

命令の上のオペランドのプレースホルダで、照会されるオペランド(<Operand1>)を指定します。 命令の下のオペランドのプレースホルダで、エッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、ビットメモリが上書きされます。 このステップを実行するとエッジ評価に影響を与え、結果はもう固有ではありません。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

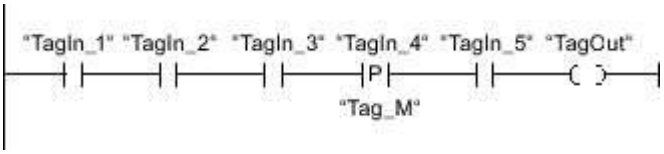
パラメータ

次の表に、「立ち上がりエッジのオペランドスキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	スキャン対象の信号
<Operand2>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	以前の照会の信号状態が保存されているエッジメモリビット

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、オペランド「TagOut」がセットされます。

- オペランド「TagIn_1」、「TagIn_2」および「TagIn_3」の信号状態が「1」であること。
- オペランド「TagIn_4」に立ち上がりエッジが発生していること。 前のスキャンの信号状態がエッジメモリビット「Tag_M」に保存されています。
- オペランド「TagIn_5」の信号状態が「1」であること。

--|N|--:立ち下りエッジのオペランドスキャン

説明

「立ち下がりエッジのオペランドスキャン」命令を使用すると、指定したオペランド(<Operand1>)で「1」から「0」への信号状態の切り替えが発生したかどうかを判断することができます。 この命令は、<Operand1>の現在の信号状態と、エッジメモリビット(<operand2>)に保存された前のスキャンの信号状態を比較します。 この命令が論理演算(RLO)の結果の「1」から「0」への切り替えを検出した場合、負の信号立ち下がりエッジがあります。

信号立ち下りエッジを検出した場合、命令の出力の信号状態は「1」です。 それ以外のすべての場合、この命令の出力の信号状態は「0」です。

命令の上のオペランドのプレースホルダで、照会されるオペランド(<Operand1>)を指定します。 命令の下のオペランドのプレースホルダで、エッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、ビットメモリが上書きされます。 このステップを実行するとエッジ評価に影響を与え、結果はもう固有ではありません。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

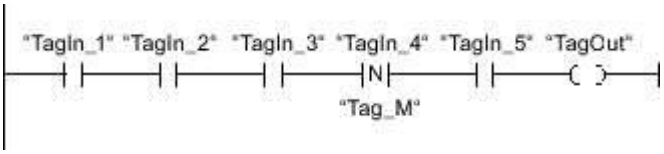
パラメータ

次の表に、「立ち下がりエッジのオペランドスキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	スキャン対象の信号
<Operand2>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	以前の照会の信号状態が保存されているエッジメモリビット

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、オペランド「TagOut」がセットされます。

- オペランド「TagIn_1」、「TagIn_2」および「TagIn_3」の信号状態が「1」であること。
- オペランド「TagIn_4」に信号立ち下りエッジが発生していること。 前のスキャンの信号状態がエッジメモリビット「Tag_M」に保存されています。
- オペランド「TagIn_5」の信号状態が「1」であること。

--(P)--: 信号立ち上がりエッジのオペランドの設定

説明

「立ち上がりエッジのオペランド設定」命令を使用し、論理演算の結果(RLO)で「0」から「1」への切り替えがあった場合に、指定したオペランド(<Operand1>)をセットすることができます。 この命令は、現在の RLO と、エッジメモリビット(<Operand2>)に保存された以前の照会からの RLO を比較します。 命令が RLO の「0」から「1」への切り替えを検出すると、信号立ち上がりエッジが存在します。

信号立ち上がりエッジが検出された場合、1 プログラムサイクルの間<Operand1>が信号状態「1」にセットされます。それ以外のすべての場合、オペランドの信号状態は「0」です。

命令の上のオペランドプレースホルダにセット対象となるオペランド(<Operand1>)を指定します。命令の下のオペランドプレースホルダにエッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、ビットメモリが上書きされます。このステップを実行するとエッジ評価に影響を与え、結果はもう固有ではありません。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

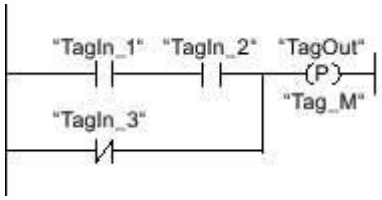
パラメータ

次の表に、「立ち上がりエッジのオペランド設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Output	BOOL	I、Q、M、D、L	立ち上がりエッジによって設定されたオペランド
<Operand2>	InOut	BOOL	I、Q、M、D、L	エッジメモリビット

例

次の例で、命令がどのように機能するかを示します。



コイルの入力における信号状態が「0」から「1」に切り替わると(信号立ち下がりエッジ)、1 回のプログラムサイクルの間にオペランド「TagOut」がセットされます。 それ以外のすべての場合、オペランド「TagOut」の信号状態は「0」です。

--(N)--: 信号立ち下がりエッジのオペランドの設定

説明

「信号立ち下がりエッジのオペランドの設定」命令を使用し、論理演算の結果(RLO)で「1」から「0」への切り替えがあった場合に、指定したオペランド(<Operand1>)をセットすることができます。 この命令は、現在の RLO と、エッジメモリビット(<Operand2>)に保存された以前の照会からの RLO を比較します。 命令が RLO の「1」から「0」への切り替えを検出すると、立ち下がりエッジが存在します。

信号立ち下がりエッジが検出された場合、1 プログラムサイクルの間<Operand1>が信号状態「1」にセットされます。それ以外のすべての場合、オペランドの信号状態は「0」です。

命令の上のオペランドプレースホルダにセット対象となるオペランド(<Operand1>)を指定します。命令の下のオペランドプレースホルダにエッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、ビットメモリが上書きされます。このステップを実行するとエッジ評価に影響を与え、結果はもう固有ではありません。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

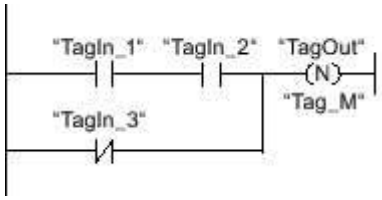
パラメータ

次の表に、「立ち下がりエッジのオペランドの設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Output	BOOL	I、Q、M、D、L	立ち下がりエッジによって設定されたオペランド
<Operand2>	InOut	BOOL	I、Q、M、D、L	エッジメモリビット

例

次の例で、命令がどのように機能するかを示します。



コイルの入力における信号状態が「1」から「0」に切り替わると(信号立ち下がりエッジ)、1 回のプログラムサイクルの間にオペランド「TagOut」がセットされます。 それ以外のすべての場合、オペランド「TagOut」の信号状態は「0」です。

P_TRIG: 立ち上がりエッジの RLO スキャン

説明

「立ち上がりエッジの RLO スキャン」命令を使用して、論理演算の結果(RLO)の信号状態での「0」から「1」への切り替えを照会します。 この命令は、RLO の現在の信号状態と、エッジメモリビット(<Operand>)に保存された前に照会された信号状態を比較します。 命令が RLO の「0」から「1」への切り替えを検出すると、信号立ち上がりエッジが存在します。

立ち上がりエッジを検出した場合、命令の出力の信号状態は「1」です。それ以外のすべての場合、この命令の出力の信号状態は「0」です。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、ビットメモリが上書きされます。このステップを実行するとエッジ評価に影響を与え、結果はもう固有ではありません。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

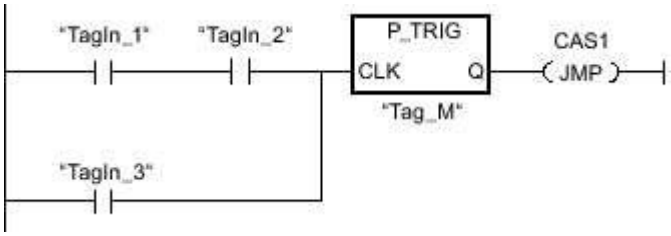
パラメータ

次の表に、「立ち上がりエッジの RLO スキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L	現在の RLO
<Operand>	InOut	BOOL	M、D	以前の照会の RLO が保存されているエッジメモリビット。
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように機能するかを示します。



以前の照会の RLO は、エッジメモリビット「Tag_M」に保存されています。 RLO の信号状態で「0」から「1」の切り替えが検出された場合、プログラムがジャンプラベル CAS1 にジャンプします。

N_TRIG: 立ち下がりエッジの RLO スキャン

説明

「立ち下がりエッジの RLO スキャン」命令を使用して、論理演算の結果(RLO)の信号状態での「1」から「0」への切り替えを照会します。 この命令は、RLO の現在の信号状態と、エッジメモリビット(<Operand>)に保存された前に照会された信号状態を比較します。 命令が RLO の「1」から「0」への切り替えを検出すると、立ち下がりエッジが存在します。

信号立ち下りエッジを検出した場合、命令の出力の信号状態は「1」です。それ以外のすべての場合、この命令の出力の信号状態は「0」です。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、ビットメモリが上書きされます。このステップを実行するとエッジ評価に影響を与え、結果はもう固有ではありません。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

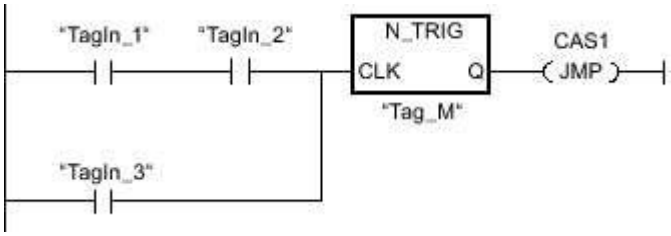
パラメータ

次の表に、「立ち下がりエッジの RLO スキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L	現在の RLO
<Operand>	InOut	BOOL	M、D	以前の照会の RLO が保存されているエッジメモリビット。
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように機能するかを示します。



以前の照会の RLO は、エッジメモリビット「Tag_M」に保存されています。 RLO の信号状態で「1」から「0」の切り替えが検出された場合、プログラムがジャンプラベル CAS1 にジャンプします。

R_TRIG: 信号立ち上がりエッジを検出

説明

「信号立ち上がりエッジの検出」命令を使用して、CLK 入力で「0」から「1」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「0」から「1」への状態変化を検出すると、Q 出力で信号立ち上がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

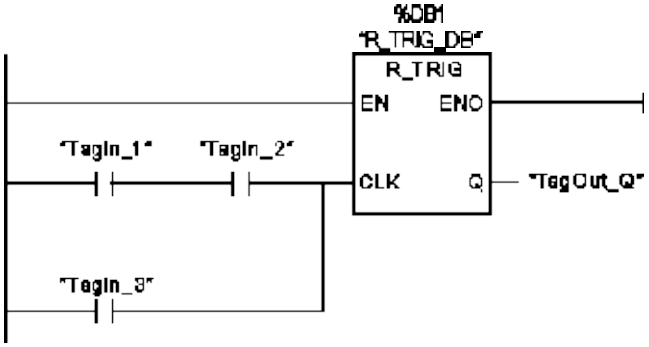
パラメータ

次の表に、「信号立ち上がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
CLK	Input	BOOL	I、Q、M、D、L、または定数	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。



CLK 入力のタグでの前の状態は、「R_TRIG_DB」タグに保存されます。「0」から「1」へのシグナル状態の変化が「TagIn_1」および「TagIn_2」オペランド または「TagIn_3」オペランドで検出されると、「TagOut_Q」出力のシグナル状態は 1 サイクルについて「1」になります。

F_TRIG: 信号立ち下がりエッジの検出

説明

「信号立ち下がりエッジの検出」命令を使用して、CLK 入力で「1」から「0」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「1」から「0」への状態変化を検出すると、Q 出力で信号立ち下がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

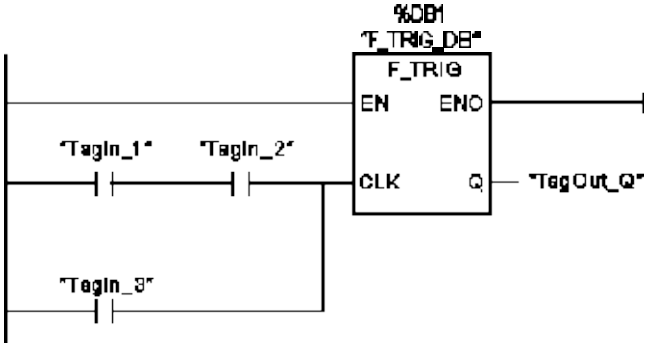
パラメータ

次の表に、「信号立ち下がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
CLK	Input	BOOL	I、Q、M、D、L、または定数	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。



CLK 入力のタグでの前の状態は、「F_TRIG_DB」タグに保存されます。 「1」から「0」へのシグナル状態の変化が「TagIn_1」および「TagIn_2」オペランド または「TagIn_3」オペランドで検出されると、「TagOut_Q」出力のシグナル状態は 1 サイクルについて「1」になります。

タイマ

この章には下記に関する情報が記載されています：

TP: パルスの生成

説明

「パルス生成」命令を使用して、プログラムされた持続時間の出力 Q を設定できます。 入力 IN で論理演算の結果(RLO)が「0」から「1」(立ち上がりエッジ)に切り替わると、命令が開始します。 命令が開始されると、プログラムされた時間 PT が開始します。 その後の入力信号に関係なく、出力 Q が持続時間 PT の間セットされます。 新規の立ち上がりエッジが検出された場合でさえも、PT 持続時間が動作している限り、出力 Q は影響されません。

ET 出力での現在の時間値をスキャンできます。 時間値は、T#0s において開始し、長さの値 PT に達すると終了します。 持続時間 PT が経過して入力 IN の信号状態が「0」の場合、ET 出力がリセットされます。

「パルス生成」命令の各呼び出しは、命令データが保存される IEC タイマに割り当てる必要があります。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、ET 出力はタイマの期限が切れるとすぐに定数値を返します。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TP_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TP_TIME、TP_LTIME、または IEC_TIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TP_TIME、TP_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「パルス生成」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

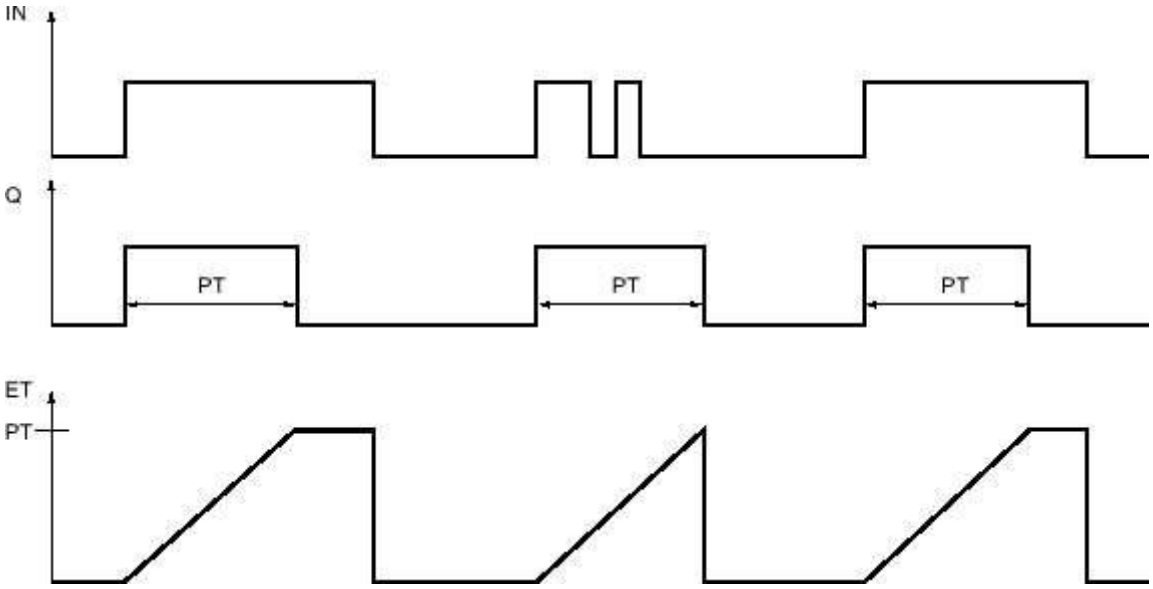
次の表に、「パルス生成」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	パルスの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	パルス出力
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「パルスタイマ」命令のパルスタイミング図を示します。



TON: オンディレータイマ

説明

「オンディレータイマ」命令を使用し、プログラムされた持続時間 PT によって Q 出力のセットを遅延することができます。 入力 IN で論理演算の結果(RLO)が「0」から「1」(立ち上がりエッジ)に切り替わると、命令が開始します。 命令が開始されると、プログラムされた時間 PT が開始します。 持続時間 PT が経過すると、出力 Q の信号状態は「1」になります。 出力 Q は、開始入力が「1」の間はセットされた状態が続きます。 開始入力の信号状態が「1」から「0」に切り替わると、Q 出力がリセットされます。 開始入力で信号の新しい立ち上がりエッジが検出されると、タイマファンクションが再開されます。

現在の時間値は、ET 出力で照会できます。 時間値は、T#0s において開始し、長さの値 PT に達すると終了します。 IN 入力が信号状態「0」に切り替わると、ET 出力が直ちにリセットされます。

「オンディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、ET 出力はタイマの期限が切れるとすぐに定数値を返します。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TON_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TON_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TON_TIME、TON_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「オンディレータイマ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

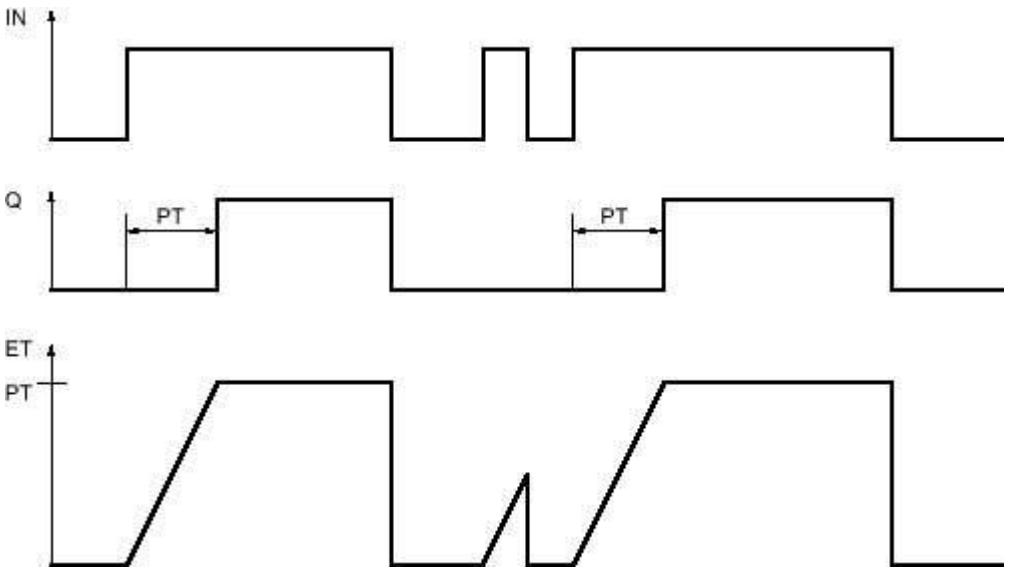
次の表に、「オンディレータイマ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	オンディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	時間 PT の経過時にセットされる出力。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オンディレータイマ」命令のパルスタイミング図を示します。



TOF: オフディレータイマ

説明

「オフディレータイマ」命令を使用し、プログラムされた持続時間 PT によって Q 出力のリセットを遅延することができます。 Q 出力は、入力 IN で論理演算の結果(RLO)が「0」から「1」に切り替わった場合(立ち上がりエッジ)に設定されます。 入力 IN が信号状態「0」に戻ると、プログラムされた時間 PT が開始します。 長さ PT が実行している限り、出力 Q はセットされた状態が継続します。 持続時間 PT が経過すると、Q 出力がリセットされます。 持続時間 PT が経過する前に入力 IN の信号状態が「1」に切り替わると、タイマがリセットされます。 出力 Q の信号状態は、「1」のままになります。

現在の時間値は、ET 出力で照会できます。 時間値は、T#0s において開始し、長さの値 PT に達すると終了します。 持続時間 PT が経過すると、入力 ETIN が「1」に戻るまで、出力は現在値にセットされた状態が継続します。 持続時間 PT が経過する前に入力 IN が「1」に切り替わると、ET 出力が値 T#0s にリセットされます。

「オフディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、ET 出力はタイマの期限が切れるとすぐに定数値を返します。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TOF_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TOF_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TOF_TIME、TOF_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「オフディレータイマ」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

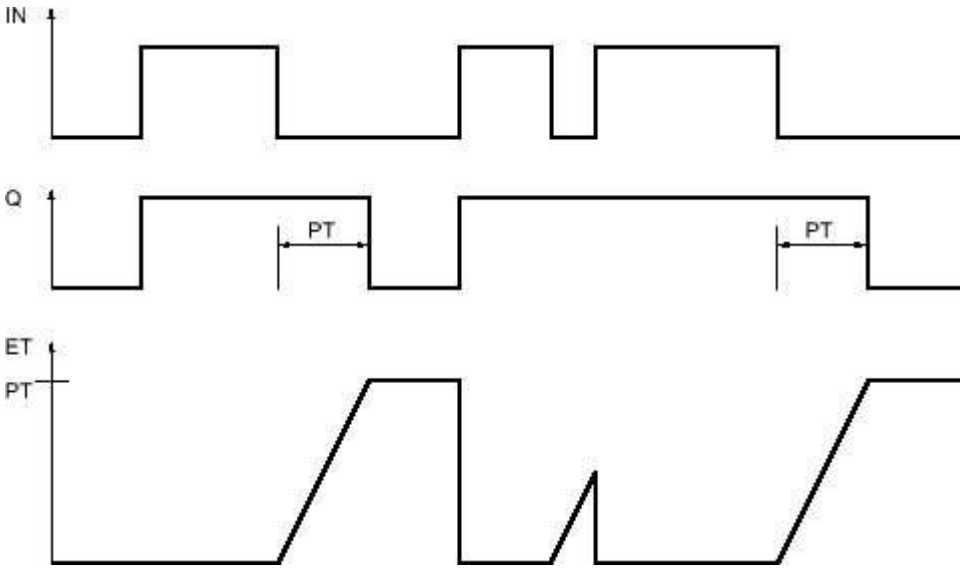
次の表に、「オフディレータイマ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	オフディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	タイマ PT の期限が切れたときにリセットされる出力。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オフディレータイマ」命令のパルスタイミング図を示します。



TONR: タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用して、PT パラメータによってセットされた時間内の時間値を累積します。 入力 IN のシグナル状態が「0」から「1」に切り替わると(信号立ち上がりエッジ)、命令が実行され、時間 PT が開始します。 持続時間 PT が経過している間、IN 入力のシグナル状態が「1」のときに記録された時間値が累積されます。 累積時間は、出力 ET に書き込まれ、そこで照会できます。 持続時間 PT の期限が切れると、出力 Q のシグナル状態は「1」です。 IN パラメータのシグナル状態が「1」から「0」(立ち下がりエッジ)に切り替わった場合でも、Q パラメータは「1」にセットされた状態が継続します。

開始入力のシグナル状態に関係なく、R 入力が出力 ET と Q をリセットします。

「タイムアキュムレータ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TONR_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TONR_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TONR_TIME、TONR_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが自動的に開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「タイムアキュムレータ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

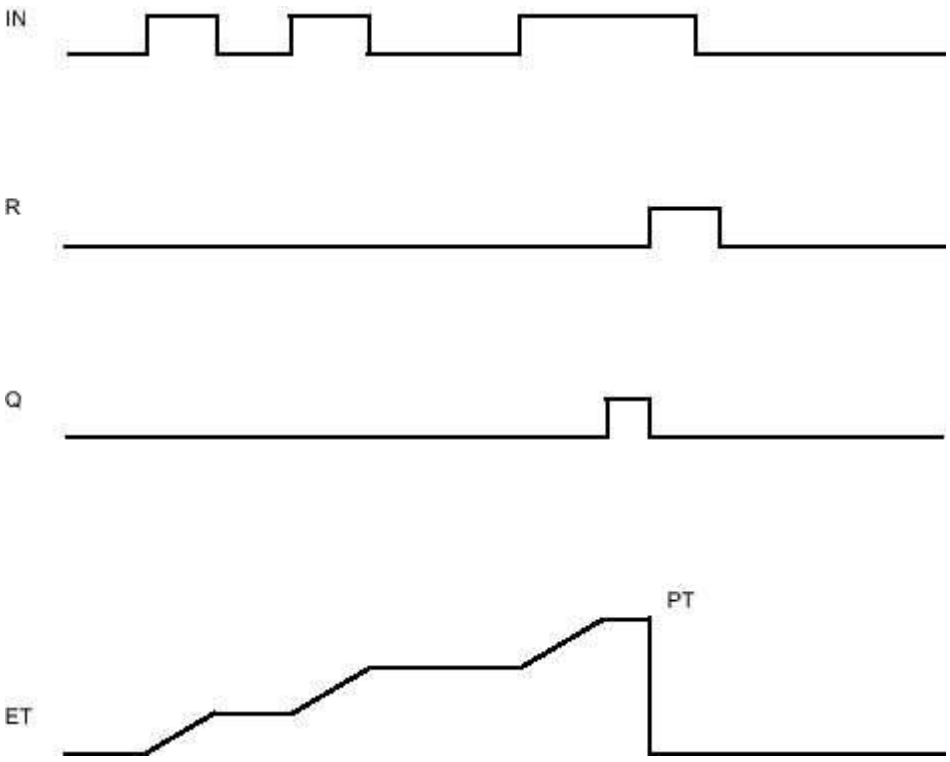
次の表に、「タイムアキュムレータ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	開始入力
R	Input	BOOL	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	リセット入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	時間記録の最大持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	時間 PT の経過時にセットされる出力。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	累積時間

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「タイムアキュムレータ」命令のパルスタイミング図を示します。



---(TP)---:パルスタイマの起動

説明

「パルスタイマの起動」命令を使用して、パルスとして指定された持続時間分、IEC タイマを起動します。 IEC タイマは、論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がりエッジ)に起動されます。 RLO がその後に切り替わるかどうかに関係なく、IEC タイマは指定された持続時間分、実行されます。 IEC タイマを実行しても、新らしい立ち上がりエッジの検出に影響ありません。 IEC タイマを実行している限り、タイマステータスが「1」であるか問い合わせると、信号状態が「1」に戻ります。 IEC タイマの期限が切れると、タイマステータスが信号状態「0」に戻ります。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、さまざまな実行レベルで開始し、IEC タイマを照会することが可能です。

S7-1200 CPU の場合

「パルスタイマの起動」命令は、データタイプ IEC_TIMER または TP_TIME の構造体にデータを保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TP_LTIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「パルスタイマの起動」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体にデータを保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TP_TIME、TP_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および割り当てられた IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは IEC タイマの Q 構造体コンポーネントに保存されます。 ノーマルオープンを使用してタイマ状態が「1」かどうかを問い合わせ、ノーマルクローズを使用して「0」かどうかを問い合わせることができます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「パルスタイマの起動」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「パルスタイマの起動」命令のパラメータを示します。

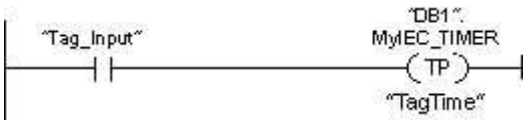
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<持続時間>	Input	TIME	TIME, LTIME	I、Q、M、D、L または定数	IEC タイマが動作する時間。

<IEC タイマ>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D、L	起動される IEC タイマ。

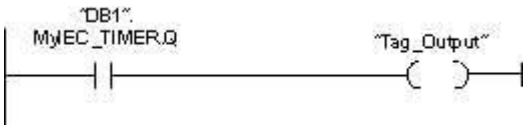
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「パルスタイマの起動」命令は、オペランド Tag_Input の信号状態が「0」から「1」に切り替わると実行されます。 タイマ "DB1".MyIEC_TIMER は、オペランド「TagTime」内に格納された時間で開始されます。



タイマ「DB1」。 MyIEC_TIMER が実行中の間は、タイマステータス("DB1".MyIEC_TIMER.Q)の信号状態は「1」であり、オペランド「Tag_Output」がセットされます。 IEC タイマの期限が切れると、時間ステータスが信号状態「0」に戻り Tag_Output、オペランドがリセットされます。

---(TON)---:オンディレイタイマの起動

説明

「オンディレータイマの起動」命令を使用して、オンディレーとして指定された持続時間分、IEC タイマを起動します。 IEC タイマは、論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がりエッジ)に起動されます。 IEC タイマは指定された時間、実行します。 命令の入力時の RLO の信号状態が「1」の場合、出力は信号状態は「1」を返します。 期限が切れる前に RLO が「0」に切り替わる場合、IEC タイマがリセットされます。 この場合、タイマステータスが「1」であるか問い合わせると、信号状態が「0」に戻ります。 命令の入力において次の立ち上がりエッジを検出すると、IEC タイマが再起動します。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、さまざまな実行レベルで開始し、IEC タイマを照会することが可能です。

S7-1200 CPU の場合

「オンディレータイマの起動」命令がそのデータをデータタイプ IEC_TIMER または TON_TIME の構造体に保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TON_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「オンディレータイマの起動」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体にデータを保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TON_TIME、TON_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および割り当てられた IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは IEC タイマの ET 構造体コンポーネントに保存されます。ノーマルオープンを使用してタイマ状態が「1」かどうかを問い合わせ、ノーマルクローズを使用して「0」かどうかを問い合わせることができます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「オンディレータイマの起動」命令の実行には、あらかじめ論理演算が必要です。これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「オンディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<持続時間>	Input	TIME	TIME, LTIME	I、Q、M、D、L または定数	IEC タイマが動作する時間。
<IEC タイマ>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D、L	起動される IEC タイマ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

「オンディレータイマの起動」命令は、オペランド「Tag_Input」の信号状態が「0」から「1」に切り替わると実行されます。 「MyIEC_TIMER」タイマは、オペランド「TagTime」内に格納された時間で開始されます。

タイマ「MyIEC_TIMER」の期限が切れ、オペランド「Tag_Input」の信号状態が「1」の場合、タイマステータス("MyIEC_TIMER).Q を照会すると、信号状態「1」が返され、オペランド「Tag_Output」が設定されます。 オペランド「Tag_Input」が信号状態「0」に切り替わった場合、タイマステータスを照会すると信号状態「0」が返され、オペランド「Tag_Output」がリセットされます。

---(TOF)---:オフディレイタイマの起動

説明

「オフディレータイマの起動」命令を使用して、オンディレーとして指定された持続時間分、IEC タイマを起動します。 命令の入力での論理演算の結果(RLO)の信号状態が「1」の場合、「1」のタイマステータスの照会は、信号状態「0」を返します。 RLO が「1」から「0」(立ち下がりエッジ)に切り替わると、指定された計測時間の IEC タイマが起動します。 IEC タイマが実行している限り、タイマステータスの信号状態は「1」のままです。 タイマの期限が切れ、命令の入力の RLO の信号状態が「0」の場合、タイマステータスは信号状態「0」にセットされます。 期限が切れる前に RLO が「1」に切り替わる場合、IEC タイマがリセットされ、タイマステータスの信号状態は「1」のままになります。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、さまざまな実行レベルで開始し、IEC タイマを照会することが可能です。

S7-1200 CPU の場合

「オンディレータイマの起動」命令がそのデータをデータタイプ IEC_TIMER または TOF_TIME の構造体に保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TOF_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「オフディレータイマの起動」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体にデータを保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TOF_TIME、TOF_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および割り当てられた IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは IEC タイマの ET 構造体コンポーネントに保存されます。ノーマルオープンを使用してタイマ状態が「1」かどうかを問い合わせ、ノーマルクローズを使用して「0」かどうかを問い合わせることができます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「オフディレータイマの起動」命令の実行には、あらかじめ論理演算が必要です。これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「オフディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<持続時間>	Input	TIME	TIME, LTIME	I、Q、M、D、L または定数	IEC タイマが動作する時間。
<IEC タイマ>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTIMER, TOF_TIME, TOF_LTIME	D、L	起動される IEC タイマ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

「オフディレータイマの起動」命令は、オペランド「Tag_Input」の信号状態が「1」から「0」に切り替わると実行されます。 タイマ#MyIEC_TIMER は、オペランド「TagTime」内に格納された時間で開始されます。

タイマ#MyIEC_TIMER が作動している限り、時間のステータス(#MyIEC_TIMER.Q)を照会すると信号状態「1」が返され、オペランド「Tag_Output」が設定されます。タイマの期限が切れ、オペランド「Tag_Input」が信号状態「0」になった場合、タイマステータスを照会すると信号状態「0」が返されます。 オペランド「Tag_Input」の信号状態が「1」に切り替わってからタイマ#MyIEC_TIMER の期限が切れる場合、タイマはリセットされます。オペランド「Tag_Input」の信号状態が「1」の場合、タイマステータスを照会すると信号状態「1」が返されます。

---(TONR)---:タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用し、命令「1」の入力時の信号の長さを記録することが可能です。 この命令は、入力での論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がり工ッジ)に開始されます。 時間は、RLO が「1」である限り記録されます。 RLO が「0」に切り替わると、この命令は中断します。 RLO が「1」に戻ると、時間の記録が継続されます。 記録された時間が指定された持続時間値を超え、コイルの入力での RLO が「1」の場合、「1」のタイマステータスを照会すると信号状態「1」が返されます。

「タイマのリセット」命令を使って、タイマステータスおよび現在期限切れのタイマを「0」にリセットできます。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、さまざまな実行レベルで開始し、IEC タイマを照会することが可能です。

S7-1200 CPU の場合

「タイムアキュムレータ」命令がそのデータをデータタイプ IEC_TIMER または TONR_TIME の構造体に保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TONR_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「タイムアキュムレータ」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体にデータを保存します。構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TONR_TIME、TONR_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および割り当てられた IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは IEC タイマの ET 構造体コンポーネントに保存されます。ノーマルオープンを使用してタイマ状態が「1」かどうかを問い合わせ、ノーマルクローズを使用して「0」かどうかを問い合わせることができます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「タイムアキュムレータ」命令を実行するには、その前に論理演算を行う必要があります。これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「タイムアキュムレータ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<持続時間>	Input	TIME	TIME, LTIME	I、Q、M、D、L または定数	IEC タイマが動作する時間。
<IEC タイマ>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D、L	起動される IEC タイマ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

「タイムアキュムレータ」命令は RLO の立ち上がり エッジで実行されます。オペランド「Tag_Input」の信号状態が「1」である限り、時間が記録されます。

記録された時間がオペランド「TagTime」の値を超えている場合、タイマステータス("MyIEC_TIMER".Q)を照会すると信号状態「1」が返され、オペランド「Tag_Output」がセットされます。

---(RT)---:タイマのリセット

説明

「タイマのリセット」命令を使用し、IEC タイマを「0」にリセットすることが可能です。 この命令は、コイルの入力における論理演算の結果(RLO)が「1」の場合にのみ実行されます。 電流がコイルに流れる場合(RLO は「1」)、指定されたデータブロックにおけるタイマの構造体コンポーネントが「0」にリセットされます。 命令の入力における RLO が「0」の場合、タイマは未変更のままです。

この命令は、RLO に影響を与えません。 コイルの入力における RLO は、直接コイルの出力に送信されます。

プログラムで宣言された IEC タイマに「タイマのリセット」命令を割り当てます。

命令のデータは命令が呼び出されるときのみ更新され、割り当てられた IEC タイマがアクセスされるたびには更新されません。データを問い合わせる場合、命令の呼び出しから次の命令の呼び出しまでのみ同一になります。

パラメータ

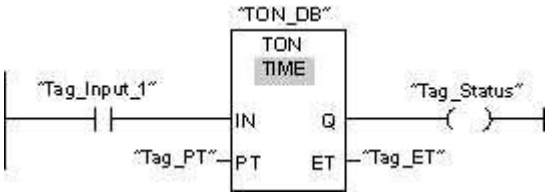
次の表に、「タイマのリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<IEC タイマ>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	リセットされる IEC タイマ

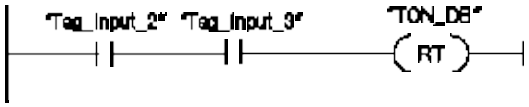
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「オンディレー生成」命令は、オペランド「Tag_Input_1」の信号状態が「0」から「1」に切り替わると実行されます。 「TON_DB」インスタンスデータブロックに保存されたタイマは、オペランド Tag_PT によって指定された時間の長さ分、実行を開始します。



オペランド「Tag_Input_2」と「Tag_Input_3」の信号状態が「1」の場合、「タイマのリセット」命令が実行され、「TON_DB」データブロックに保存されたタイマがリセットされます。

---(PT)---: 時間の長さをロード

説明

「時間の長さをロード」命令を使用して、IEC タイマの時間を設定することが可能です。 命令の入力において、論理演算の結果(RLO)の信号状態が「1」の場合、サイクルごとにこの命令を実行します。この命令を実行すると、指定された IEC タイマの構造体に指定された時間を書き込みます。

注記
命令の実行中に、指定された IEC タイマが動作している場合、指定された IEC タイマの現在の時刻が上書きされます。 これによって、IEC タイマのタイマステータスを変更できます。

プログラムで宣言した IEC タイマを「時間の長さをロード」命令に割り当てます。

命令のデータは命令が呼び出されるとき、および割り当てられた IEC タイマがアクセスされるたびにのみ更新されます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

パラメータ

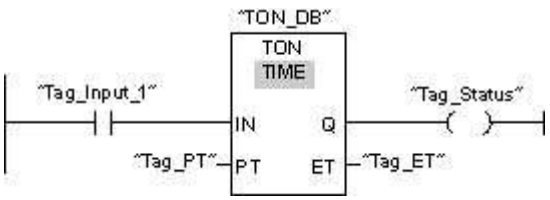
次の表に、「時間の長さをロード」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<持続時間>	Input	TIME	TIME, LTIME	I、Q、M、D、L または定数	IEC タイマが動作する時間。
<IEC タイマ>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	持続時間を設定する IEC タイマ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「オンディレー生成」命令は、オペランド「Tag_Input_1」の信号状態が「0」から「1」に切り替わると実行されます。インスタンスデータブロック「TON_DB」に保存された IEC タイマは、オペランド「Tag_PT」で指定された持続時間で起動します。

「時間の長さをロード」命令は、オペランド「Tag_Input_2」の信号状態が「1」の場合に実行されます。この命令は、インスタンスデータブロック「TON_DB」に持続時間「Tag_PT_2」を書き込み、同時にデータブロック内のオペランド「Tag_PT」の値を上書きします。 タイマステータスの信号状態は、次の照会時、または "MyTimer".Q または "MyTimer"."ET へのアクセス時に変化する可能性があります。

レガシー

この章には下記に関する情報が記載されています：

S_PULSE: パルスタイマパラメータの割り当てと起動

説明

「パルスタイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S の信号状態が「1」になると、タイマでプログラミングされた時間(TV)の期限がすぐに切れます。 プログラミングされた時間の期限が切れる前に、入力 S が信号状態「0」に切り替わった場合、タイマが停止します。 この場合、出力 Q の信号状態は「0」です。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。 現在の時間値は、出力 BI で BI コード化され、出力 BCD で BCD コード化されます。

タイマが動作していて、かつ入力 R の信号状態が「1」に切り替わった場合、現在の時間値およびタイムベースともに 0 にセットされます。 タイマが動作していない場合、R 入力の信号状態が「1」であっても効果はありません。

「パルスタイマパラメータの割り当てと開始」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

命令データは、アクセスのたびに更新されます。このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「パルスタイマパラメータの割り当てと開始」命令のパラメータを示します。

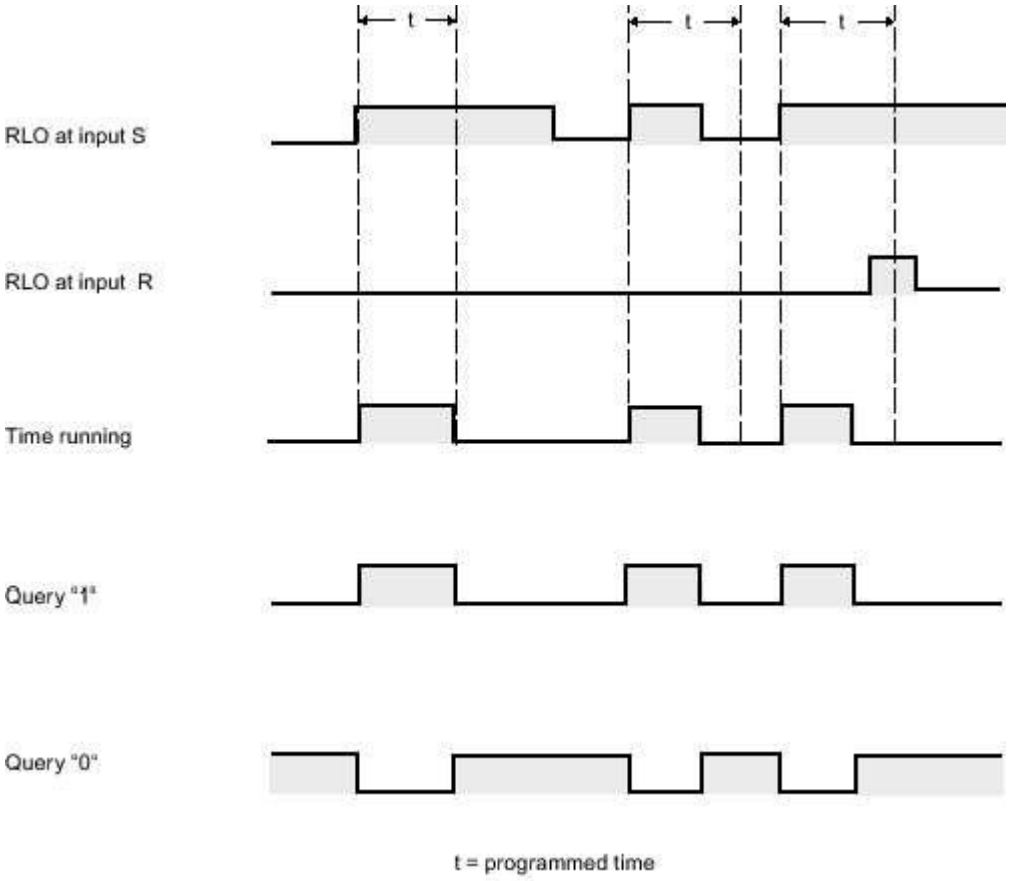
パラメータ	宣言	データタイプ	メモリ領域	説明
<タイマ>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	プリセット時間値
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(BI コード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

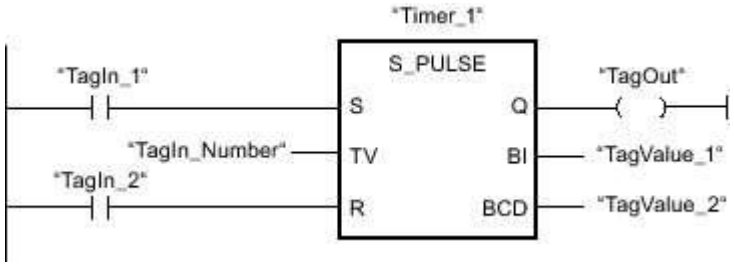
パルスタイミング図

下図は、命令「パルスタイマパラメータの割り当てと開始」のパルスタイミング図です。



例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 オペランド「TagIn_1」の信号状態が「1」である限り、タイマはオペランド「TagIn_Number」

の時間値によって期限が切れます。 オペランド「TagIn_1」の信号状態が、タイマの期限が切れる前に「1」から「0」に切り替わった場合、タイマ「Timer_1」が停止します。 この場合、オペランド「TagOut」が「0」にリセットされます。

タイマが実行中であり、オペランド「TagIn_1」の信号状態が「1」である限り、オペランド「TagOut」の信号状態は「1」です。 タイマの期限が切れるか、またはリセットされると、オペランド「TagOut」が「0」にリセットされます。

S_PEXT: 拡張パルスタイマパラメータを割り当てと起動 

説明

「拡張パルスタイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S が信号状態「0」に切り替わっても、タイマでプログラムされた持続時間(TV)の期限が切れます。 タイマが実行中である限り、出力 Q の信号状態は「1」です。 タイマの期限が切れると、出力 Q が「0」にリセットされます。 タイマの実行中に入力 S の信号状態が「0」から「1」に切り替わると、タイマは入力 TV でプログラミングされた時間によって再開されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。 現在の時間値は、出力 BI で BI コード化され、出力 BCD で BCD コード化されます。

タイマが動作していて、かつ入力 R の信号状態が「1」に切り替わった場合、現在の時間値およびタイムベースともに 0 にセットされます。 タイマが動作していない場合、R 入力の信号状態が「1」であっても効果はありません。

「拡張パルスタイマパラメータの割り当てと開始」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

命令データは、アクセスのたびに更新されます。このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「拡張パルスタイマパラメータの割り当てと開始」命令のパラメータを示します。

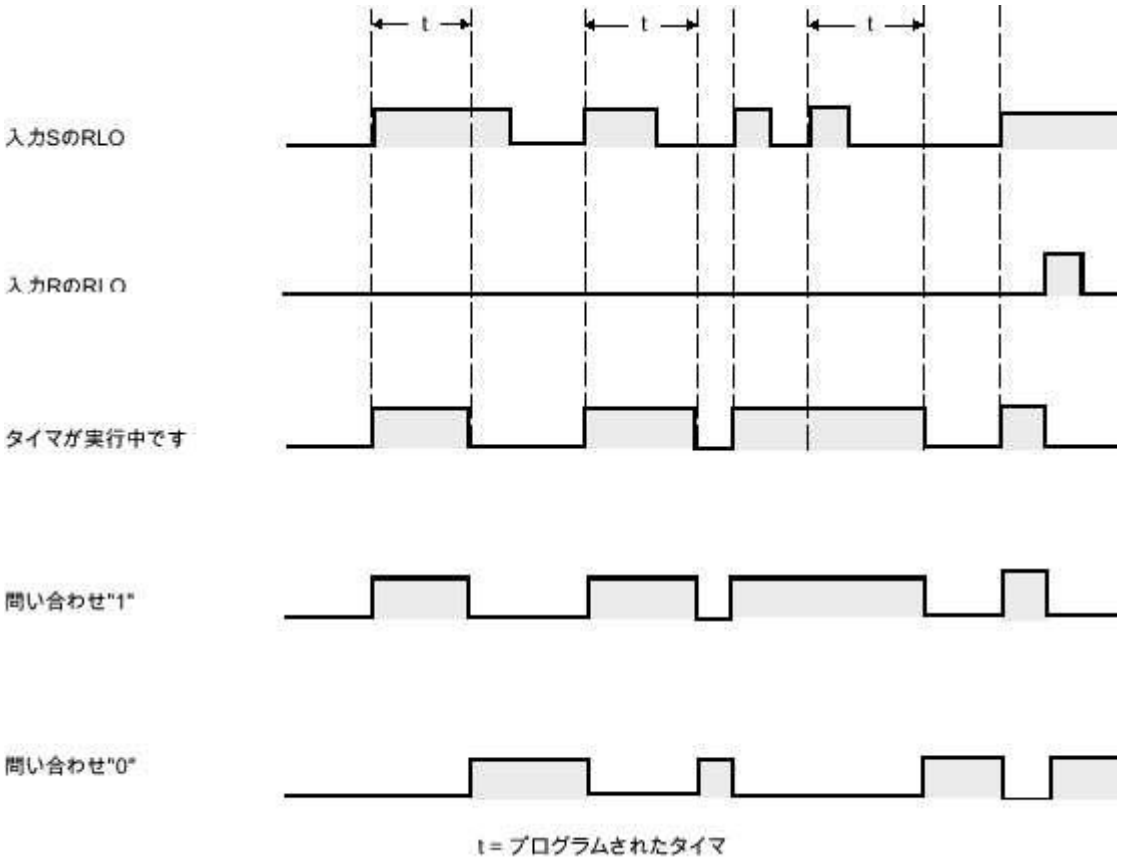
パラメータ	宣言	データタイプ	メモリ領域	説明
<タイマ>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	プリセット時間値
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(BI コード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

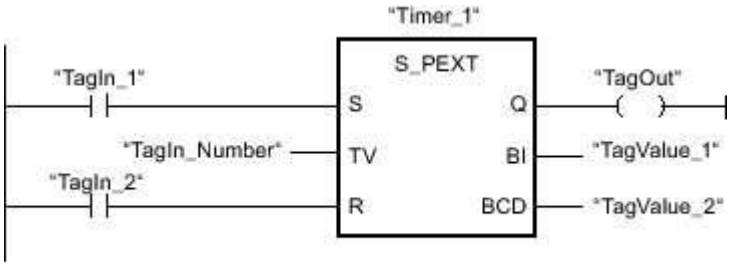
パルスタイミング図

次の図に、「拡張パルスタイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 タイマは、入力 S での立ち下がり工ッジによる影響を受けずに、オペランド「TagIn_Number」の時間値によって期限が切れます。 オペランド「TagIn_1」の信号状態が、タイマの期限が切れる前に「0」から「1」に切り替わった場合、タイマは再開します。

タイマが実行中である限り、オペランド「TagOut」の信号状態は「1」です。 タイマの期限が切れるか、またはリセットされると、オペランド「TagOut」が「0」にリセットされます。

S_ODT: オフディレイタイマパラメータの割り当てと起動

説明

「オンディレータイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S の信号状態が「1」になると、タイマでプログラミングされた時間(TV)の期限がすぐに切れます。 タイマの時間が正常に経過し、さらに入力 S の信号状態が「1」のままである場合、出力 Q が信号状態「1」を返します。 タイマの実行中に入力 S の信号状態が「1」から「0」に切り替わると、タイマは停止します。 この場合、出力 Q が信号状態「0」にリセットされます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。 現在の時間値は、出力 BI で BI コード化され、出力 BCD で BCD コード化されます。

タイマが動作していて、かつ入力 R の信号状態が「0」から「1」に切り替わった場合、現在の時間値およびタイムベースともに 0 にセットされます。 この場合、出力 Q の信号状態は「0」です。 タイマが実行中でなく、入力 S の RLO が「1」の場合でも、R 入力の信号状態が「1」の場合、タイマがリセットされます。

ボックスの上のプレースホルダーで、命令のタイマを指定します。 タイマは、データタイプ TIMER で宣言する必要があります。

「オンディレータイマパラメータの割り当てと開始」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

命令データは、アクセスのたびに更新されます。このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「オンディレータイマパラメータの割り当てと開始」命令のパラメータを示します。

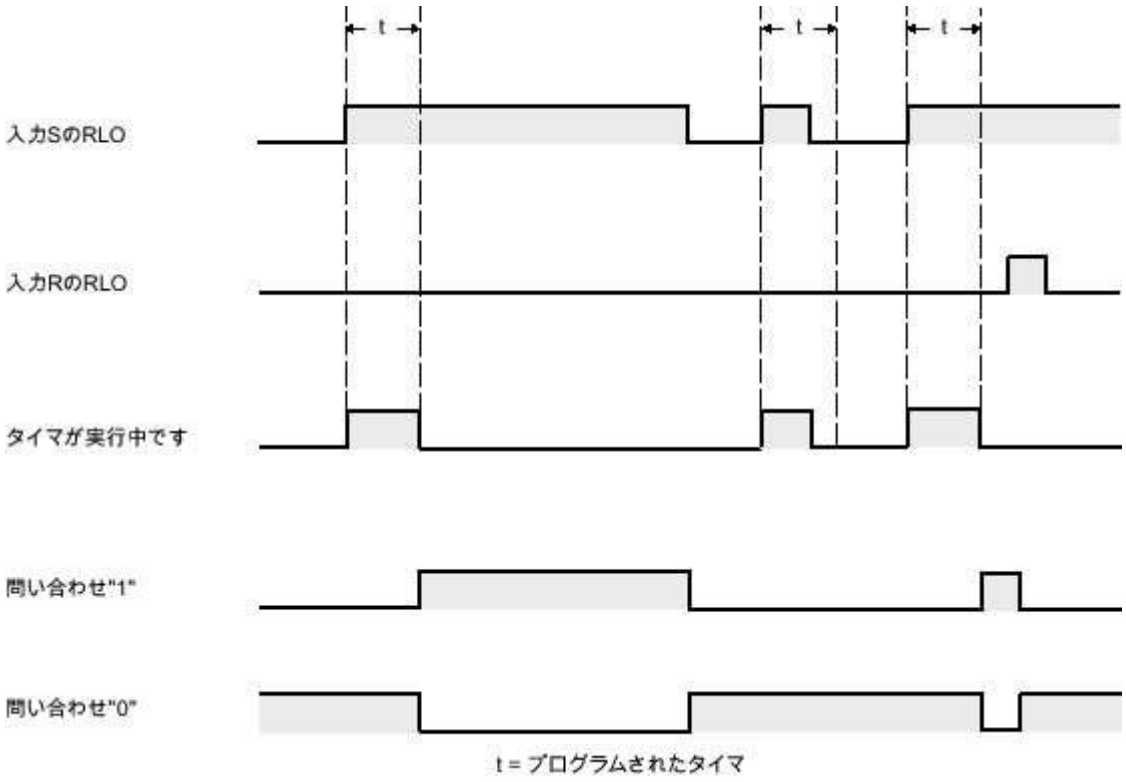
パラメータ	宣言	データタイプ	メモリ領域	説明
<タイマ>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	プリセット時間値

R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(BI コード)
BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

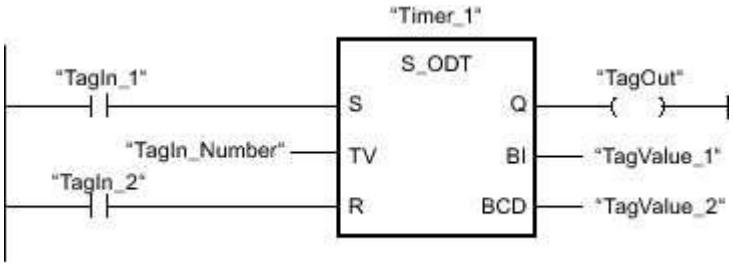
パルスタイミング図

次の図に、「オンディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 オペランド「TagIn_Number」の時間値によって、タイマの期限が切れます。 タイマの期限が切れてオペランドの信号状態が「1」の場合、オペランド「TagOut」が「1」にセットされます。 オペランド「TagIn_1」の信号状態が、タイマの期限が切れる前に「1」から「0」に切り替わった場合、タイマは停止します。 この場合、オペランド「TagOut」の信号状態は「0」です。

S_ODTS: 保持型オンディレイタイマパラメータの割り当てと起動

説明

「保持型オンディレータイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S が信号状態「0」に切り替わっても、タイマでプログラムされた持続時間(TV)の期限が切れます。 タイマの期限が切れた場合、「Q」出力は、入力「S」での信号状態に関わらず信号状態「1」を返します。 タイマの実行中に入力 S の信号状態が「0」から「1」に切り替わると、タイマは入力(TV)でプログラミングされた時間によって再開されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。 現在の時間値は、出力 BI で BI コード化され、出力 BCD で BCD コード化されます。

入力 R の信号状態「1」は、開始入力 S での信号状態に関わらず、現在の時間値およびタイムベースを「0」にリセットします。 この場合、出力 Q の信号状態は「0」です。

「保持型オンディレータイマパラメータの割り当てと開始」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

命令データは、アクセスのたびに更新されます。このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「保持型オンディレータイマパラメータの割り当てと開始」命令のパラメータを示します。

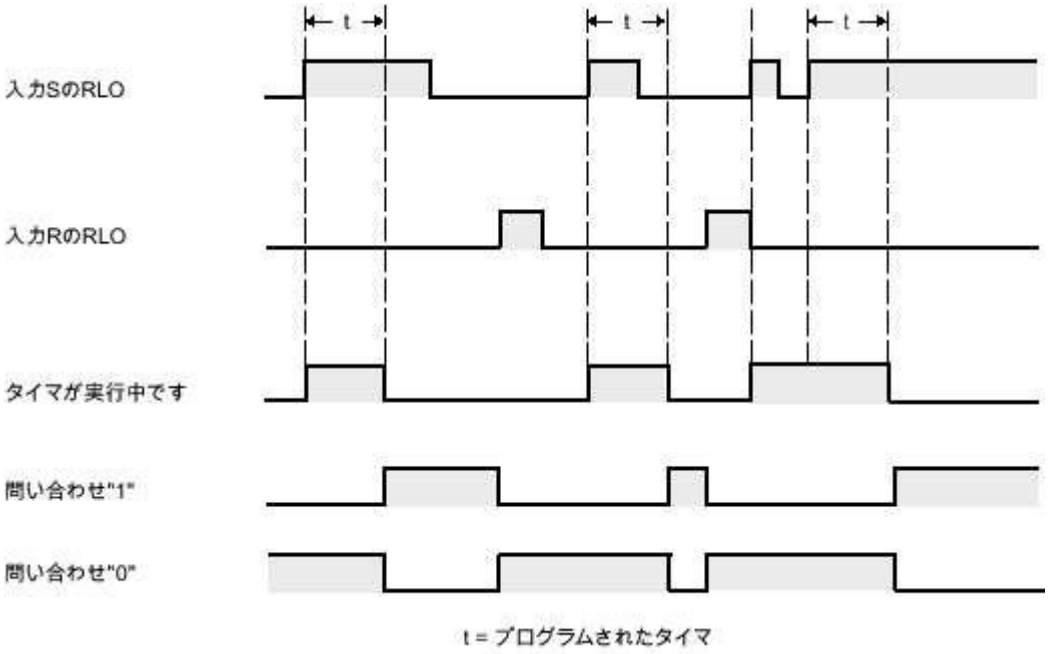
パラメータ	宣言	データタイプ	メモリ領域	説明
<タイマ>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	プリセット時間値
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(BI コード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

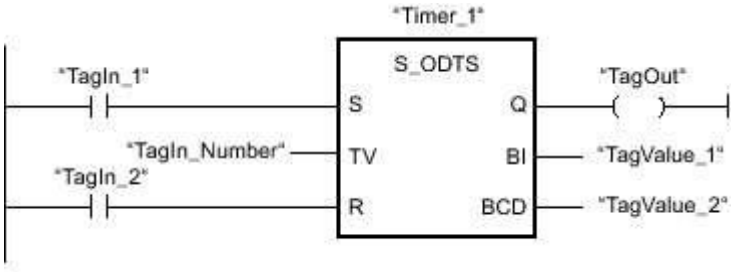
パルスタイミング図

次の図に、「保持型オンディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 オペランド「TagIn_1」が信号状態「0」に切り替わっても、タイマはオペランド「TagIn_Number」の時間値によって期限が切れます。 タイマの期限が切れると、オペランド「TagOut」が「1」にセットされます。 タイマの実行中にオペランド「TagIn_1」の信号状態が「0」から「1」に切り替わった場合、タイマは再開します。

S_OFFDT: オフディレイタイマパラメータの割り当てと起動

説明

「オフディレータイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「1」から「0」への切り替え(信号立ち下がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、プログラムされた時間(TV)によって期限が切れます。 タイマが動作している場合、または入力 S が信号状態「1」を返す場合は、入力 Q の信号状態は「1」となります。 タイマの期限が切れ、入力 S の信号状態が「0」の場合、出力 Q が信号状態「0」にリセットされます。 タイマの実行中に入力 S の信号状態が「0」から「1」に切り替わると、タイマは停止します。 タイマは、入力 S で信号の立ち下がりエッジが検出された後にのみ再開されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。 現在の時間値は、出力 BI で BI コード化され、出力 BCD で BCD コード化されます。

入力 R での信号状態「1」は、現在の時間値およびタイムベースを「0」にリセットします。 この場合、出力 Q の信号状態は「0」です。

「オフディレータイマパラメータの割り当てと開始」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

命令データは、アクセスのたびに更新されます。このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「オフディレータイマパラメータの割り当てと開始」命令のパラメータを示します。

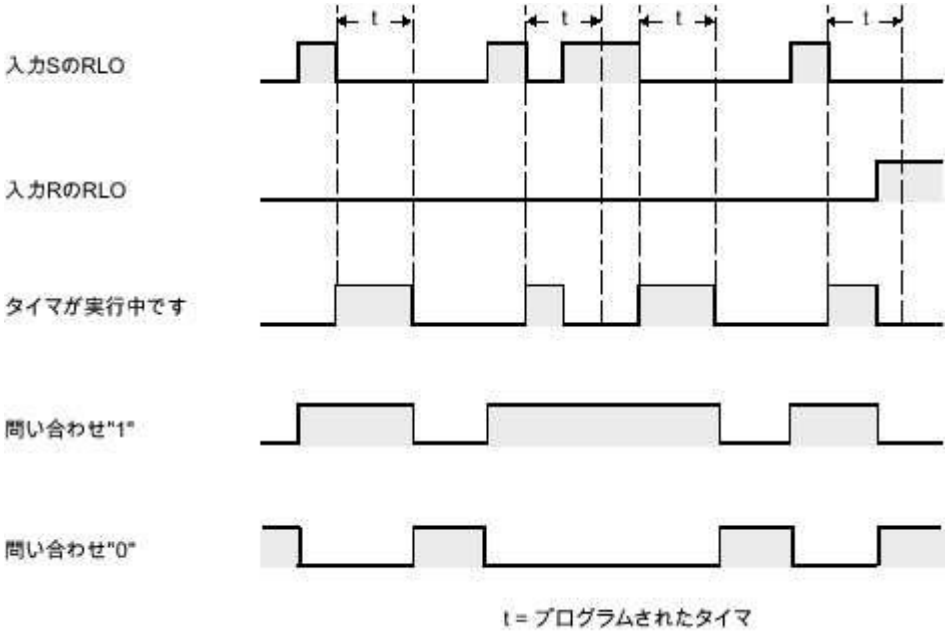
パラメータ	宣言	データタイプ	メモリ領域	説明
<タイマ>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	プリセット時間値
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(BI コード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

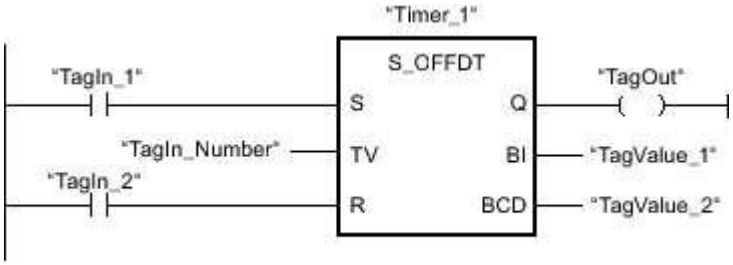
パルスタイミング図

次の図に、「オフディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「1」から「0」に切り替わると起動します。 オペランド「TagIn_Number」の時間値によって、タイマの期限が切れます。 タイマが実行中のときにオペランド「TagIn_1」の信号状態が「0」の場合、オペランド「TagOut」が「1」にセットされます。 タイマの実行中にオペランド「TagIn_1」の信号状態が「0」から「1」に切り替わった場合、タイマがリセットされます。

---(SP): パルスタイマの起動

説明

「パルスタイマの起動」命令は、論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、RLO の信号状態が「1」である限り、指定された時間の間実行されます。 タイマが実行中である限り、「1」に対するタイマステータスをクエリすると、信号状態「1」が返されます。 時間値の期限が切れる前に、RLO で「1」から「0」への切り替えがあった場合、タイマが停止します。 この場合、「1」に対するタイマステータスをクエリすると、信号状態「0」が返されます。

この時間は、内部で時間値とタイムベースから構成されます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。

「パルスタイマの起動」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

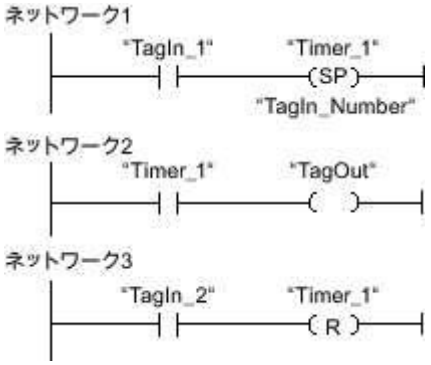
次の表に、「パルスタイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<持続時間>	Input	S5TIME, WORD	I、Q、M、D、L、または定数	タイマの期限が切れる時間。
<タイマ>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 タイマは、オペランド「TagIn_1」の信号状態が「1」である限り、オペランド「TagIn_Number」の時間値で期限切れになります。 オペランド「TagIn_1」の信号状態が、タイマの期限が切れる前に「1」から「0」に切り替わった場合、タイマは停止します。 タイマが実行中である限り、オペランド「TagOut」の信号状態は「1」です。 オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると、タイマがリセットされ、それによってタイマが停止し、現在の時間値が「0」にセットされます。

---(SE): 拡張パルスタマの起動

説明

「拡張パルスタイマの起動」命令は、論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 RLO が信号状態「0」に切り替わった場合でも、タイマは指定された時間によって期限が切れます。 タイマが実行中である限り、「1」に対するタイマステータスをクエリすると、信号状態「1」が返されます。 タイマの実行中に RLO が「0」から「1」に切り替わった場合、タイマはプログラミングされた時間によって再開されます。「1」に対するタイマステータスをクエリすると、タイマの期限が切れるときに信号状態「0」が返されます。

この時間は、内部で時間値とタイムベースから構成されます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。

「拡張パルスタイマの起動」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

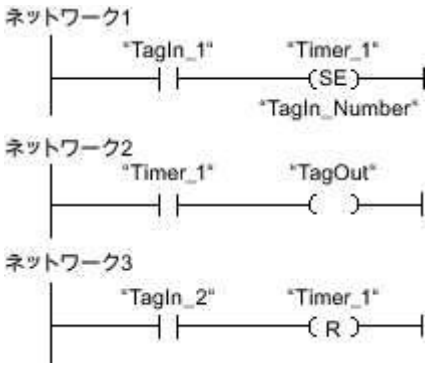
次の表に、「拡張パルスタイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<持続時間>	Input	S5TIME, WORD	I、Q、M、D、L、または定数	タイマの期限が切れる時間。
<タイマ>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 タイマは、オペランド「TagIn_Number」の時間値で期限が切れます。RLO の立ち下がりエッジに影響されません。 タイマが実行中である限り、オペランド「TagOut」の信号状態は「1」です。 オペランド「TagIn_1」の信号状態が、タイマの期限が切れる前に「0」から「1」に切り替わった場合、タイマは再開します。

---(SD): オンディレイタイマの起動

説明

「オンディレータイマの起動」命令は、論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、RLO が「1」である限り、指定された時間の間実行されます。 タイマの期限が切れ、RLO の信号状態がまだ「1」である場合、タイマステータス「1」を照会すると、信号状態「1」が返されます。 タイマの実行中に RLO が「1」から「0」に切り替わった場合、タイマが停止します。 この場合、「1」に対するタイマステータスをクエリすると、信号状態「0」が返されます。

この時間は、内部で時間値とタイムベースから構成されます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。

「オンディレータイマの起動」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

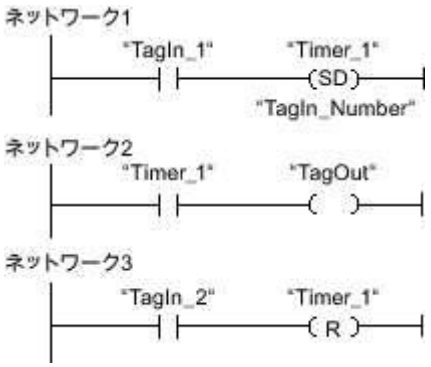
次の表に、「オンディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<持続時間>	Input	S5TIME, WORD	I、Q、M、D、L、または定数	タイマの期限が切れる時間。
<タイマ>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 オペランド「TagIn_Number」の時間値によって、タイマの期限が切れます。 タイマの期限が切れて RLO の信号状態が「1」の場合、オペランド「TagOut」が「1」にセットされます。 オペランド「TagIn_1」の信号状態が、タイマの期限が切れる前に「1」から「0」に切り替わった場合、タイマは停止します。 オペランド「TagIn_2」の信号状態が「1」の場合、タイマ「Timer_1」がリセットされ、それによってタイマが停止し、現在の時間値が「0」にセットされます。

---(SS) 拡張オンディレイタイマの起動

説明

「保持型オンディレータイマの起動」命令は、論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 RLO が信号状態「0」に切り替わった場合でも、タイマは指定された時間によって期限が切れます。 「1」に対するタイマステータスをクエリすると、タイマの期限が切れるときに信号状態「1」が返されます。 タイマの期限が切れた後、タイマは明示的にリセットした場合にのみ再起動できます。

この時間は、内部で時間値とタイムベースから構成されます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。

「保持型オンディレータイマの起動」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

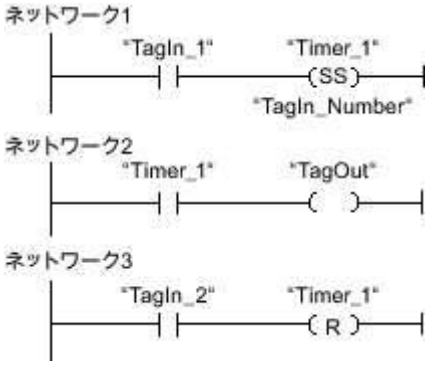
次の表に、「保持型オンディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<持続時間>	Input	S5TIME, WORD	I、Q、M、D、L、または定数	タイマの期限が切れる時間。
<タイマ>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると起動します。 オペランド「TagIn_Number」の時間値によって、タイマの期限が切れます。 タイマの期限が切れると、オペランド「TagOut」が「1」にセットされます。 タイマの実行中にオペランド「TagIn_1」の信号状態が「0」から「1」に切り替わった場合、タイマは再開します。 オペランド「TagIn_2」の信号状態が「1」の場合、タイマ「Timer_1」がリセットされ、それによってタイマが停止し、現在の時間値が「0」にセットされます。

---(SF): オフディレイタイマの起動

説明

「オフディレータイマの起動」命令は、論理演算(RLO)の結果で「1」から「0」への切り替え(信号立ち下がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、指定された時間によって期限が切れます。 タイマが実行中である限り、「1」に対するタイマステータスをクエリすると、信号状態「1」が返されます。 タイマの実行中に RLO が「0」から「1」に切り替わった場合、タイマが停止します。 RLO が「1」から「0」に切り替わると、タイマは必ず再起動されます。

この時間は、内部で時間値とタイムベースから構成されます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値が変更される時間を示します。

「オフディレータイマの起動」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

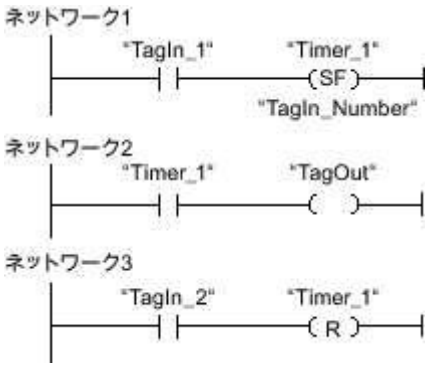
次の表に、「オフディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<持続時間>	Input	S5TIME, WORD	I、Q、M、D、L、または定数	タイマの期限が切れる時間。
<タイマ>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



タイマ「Timer_1」は、オペランド「TagIn_1」の信号状態が「1」から「0」に切り替わると起動します。 オペランド「TagIn_Number」の時間値によって、タイマの期限が切れます。 タイマが実行中である限り、オペランド「TagOut」が「1」にセットされます。 タイマの実行中にオペランド「TagIn_1」の信号状態が「1」から「0」に切り替わった場合、タイマは再開します。 オペランド「TagIn_2」の信号状態が「1」の場合、タイマ「Timer_1」がリセットされ、それによってタイマが停止し、現在の時間値が「0」にセットされます。

カウンタ演算

この章には下記に関する情報が記載されています：

- CTU: カウントアップ (S7-1200, S7-1500)
- CTD: カウントダウン (S7-1200, S7-1500)
- CTUD: カウントアップ/カウントダウン (S7-1200, S7-1500)
- レガシー (S7-1500)

CTU: カウントアップ

説明

出力 CV の値のインクリメントには、「カウントアップ」命令を使用できます。 CU 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、この命令が実行され、CV 出力のカウンタ現在値が 1 ずつインクリメントされます。 立ち上がりエッジを検出するたびに、CV 出力において指定されたデータタイプの上限値に達するまで、カウンタ値がインクリメントされます。 上限値に達すると、CU 入力のシグナル状態はこの命令に影響を与えなくなります。

Q 出力でカウンタステータスを照会することができます。 Q 出力のシグナル状態は、PV パラメータによって決まります。 カウンタ現在値が PV パラメータ値以上の場合、Q 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、Q 出力のシグナル状態は「0」になります。

入力 R のシグナル状態が「1」に切り替わると、CV 出力の値がゼロにリセットされます。R 入力のシグナル状態が「1」である限り、CU 入力のシグナル状態は命令に影響しません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTU_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyIEC_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。 このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。 インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントアップ」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

次の表に、「カウントアップ」命令のパラメータを示します。

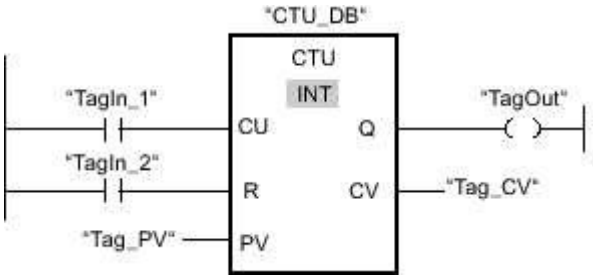
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
CU	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウント入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	リセット入力
PV	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	出力 Q がセットされる値。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	I、Q、M、D、L、P	カウンタ現在値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「0」から「1」に切り替わると、「カウントアップ」命令が実行され、「Tag_CV」カウンタ現在値が 1 インクリメントされます。 それ以降、信号立ち上がり工ッジが検出されるたびに、データタイプの上限値(INT = 32767)に達するまで、カウンタ値がインクリメントされます。

PV パラメータの値が「TagOut」出力を確定する限度として採用されます。 カウンタ現在値がオペランド「Tag_PV」の値以上の場合、「TagOut」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut」出力のシグナル状態は「0」になります。

CTD: カウントダウン

説明

「カウントダウン」命令を使用して、出力 CV の値をデクリメントします。 CD 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、この命令が実行され、CV 出力のカウンタ現在値が 1 ずつデクリメントされます。 指定されたデータタイプの下限値に達するまで、信号の立ち上がりエッジが検出されるたびにカウンタがデクリメントされます。 下限値に達すると、CD 入力のシグナル状態はこの命令に影響を与えなくなります。

Q 出力でカウンタステータスを照会することができます。 カウンタ現在値がゼロ以下の場合、Q 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、Q 出力のシグナル状態は「0」になります。

LD 入力のシグナル状態が「1」に切り替わると、CV 出力の値が PV パラメータの値にセットされます。LD 入力のシグナル状態が「1」である限り、CD 入力のシグナル状態は命令に影響しません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyIEC_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントダウン」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

次の表に、「カウントダウン」命令のパラメータを示します。

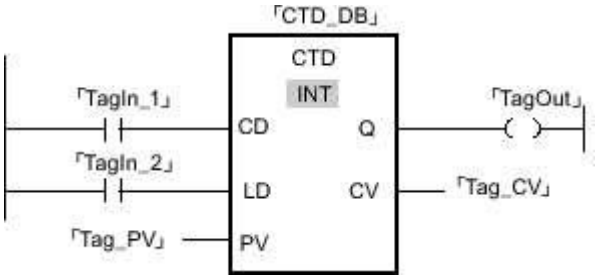
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
CD	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウント入力
LD	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	CV 出力が LD = 1 にセットされる値。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	I、Q、M、D、L、P	カウンタ現在値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CV」出力の値が 1 デクリメントされます。 それ以降、信号立ち上がりエッジが検出されるたびに、指定したデータタイプの下限値(INT = -32768)に達するまで、カウンタ値がデクリメントされます。

カウンタ現在値がゼロ以下である限り、「TagOut」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut」出力のシグナル状態は「0」になります。

CTUD: カウントアップ/カウントダウン

説明

「カウントアップ/カウントダウン」命令を使用して、CV 出力のカウンタ値のインクリメント、およびデクリメントが可能です。 CU 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、カウンタ現在値が 1 インクリメントされ、CV 出力に格納されます。 CD 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、CV 出力のカウンタ値が 1 デクリメントされます。 1 つのプログラムサイクル内で CU および CD 入力に立ち上がりエッジが発生した場合、CV 出力のカウンタ現在値は変わりません。

カウンタ値は、CV 出力で指定したデータタイプの上限値に達するまでインクリメントすることができます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。カウンタ値は、指定されたデータタイプの下限値に達するとデクリメントされなくなります。

LD 入力のシグナル状態が「1」に切り替わると、CV 出力のカウンタ値が PV パラメータの値にセットされます。LD 入力のシグナル状態が「1」である限り、CU および CD 入力のシグナル状態は命令に影響しません。

R 入力のシグナル状態が「1」に切り替わると、カウンタ値がゼロに設定されます。R 入力のシグナル状態が「1」である限り、CU、CD、および LD 入力のシグナル状態が切り替わっても「カウントアップ/カウントダウン」命令には影響を与えません。

QU 出力でアップカウンタのステータスを照会することができます。 カウンタ現在値が PV パラメータ値以上の場合、QU 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、QU 出力のシグナル状態は「0」になります。

QD 出力でダウンカウンタのステータスを照会することができます。 カウンタ現在値がゼロ以下の場合、QD 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、QD 出力のシグナル状態は「0」になります。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ/カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTUD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyIEC_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントアップ/カウントダウン」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

次の表に、「カウントアップ/カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
CU	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウントダウン入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	リセット入力

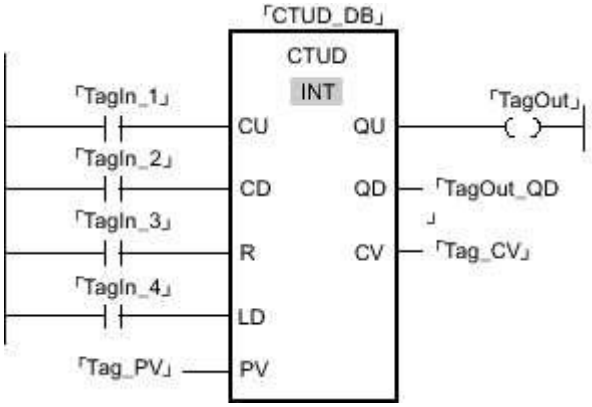
LD	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	出力 QU がセットされる値。 /CV 出力が LD = 1 にセットされる値。
QU	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	アップカウンタのステータス
QD	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウントダウンのステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	I、Q、M、D、L、P	カウンタ現在値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」または「TagIn_2」入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、「カウントアップ/カウントダウン」命令が実行されます。 「TagIn_1」入力で立ち上がりエッジが発生する場合、カウンタ現在値は 1 ずつインクリメントされ、「Tag_CV」出力に保存されます。 「TagIn_2」入力で立ち上がりエッジが発生する場合、カウンタ値は 1 ずつデクリメントされ、「Tag_CV」出力に保存されます。 CU 入力で立ち上がりエッジが発生した場合、カウンタ値が上限値 32767 に達するまでインクリメントされます。入力 CD に立ち上がりエッジがある場合、カウンタ値が INT の下限値-32768 に達するまでデクリメントされます。

カウンタ現在値が「Tag_PV」入力の値以上である限り、「TagOut」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut」出力のシグナル状態は「0」になります。

カウンタ現在値がゼロ以下である限り、「TagOut_QD」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut_QD」出力のシグナル状態は「0」になります。

レガシー

この章には下記に関する情報が記載されています：

- S_CU: パラメータおよびカウントアップの割り当て (S7-1500)
- S_CD: パラメータおよびカウントダウンの割り当て (S7-1500)
- S_CUD: パラメータおよびカウントアップ/カウントダウンの割り当て (S7-1500)
- ---(SC): カウンタ値の設定 (S7-1500)
- ---(CU):カウントアップ (S7-1500)
- ---(CD):カウントダウン (S7-1500)

S_CU: パラメータおよびカウントアップの割り当て

説明

「パラメータおよびカウントアップの割り当て」命令を使用して、カウンタ値のインクリメントが可能です。 CU 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 インクリメントされます。 現在のカウンタ値は、出力 CV に 16 進数値で出力され、出力 CV_BCD で BCD コード化されます。 カウンタ値は、制限値の「999」に達するまでインクリメントできます。 制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。

入力 S の信号状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタが設定され、入力 CU の RLO が「1」の場合、信号エッジで切り替えが検出されていない場合でも、カウンタはその次のスキャンサイクルで適宜カウントします。

R 入力の信号状態が「1」に切り替わると、カウンタ値がゼロに設定されます。 R 入力の信号状態が「1」である限り、CU および S 入力の信号状態の処理はこのカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、出力 Q の信号状態は「1」となります。 カウンタ値がゼロに等しい場合、出力 Q の信号状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「パラメータおよびカウントアップの割り当て」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

パラメータ

次の表に、「パラメータおよびカウントアップの割り当て」命令のパラメータを示します。

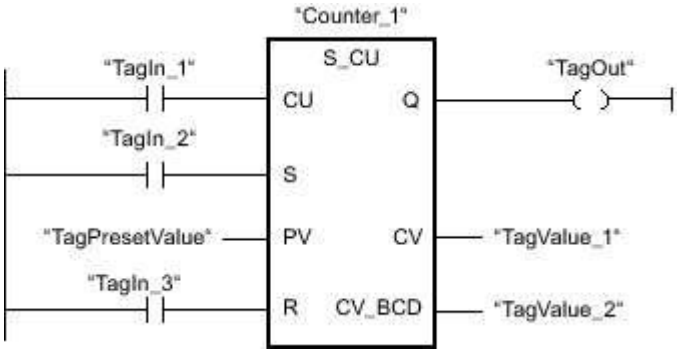
パラメータ	宣言	データタイプ	メモリ領域	説明
<Counter>	InOut/Input	COUNTER	C	命令のカウンタ カウンタの数は CPU によって異なります。
CU	Input	BOOL	I、Q、M、D、L	カウントアップ入力
S	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウンタのプリセット用の入力
PV	Input	WORD	I、Q、M、D、L、または定数	カウンタ値のプリセット(C#0 ~ C#999)
R	Input	BOOL	I、Q、M、D、L、T、C、または定数	リセット入力
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値 (16 進数)

CV_BCD	Output	WORD, S5TIME, DATE	I、Q、M、D、L	現在のカウンタ値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn_1」入力の信号状態が「0」から「1」に切り替わり(立ち上がりエッジ)、現在のカウンタ値が「999」未満の場合、カウンタ値が 1 インクリメントされます。 入力「TagIn_2」の信号状態が「0」から「1」に切り替わると、カウンタ値がオペランド「TagPresetValue」の値にセットされます。 オペランド「TagIn_3」の信号状態が「1」の場合、カウンタ値が「0」にリセットされます。

現在のカウンタ値は、オペランド「TagValue_1」に 16 進数値で保存され、オペランド「TagValue_2」で BCD コード化されます。

現在のカウンタ値が「0」と等しくない場合、「TagOut」出力の信号状態は「1」です。

S_CD: パラメータおよびカウントダウンの割り当て

説明

「パラメータおよびカウントダウンの割り当て」命令を使用して、カウンタ値のデクリメントが可能です。 CD 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 デクリメントされます。 現在のカウンタ値は、出力 CV に 16 進数値で出力され、出力 CV_BCD で BCD コード化されます。 カウンタ値は、下限値「0」に達するまでデクリメントされます。 下限値に達すると、立ち上がりエッジでカウンタ値はそれ以上デクリメントされません。

入力 S の信号状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタが設定され、入力 CD の RLO が「1」の場合、信号エッジで切り替えが検出されていない場合でも、カウンタはその次のスキャンサイクルで適宜カウントします。

R 入力の信号状態が「1」に切り替わると、カウンタ値がゼロに設定されます。 R 入力の信号状態が「1」である限り、CD および S 入力の信号状態の処理はこのカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、出力 Q の信号状態は「1」となります。 カウンタ値がゼロに等しい場合、出力 Q の信号状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「パラメータおよびカウントダウンの割り当て」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

パラメータ

次の表に、「パラメータおよびカウントダウンの割り当て」命令のパラメータを示します。

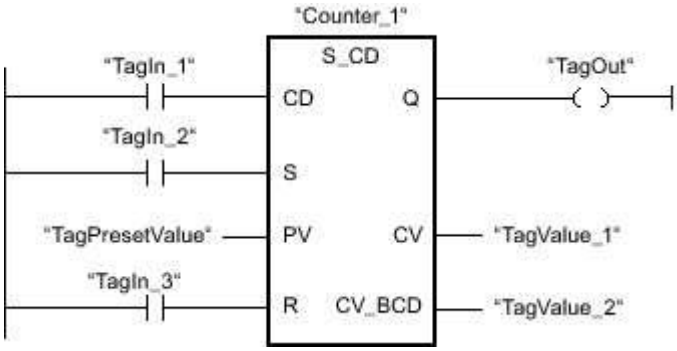
パラメータ	宣言	データタイプ	メモリ領域	説明
<Counter>	InOut/Input	COUNTER	C	命令のカウンタ カウンタの数は CPU によって異なります。
CD	Input	BOOL	I、Q、M、D、L、または定数	カウントダウン入力
S	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウンタのプリセット用の入力
PV	Input	WORD	I、Q、M、D、L、または定数	カウンタ値のプリセット(C#0 ~ C#999)
R	Input	BOOL	I、Q、M、D、L、T、C、または定数	リセット入力
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値 (16 進数)
CV_BCD	Output	WORD, S5TIME, DATE	I、Q、M、D、L	現在のカウンタ値 (BCD フォーマット)

Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn_1」入力の信号状態が「0」から「1」に切り替わり(信号立ち上がりエッジ)、現在のカウンタ値が「0」よりも大きい場合、カウンタ値が 1 デクリメントされます。 入力「TagIn_2」の信号状態が「0」から「1」に切り替わると、カウンタ値がオペランド「TagPresetValue」の値にセットされます。 オペランド「TagIn_3」の信号状態が「1」の場合、カウンタ値が「0」にリセットされます。

現在のカウンタ値は、オペランド「TagValue_1」に 16 進数値で保存され、オペランド「TagValue_2」で BCD コード化されます。

現在のカウンタ値が「0」と等しくない場合、「TagOut」出力の信号状態は「1」です。

S_CUD: パラメータおよびカウントアップ/カウントダウンの割り当て

説明

「パラメータおよびカウントアップ/カウントダウンの割り当て」命令を使用して、カウンタ値のインクリメントおよびデクリメントが可能です。 CU 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 インクリメントされます。 CD 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 デクリメントされます。 現在のカウンタ値は、出力 CV に 16 進数値で出力され、出力 CV_BCD で BCD コード化されます。 1 つのプログラムサイクル内で CU および CD 入力に立ち上がりエッジが発生した場合、カウンタ値は変わりません。

カウンタ値は、上限値の「999」に達するまでインクリメントできます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。 下限値「0」に達すると、それ以降、カウンタ値はデクリメントされません。

入力 S の信号状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタが設定され、入力 CU および CD の RLO が「1」の場合、信号エッジで切り替えが検出されなかった場合でも、カウンタはその次のスキャンサイクルで適宜カウントします。

R 入力の信号状態が「1」に切り替わると、カウンタ値がゼロに設定されます。 R 入力の信号状態が「1」である限り、CU、CD および S 入力の信号状態の処理はこのカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、出力 Q の信号状態は「1」となります。 カウンタ値がゼロに等しい場合、出力 Q の信号状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「パラメータおよびカウントアップ/カウントダウンの割り当て」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワーク内またはネットワークの最後に配置できます。

パラメータ

次の表に、「パラメータおよびカウントアップ/カウントダウンの割り当て」命令のパラメータを示します。

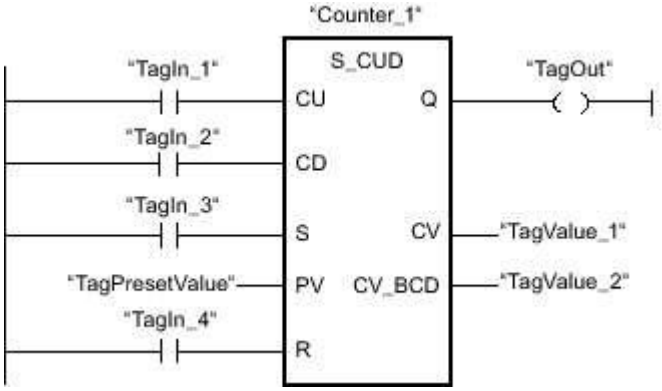
パラメータ	宣言	データタイプ	メモリ領域	説明
<Counter>	InOut/Input	COUNTER	C	命令のカウンタ カウンタの数は CPU によって異なります。
CU	Input	BOOL	I、Q、M、D、L	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウントダウン入力
S	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウンタのプリセット用の入力
PV	Input	WORD	I、Q、M、D、L、または定数	カウンタ値のプリセット(C#0 ~ C#999)

R	Input	BOOL	I、Q、M、D、L、T、C、または定数	リセット入力
BI	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値 (16 進数)
CV_BCD	Output	WORD, S5TIME, DATE	I、Q、M、D、L	現在のカウンタ値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn_1"」または「TagIn_2」入力の信号状態が「0」から「1」に切り替わった場合(立ち上がり工ッジ)、「パラメータおよびカウントアップ/カウントダウンの割り当て」命令が実行されます。 「TagIn_1」入力で立ち上がり工ッジが発生し、現在のカウンタ値が「999」未満の場合、カウンタ値が 1 インクリメントされます。 「TagIn_2」入力で立ち上がり工ッジが発生し、現在のカウンタ値が「0」よりも大きい場合、カウンタ値が 1 デクリメントされます。

入力「TagIn_3」の信号状態が「0」から「1」に切り替わると、カウンタ値がオペランド「TagPreset-Value」の値にセットされます。 オペランド「TagIn_4」の信号状態が「1」の場合、カウンタ値が「0」にリセットされます。

現在のカウンタ値は、オペランド「TagValue_1」に 16 進数値で保存され、オペランド「TagVal-ue_2」で BCD コード化されます。

現在のカウンタ値が「0」と等しくない場合、「TagOut」出力の信号状態は「1」です。

---(SC): カウンタ値の設定

説明

「カウンタ値のセット」命令を使用して、カウンタの値を設定できます。 入力における論理演算の結果(RLO)が「0」から「1」に切り替わると、命令が実行されます。 命令が実行されると、カウンタは指定されたカウンタ値に設定されます。

「カウンタ値のセット」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

パラメータ

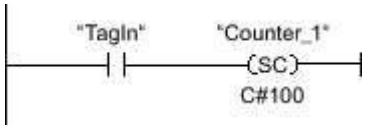
次の表に、「カウンタ値のセット」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<カウント値>	Input	WORD	I、Q、M、D、L、または定数	カウンタがプリセットされる BCD フォーマットの値。 (C#0 ~ C#999)
<Counter>	InOut/Input	COUNTER	C	プリセットされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn」オペランドの信号状態が「0」から「1」に切り替わると、カウンタ「Counter_1」が値「100」で開始されます。

---(CU):カウントアップ

説明

「カウントアップ」命令を使用する、論理演算の結果(RLO)で信号立ち上がりエッジが検出された場合に、指定されたカウンタを 1 つずつインクリメントできます。 カウンタ値は、制限値の「999」に達するまでインクリメントできます。制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。

「カウントアップ」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

パラメータ

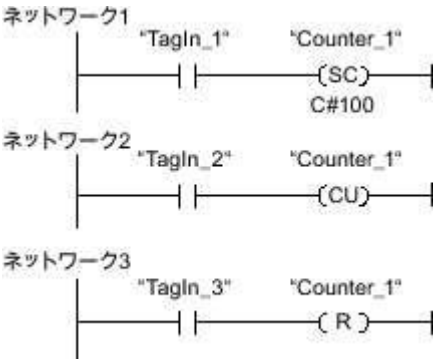
次の表に、「カウントアップ」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<カウンタ>	InOut/Input	COUNTER	C	値がインクリメントされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると(立ち上がりエッジ)、カウンタ「Counter_1」が値「100」でプリセットされます。

オペランド「TagIn_2」の信号状態が「0」から「1」に切り替わると、カウンタ「Counter_1」の値は、1 つずつインクリメントされます。

オペランド「TagIn_3」の信号状態が「1」の場合、カウンタ「Counter_1」の値が「0」にリセットされます。

---(CD):カウントダウン

説明

「カウントダウン」命令を使用する、論理演算の結果(RLO)で信号立ち上がりエッジが検出された場合に、指定されたカウンタを 1 つずつインクリメントできます。 カウンタ値は、制限値の「0」に達するまでデクリメントできます。 制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値は変更されなくなります。

「カウントダウン」命令は、エッジ評価のために事前の論理演算が必要であり、ネットワークの右側にのみ配置できます。

パラメータ

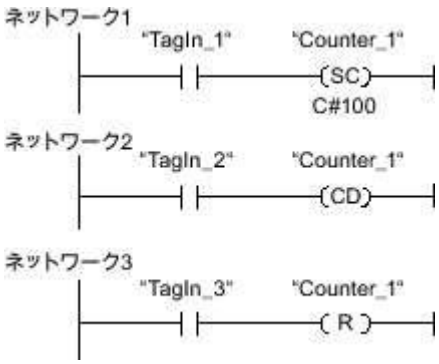
次の表に、「カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<カウンタ>	InOut/Input	COUNTER	C	値がデクリメントされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると(立ち上がりエッジ)、カウンタ「Counter_1」が値「100」でプリセットされます。

オペランド「TagIn_2」の信号状態が「0」から「1」に切り替わると、カウンタ「Counter_1」の値は、1 つずつデクリメントされます。

オペランド「TagIn_3」の信号状態が「1」の場合、カウンタ「Counter_1」の値が「0」にリセットされます。

比較演算

この章には下記に関する情報が記載されています：

CMP ==: 等価

説明

「等価」命令を使用して、第 1 の比較値(<Operand1>)が第 2 の比較値(<Operand2>)と等しいかどうかを確認します。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。 命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダで、最初の比較値(<Operand1>)を指定します。 命令よりも下のオペランドプレースホルダの 2 番目の比較値(<Operand2>)を指定します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。 比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

「等価」命令を使用して、文字列の個々の文字を比較することもできます。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

IEC チェックが有効になっている場合、比較するオペランドが同じデータタイプであることが必要です。IEC チェックが有効になっていない場合、オペランドの幅(長さ)が同じであることが必要です。浮動小数点数を比較する場合、IEC チェックの設定とは関係なく、比較対象のオペランドのデータタイプが同じであることが必要です。

注記

浮動小数点数の比較

データタイプ REAL または LREAL を比較する場合は、命令「CMP ==: 等しい」の代わりに、命令「IN_RANGE: 範囲内の値」を使用します。

注記

データタイプ PORT の比較

「等しい」命令を使用して「PORT」データタイプのオペランドを比較するには、命令ボックスのドロップダウンリストで WORD データタイプを選択する必要があります。

パラメータ

次の表に、「等しい」命令のパラメータをリストします。

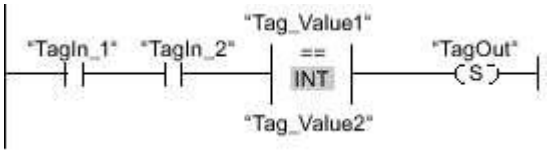
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Operand1>	Input	ビット列、整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
<Operand2>	Input	ビット列、整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- 「Tag_Value1」 = 「Tag_Value2」の場合に、比較命令の条件が満たされていること。

CMP <>: 不等価

説明

「不等価」命令を使用して、第 1 の比較値(<Operand1>)が第 2 の比較値(<Operand2>)と等しくないかどうかを確認します。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。 命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダで、最初の比較値(<Operand1>)を指定します。 命令よりも下のオペランドプレースホルダの 2 番目の比較値(<Operand2>)を指定します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。 比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

「不等価」命令を使用して、文字列の個々の文字を比較することもできます。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

IEC チェックが有効になっている場合、比較するオペランドが同じデータタイプであることが必要です。IEC チェックが有効になっていない場合、オペランドの幅(長さ)が同じであることが必要です。浮動小数点数を比較する場合、IEC チェックの設定に関わらず比較するオペランドは同じデータタイプであることが必要です。

注記

データタイプ PORT の比較

「次の値に等しくない」命令を使用して「PORT」データタイプのオペランドを比較するには、命令ボックスのドロップダウンリストで WORD データタイプを選択する必要があります。

パラメータ

次の表に、「次の値に等しくない」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Operand1>	Input	ビット列、整数、浮動小数点数、文字列、	ビット列、整数、浮動小数点数、文字列、	I、Q、M、D、L、P、または定数	最初の比較値

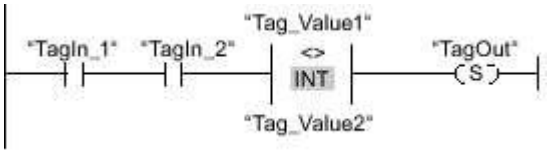
		TIME、DATE、TOD、DTL	TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT		
<Operand2>	Input	ビット列、整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- 「Tag_Value1」 <> 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP >=: 以上

説明

「以上」命令を使用して、第 1 の比較値(<Operand1>)が第 2 の比較値(<Operand2>)以上かどうかを確認します。 比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダで、最初の比較値(<Operand1>)を指定します。 命令よりも下のオペランドプレースホルダの 2 番目の比較値(<Operand2>)を指定します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。 長い方の文字列の左の部分が、短い方の文字列と同一の場合、長い方の文字列が大きいと見なされます。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

「以上」命令を使用して、文字列の個々の文字を比較することもできます。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

時間値を比較する場合、<Operand1>の時間ポイントが<Operand2>の時間ポイントよりも大きい(より最新)か等しい場合、命令の RLO は「1」です。

パラメータ

次の表に、「次の値以上」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Operand1>	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値

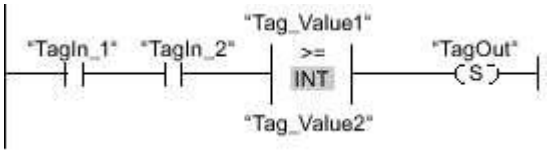
<Operand2>	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- 「Tag_Value1」 >= 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP <=: 以下

説明

「以下」命令を使用して、第 1 の比較値(<Operand1>)が第 2 の比較値(<Operand2>)以下かどうかを確認します。比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダで、最初の比較値(<Operand1>)を指定します。 命令よりも下のオペランドプレースホルダの 2 番目の比較値(<Operand2>)を指定します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。 長い方の文字列の左の部分が、短い方の文字列と同一の場合、短い方の文字列が小さいと見なされます。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

「以下」命令を使用して、文字列の個々の文字を比較することもできます。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

時間値を比較する場合、<Operand1>の時間ポイントが<Operand2>の時間ポイントよりも小さい(より前)か等しい場合、命令の RLO は「1」です。

パラメータ

次の表に、「次の値以下」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Operand1>	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値

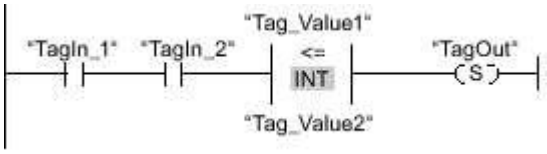
<Operand2>	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- 「Tag_Value1」 <= 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP >: 超過

説明

「超過」命令を使用して、第 1 の比較値(<Operand1>)が第 2 の比較値(<Operand2>)よりも大きいかどうかを確認します。比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダで、最初の比較値(<Operand1>)を指定します。 命令よりも下のオペランドプレースホルダの 2 番目の比較値(<Operand2>)を指定します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。長い方の文字列の左の部分が、短い方の文字列と同一の場合、長い方の文字列が大きいと見なされます。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

「超過」命令を使用して、文字列の個々の文字を比較することもできます。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

時間値を比較する場合、<Operand1>の時間ポイントが<Operand2>の時間ポイントよりも大きければ(より最新)、命令の RLO は「1」です。

パラメータ

次の表に、「次の値より大きい」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Operand1>	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
<Operand2>	Input	整数、浮動小数点数、文字列、	整数、浮動小数点数、文字列、TIME、LTIME、	I、Q、M、D、L、P、または定数	第 2 の比較値

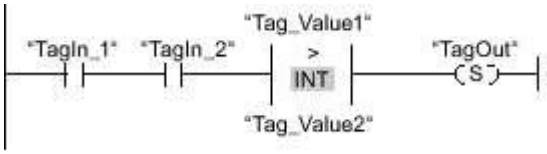
		TIME、DATE、TOD、DTL	DATE、TOD、LTOD、DTL、DT、LDT		

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- 「Tag_Value1」 > 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP <: 未満

説明

「未満」命令を使用して、第 1 の比較値(<Operand1>)が第 2 の比較値(<Operand2>)未満であるかどうかを確認します。比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダで、最初の比較値(<Operand1>)を指定します。 命令よりも下のオペランドプレースホルダの 2 番目の比較値(<Operand2>)を指定します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。長い方の文字列の左の部分が、短い方の文字列と同一の場合、短い方の文字列が小さいと見なされます。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

「未満」命令を使用して、文字列の個々の文字を比較することもできます。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

時間値を比較する場合、<Operand1>の時間ポイントが<Operand2>の時間ポイントよりも小さければ(より前)、命令の RLO は「1」です。

パラメータ

次の表に、「次の値よりも小さい」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Operand1>	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
<Operand2>	Input	整数、浮動小数点数、文字列、	整数、浮動小数点数、文字列、TIME、LTIME、	I、Q、M、D、L、P、または定数	第 2 の比較値

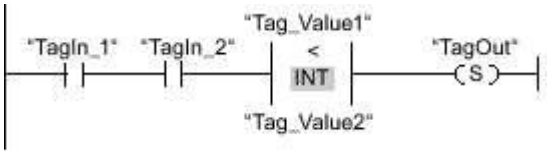
		TIME、DATE、TOD、DTL	DATE、TOD、LTOD、DTL、DT、LDT		

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- 「Tag_Value1」 < 「Tag_Value2」の場合、比較命令の条件が満たされます。

IN_RANGE: 範囲内の値

説明

「範囲内の値」命令を使用して、VAL 入力の値が特定の値の範囲内であるかどうかを確認できます。

MIN および MAX 入力で値の範囲の限度を指定します。「範囲内の値」命令は、VAL 入力の値と、MIN および MAX 入力の値を比較し、結果をボックス出力に送信します。 VAL 入力の値が、比較 MIN <= VAL または VAL <= MAX を満たしている場合、ボックス出力の信号状態は「1」です。 比較の条件が満たされない場合、ボックス出力の信号状態は「0」です。

ボックス入力の信号状態が「0」の場合、「範囲内の値」命令は実行されません。

比較ファンクションは、比較対象の値が同じデータタイプでボックス入力が相互接続されている場合にのみ実行できます。

パラメータ

次の表に、「範囲内の値」命令のパラメータを示します。

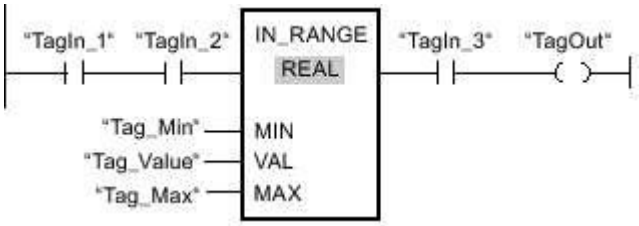
パラメータ	宣言	データタイプ	メモリ領域	説明
Box input	Input	BOOL	I、Q、M、D、L	直前の論理演算の結果
MIN	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の下限値
VAL	Input	整数、浮動小数点数	I、Q、M、D、L または定数	比較値
MAX	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の上限値
Box output	Output	BOOL	I、Q、M、D、L	比較の結果

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。

- オペランド「Tag_Value」の値が、オペランド「Tag_Min」および「Tag_Max」(MIN <= VAL または VAL <= MAX)の現在値で指定された値の範囲内になっています。
- オペランド「TagIn_3」の信号状態が「1」であること。

OUT_RANGE: 範囲外の値

説明

「範囲外の値」命令を使用して、VAL 入力の値が特定の値の範囲外であるかどうかを確認できます。

MIN および MAX 入力で値の範囲の限度を指定します。「範囲外の値」命令は、VAL 入力の値と、MIN および MAX 入力の値を比較し、結果をボックス出力に送信します。 VAL 入力の値が、比較 MIN > VAL または VAL > MAX を満たしている場合、ボックス出力の信号状態は「1」です。 REAL のデータタイプを持つ指定したオペランドが無効な値を示す場合も、ボックス出力の信号状態は「1」です。

入力 VAL の値が MIN > VAL または VAL > MAX の条件を満たさない場合、ボックス出力が信号状態「0」を返します。

ボックス入力の信号状態が「0」の場合、「範囲外の値」命令は実行されません。

比較ファンクションは、比較対象の値が同じデータタイプでボックス入力が相互接続されている場合にのみ実行できます。

パラメータ

次の表に、「範囲外の値」命令のパラメータを示します。

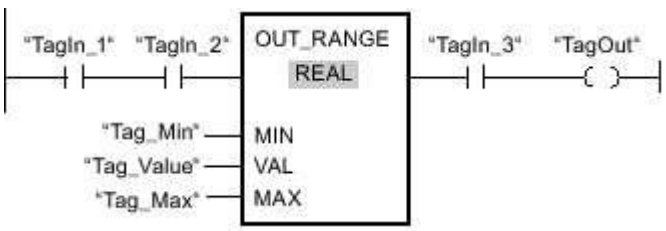
パラメータ	宣言	データタイプ	メモリ領域	説明
Box input	Input	BOOL	I、Q、M、D、L	直前の論理演算の結果
MIN	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の下限値
VAL	Input	整数、浮動小数点数	I、Q、M、D、L または定数	比較値
MAX	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の上限値
Box output	Output	BOOL	I、Q、M、D、L	比較の結果

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。
- オペランド「Tag_Value」の値が、オペランド「Tag_Min」および「Tag_Max」(MIN > VAL または VAL > MAX)の現在値で指定された値の範囲外にあります。
- オペランド"TagIn_3"の信号状態が「1」であること。

----I OK I----: 有効性のチェック

説明

「有効性チェック」命令を使って、オペランド(<operand>)の値が有効な浮動小数点数であるかチェックできます。 クエリは、命令の入力の信号状態が「1」であると、各プログラムサイクルで開始されます。

オペランドの値にクエリの実行時、有効な浮動小数点数である場合、命令の出力の信号状態は「1」になり、命令の入力の信号状態は「1」になります。 それ以外のすべての場合、「有効性チェック」命令の出力の信号状態は「0」です。

「有効性チェック」命令は、EN 機能と一緒に使用することができます。 命令ボックスを EN イネーブル入力に接続すると、このイネーブル入力は有効性クエリの結果が正である場合にのみセットされます。 このファンクションを使用して、指定したタグの値が有効な浮動小数点数である場合のみ、この命令を有効にすることができます。

パラメータ

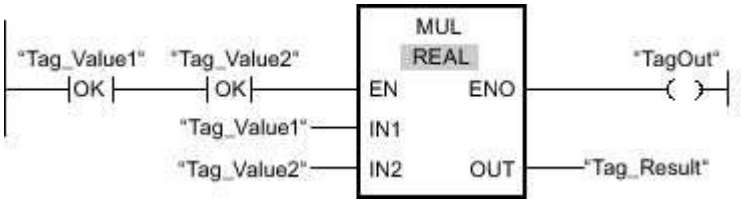
次の表に、「有効性チェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Input	浮動小数点数	I、Q、M、D、L	照会対象の値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド Tag_Value1 および Tag_Value2 の値が有効な浮動小数点数を表す場合、「乗算」 (MUL)命令が有効になり、ENO 出力がセットされます。 「乗算」(MUL)命令の実行中、オペランド「Tag_Value1」の値がオペランド「Tag_Value2」の値で乗算されます。次に、乗算の結果がオペランド「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、ENO および "TagOut" 出力が信号状態「1」にセットされます。

----I NOT_OK I----: 無効性のチェック

説明

「無効性チェック」命令を使って、オペランド(<operand>)の値が無効な浮動小数点数であるかチェックできます。 クエリは、命令の入力の信号状態が「1」であると、各プログラムサイクルで開始されます。

オペランドの値にクエリの実行時、無効な浮動小数点数である場合、命令の出力の信号状態は「1」になり、命令の入力の信号状態は「1」になります。 それ以外のすべての場合、「無効性のチェック」命令の出力の信号状態は「0」です。

パラメータ

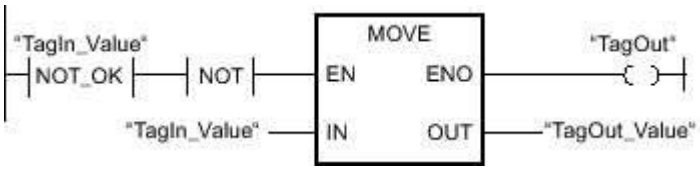
次の表に、「無効性チェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Input	浮動小数点数	I、Q、M、D、L	照会対象の値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_Value」の値が無効な浮動小数点数の場合、「値の移動」(MOVE)命令は実行されません。「TagOut」出力は、信号状態「0」にリセットされます。

VARIANT

この章には下記に関する情報が記載されています：

EQ_Type: EQUAL(等しい)かどうかデータタイプをタグのデータタイプと比較

説明

「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグのデータタイプを照会できます。 ブロックインターフェースで宣言したタグ(<Operand1>)のデータタイプと、タグ(<Operand2>)のデータタイプを「等しい」かどうか比較します。

<Operand1>のデータタイプは、VARIANT データタイプであることが必要です。 <Operand2>は、基本データタイプまたは PLC データタイプです。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダに<Operand1>を指定します。 命令の下のオペランドプレースホルダに<Operand2>を指定します。

パラメータ

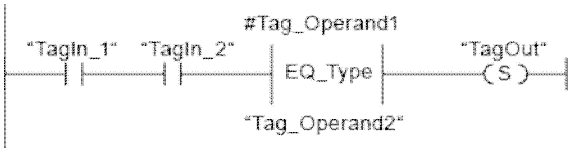
次の表に、「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	VARIANT	I、Q、M、L	最初のオペランド
<Operand2>	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、「#Tag_Operand1」オペランドが、「Tag_Operand2」と等しいこと。

NE_Type: UNEQUAL(等しくない)かどうかデータタイプをタグのデータタイプと比較

説明

「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグが所有しないデータタイプを照会できます。 ブロックインターフェースで宣言したタグ(<Operand1>)のデータタイプと、タグ(<Operand2>)のデータタイプを「等しくない」かどうか比較します。

<Operand1>のデータタイプは、VARIANT データタイプであることが必要です。 <Operand2>は、基本データタイプまたは PLC データタイプです。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダに<Operand1>を指定します。 命令の下のオペランドプレースホルダに<Operand2>を指定します。

パラメータ

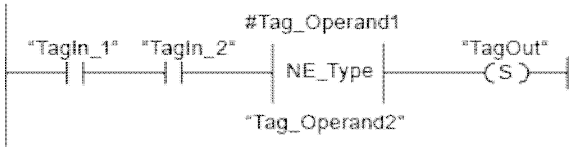
次の表に、「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	VARIANT	I、Q、M、L	最初のオペランド
<Operand2>	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、「#Tag_Operand1」オペランドが、「Tag_Operand2」と等しくないこと。

EQ_ElemType: EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較

説明

「EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグのデータタイプを照会できます。 ブロックインターフェースで宣言したタグ(<Operand1>)のデータタイプと、タグ(<Operand2>)のデータタイプを「等しい」かどうか比較します。

<Operand1>のデータタイプは、VARIANT データタイプであることが必要です。 <Operand2>は、基本データタイプまたは PLC データタイプです。

VARIANT タグ(<Operand1>)のデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダに<Operand1>を指定します。 命令の下のオペランドプレースホルダに<Operand2>を指定します。

パラメータ

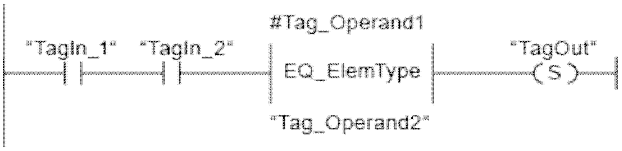
次の表に、「EQUAL (等しい)かどうか配列のデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	VARIANT	I、Q、M、L	最初のオペランド
<Operand2>	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。

- 比較命令の条件が満たされていること。すなわち、「#Tag_Operand1」オペランドが、「Tag_Operand2」と等しいこと。

NE_ElemType: UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較

説明

「UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグが所有しないデータタイプを照会できます。 ブロックインターフェースで宣言したタグ(<Operand1>)のデータタイプと、タグ(<Operand2>)のデータタイプを「等しくない」かどうか比較します。

<Operand1>のデータタイプは、VARIANT データタイプであることが必要です。 <Operand2>は、基本データタイプまたは PLC データタイプです。

VARIANT タグ(<Operand1>)のデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。命令の RLO は、次のようにラング全体の RLO と論理的に組み合わされます。

- AND で連結(比較命令が直列で接続されている場合)。
- OR で連結(比較命令が並列で接続されている場合)。

命令の上のオペランドプレースホルダに<Operand1>を指定します。 命令の下のオペランドプレースホルダに<Operand2>を指定します。

パラメータ

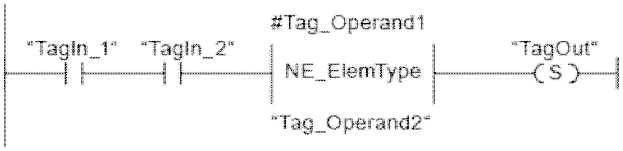
次の表に、「UNEQUAL (等しくない)かどうか配列のデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	VARIANT	I、Q、M、L	最初のオペランド
<Operand2>	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。

- 比較命令の条件が満たされていること。すなわち、「#Tag_Operand1」オペランドが、「Tag_Operand2」と等しくないこと。

IS_NULL: EQUALS NULL ポインタのチェック

説明

「EQUALS NULL ポインタのチェック」命令を使用して、VARIANT が NULL ポインタをポイントし、そのためオブジェクトをポイントしないかどうかを照会することができます。

<Operand>のデータタイプは、VARIANT であることが必要です。

注記

VARIANT タグは、ANY ポインタを指します。

VARIANT タグが ANY ポインタを指す場合、この命令は常に(ANY ポインタが NULL である場合でも)、結果 RLO = 「0」を返します。

パラメータ

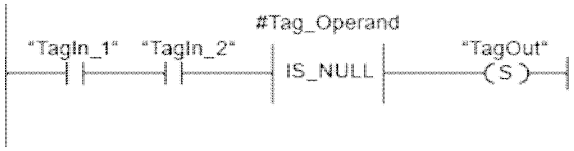
次の表に、「EQUALS NULL ポインタのチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	EQUALS NULL (NULL に等しい)かどうかで比較されるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、「#Tag_Operand」オペランドが、オブジェクトを指していないこと。

NOT_NULL: UNEQUALS NULL ポインタのチェック

説明

「UNEQUALS NULL ポインタのチェック」命令を使用して、が NULL ポインタをポイントせず、VARIANT そのためオブジェクトをポイントするかどうかを照会することができます。

<Operand>のデータタイプは、VARIANT であることが必要です。

注記

VARIANT タグは、ANY ポインタを指します。

VARIANT タグが ANY ポインタを指す場合、この命令は常に(ANY ポインタが NULL である場合でも)、結果 RLO = 「1」を返します。

パラメータ

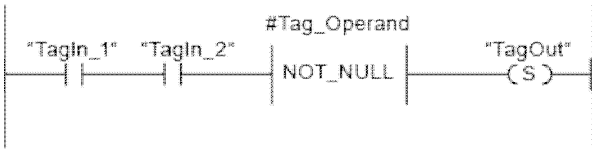
次の表に、「UNEQUALS NULL ポインタのチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	UNEQUALS NULL (NULL に等しくない)かどうかで比較されるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、「#Tag_Operand」オペランドが、オブジェクトを指していること。

IS_ARRAY: 配列のチェック

説明

「配列のチェック」命令を使用して、VARIANT が ARRAY データタイプのタグを指すかどうかを照会できます。

<Operand>のデータタイプは、VARIANT データタイプであることが必要です。

パラメータ

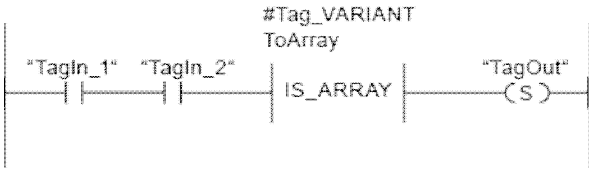
次の表に、「配列のチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	ARRAY に関する照会が行われるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、「#Tag_VARIANTToArray」オペランドのデータタイプが、「ARRAY」であること。

四則演算

この章には下記に関する情報が記載されています：

CALCULATE: 計算

説明

「計算」命令を使用して、選択したデータタイプに応じた数学演算および複雑な論理演算の計算のための式を定義し、実行します。

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。 選択したデータタイプに応じて、一定の命令のファンクションを組み合わせて、複雑な計算を実行できます。計算する式は、命令ボックスの上部にある「計算機」アイコンで開くことができるダイアログで指定します。 式には、入力パラメータの名前と命令の構文を入れることができます。 オペランド名とオペランドのアドレスは指定できません。

初期状態では、この命令ボックスには少なくとも 2 つの入力(IN1 および IN2)が含まれています。 入力数は拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。

入力の値は、指定された式を実行するのに使われます。 定義済み入力のすべてを式に使用する必要はありません。 この命令の実行結果は出力 OUT に転送されます。

注記
式の算術演算の 1 つでエラーが発生すると、結果は出力 OUT に転送されず、イネーブル出力 ENO は信号状態「1」を返します。

ボックス内で使用できない式に入力を使用すると、その入力が自動的に挿入されます。 式に新たに定義される入力は、番号が飛んでいてはいけません。 たとえば、式に IN4 入力を使用できません。ただし、IN3 入力が定義されている場合は除きます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 「計算」命令の結果が、OUT 出力で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。
- 式のいずれかの命令の実行中に、エラーが発生していること。

次の表に、選択したデータタイプに応じて、命令「計算」の式で組み合わせて実行が可能な命令を示します。

データタイプ	命令	構文	例
ビット列	AND: AND 演算	AND	IN1 AND IN2 OR IN3
	OR: OR 論理和演算	OR	
	XOR: EXCLUSIVE OR 排他的論理和演算	XOR	
	INV: 1 の補数の作成	NOT	
	SWAP: スワップ 1)	SWAP	
整数	ADD: 加算	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: 減算	-	
	MUL: 乗算	*	
	DIV: 除算	/	
	MOD: 除算の剰余を返す	MOD	

	INV: 1 の補数の作成	NOT	
	NEG: 2 の補数の作成	-(in1)	
	ABS: 絶対値の形成	ABS( )	
浮動小数点数	ADD: 加算	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: 減算	-	
	MUL: 乗算	*	
	DIV: 除算	/	
	EXPT: 累乗	**	
	ABS: 絶対値の形成	ABS( )	
	SQR: 2 乗値の形成	SQR( )	
	SQRT: 平方根の形成	SQRT( )	
	LN: 自然対数の形成	LN( )	
	EXP: 指数値の形成	EXP( )	
	FRAC: 小数部を返す	FRAC( )	
	SIN: 正弦値の形成	SIN( )	
	COS: 余弦値の形成	COS( )	
	TAN: 正接値の形成	TAN( )	
	ASIN: 逆正弦値の形成	ASIN( )	
	ACOS: 逆余弦値の形成	ACOS( )	
	ATAN: 逆正接値の形成	ATAN( )	
	NEG: 2 の補数の作成	-(in1)	
	TRUNC: 数値の切り捨て	TRUNC( )	
	ROUND: 数値の四捨五入	ROUND( )	
	CEIL: 浮動小数点数から次に大きい整数を生成	CEIL( )	
	FLOOR: 浮動小数点数から次に小さい整数を生成	FLOOR( )	
1)データタイプ BYTE では使用不可。			

パラメータ

次の表に、「計算」命令のパラメータを示します。

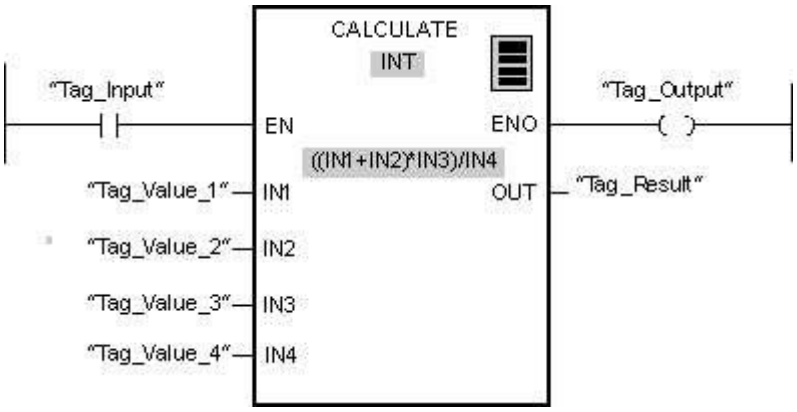
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN1	Input	ビット列、整数、浮動小数点数	I、Q、M、D、L、P、または定数	最初に利用可能な入力
IN2	Input	ビット列、整数、浮動小数点数	I、Q、M、D、L、P、または定数	2 番目に使用可能な入力
INn	Input	ビット列、整数、浮動小数点数	I、Q、M、D、L、P、または定数	追加で挿入された入力

OUT	Output	ビット列、整数、浮動小数点数	I、Q、M、D、L、P	最終結果の転送先の出力。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

「Tag_Input」入力の信号状態が「1」の場合、「計算」命令が実行されます。 オペランド「Tag_Value_1」の値がオペランド「Tag_Value_2」の値に加算されます。 合計は、オペランド「Tag_Value_3」の値で乗算します。 積は、オペランド「Tag_Value_4」の値で除算されます。 商は、命令の OUT 出力のオペランド「Tag_Result」に終了結果として転送されます。 命令が工ラーなく実行されると、イネーブル出力 ENO とオペランド「Tag_Output」が「1」にセットされます。

ADD: 加算

説明

「加算」命令を使って、入力 IN1 の値と入力 IN2 の値を加算し、出力 OUT (OUT := IN1+IN2)で合計を照会できます。

初期状態では、命令ボックスに最低 2 入力(IN1 と IN2)が含まれます。 入力数は拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。 命令を実行する場合、入力パラメータに使用できるすべての値が加算されます。 合計は OUT 出力に保存されます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- イネーブル入力 EN の信号状態が「0」であること。
- 命令の結果が、OUT 出力で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「加算」命令のパラメータを示します。

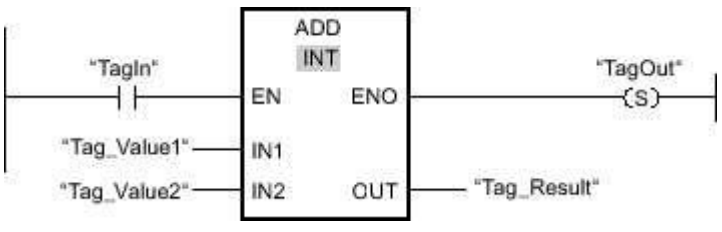
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	加算する最初の数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	加算する 2 番目の数
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	加算対象のオプションの入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	合計

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」の信号状態が「1」の場合、「加算」命令が実行されます。 オペランド「Tag_Value1」の値がオペランド「Tag_Value2」の値に加算されます。 加算の結果がオペランド「Tag_Result」に保存されます。 この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」になり、「TagOut」出力がセットされます。

SUB:減算

説明

「減算」命令を使って、入力 IN2 の値を入力 IN1 から減算し、出力 OUT (OUT := IN1-IN2)での差分を照会します。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 命令の結果が、OUT 出力で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「減算」命令のパラメータを示します。

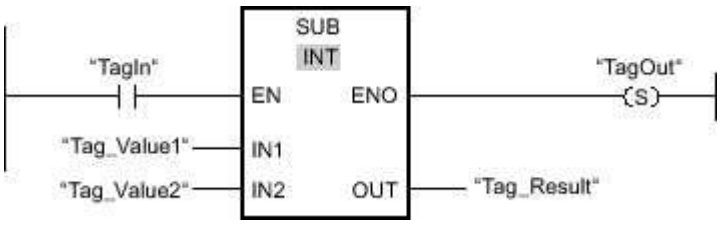
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	被減数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	減算
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	差

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」の信号状態が「1」の場合、「減算」命令が実行されます。 オペランド「Tag_Value2」の値が、オペランド「Tag_Value1」の値から減算されます。 減算の結果がオペランド「Tag_Result」に保存されます。この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」となり、「TagOut」出力がセットされます。

MUL:乗算

説明

「乗算」命令を使って、入力 IN1 の値を入力 IN2 の値で乗算し、出力 OUT (OUT := IN1 * IN2)での積を照会します。

入力の数は、命令ボックスで拡張できます。 追加された入力は、ボックスで昇順に番号付けされます。 命令が実行されると、すべての使用可能な入力パラメータの値が乗算されます。 積は OUT 出力に格納されます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN 入力の信号状態が「0」であること。
- 結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「乗算」命令のパラメータを示します。

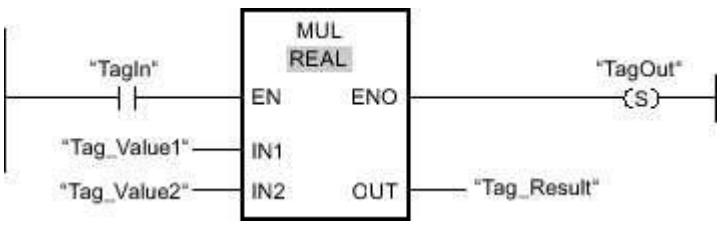
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	乗数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	乗算される数値
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	乗算できるオプションの入力値を以下に示します。
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	製品

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」の信号状態が「1」の場合、「乗算」命令が実行されます。 オペランド「Tag_Value1」の値に、オペランド「Tag_Value2」の値が乗算されます。 乗算の結果がオペランド「Tag_Result」に保存されます。この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」となり、「TagOut」出力がセットされます。

DIV:除算

説明

「除算」命令を使って、入力 IN1 の値を入力 IN2 の値で除算し、出力 OUT (OUT := IN1/IN2)での商を照会します。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 命令の結果が、OUT 出力で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「除算」命令のパラメータを示します。

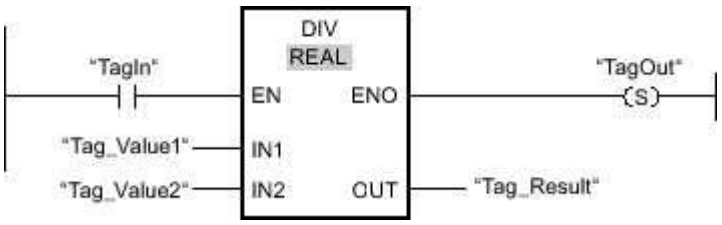
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	被除数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	除数
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	商の値

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」の信号状態が「1」の場合、「除算」命令が実行されます。 オペランド「Tag_Value1」の値が、オペランド「Tag_Value2」の値で除算されます。 除算の結果がオペランド「Tag_Result」に保存されます。この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」となり、「TagOut」出力がセットされます。

MOD:除算の剰余を返す

説明

「除算の剰余を返す」命令を使用して、入力 IN1 の値を入力 IN2 の値で除算し、出力 OUT で除算の余りを照会します。

パラメータ

次の表に、「除算の剰余を返す」命令のパラメータを示します。

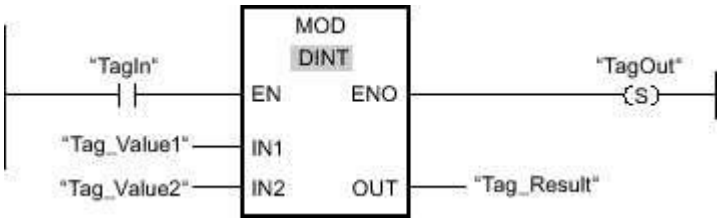
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN1	Input	整数	I、Q、M、D、L、P、または定数	被除数
IN2	Input	整数	I、Q、M、D、L、P、または定数	除数
OUT	Output	整数	I、Q、M、D、L、P	除算の余り

命令ボックスの[<Auto ???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」のシグナル状態が「1」の場合、「除算の剰余を返す」命令が実行されます。 オペランド「Tag_Value1」の値が、オペランド「Tag_Value2」の値で除算されます。除算の余りはオペランド「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

NEG:2 の補数の作成

説明

「2 の補数の作成」命令を使用して、IN 入力の値の符号を変更し、OUT 出力で結果をクエリできます。たとえば、入力 IN に正の値がある場合、この値をマイナスにした値が出力 OUT に出力されます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 命令の結果が、OUT 出力で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「2 つの補数の作成」命令のパラメータを示します。

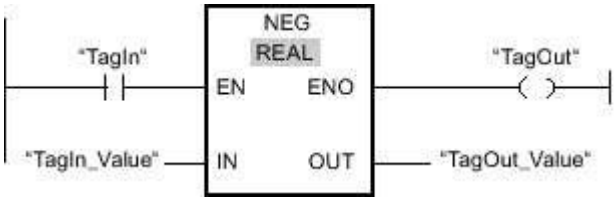
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P または定数	入力値
OUT	Output	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	入力値の 2 の補数

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」の信号状態が「1」の場合、「2 の補数の作成」命令が実行されます。 入力「TagIn_Value」での値の符号は変更され、結果は「TagOut_Value」出力に示されます。この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」となり、「TagOut」出力がセットされます。

INC: インクリメント

説明

「インクリメント」命令を使って、IN/OUT パラメータのオペランドの値を次の大きな値に変更し、結果を照会します。「インクリメント」命令は、EN イネーブル入力の信号状態が「1」の場合にのみ実行されます。 実行中にオーバーフローエラーが発生しなければ、ENO イネーブル出力の信号状態も「1」になります。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「インクリメント」命令のパラメータを示します。

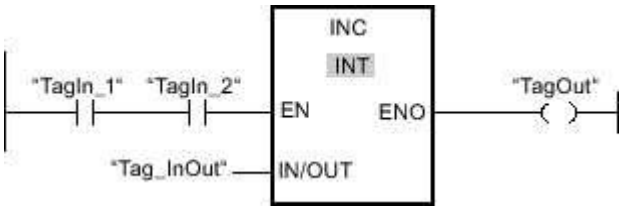
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN/OUT	InOut	整数	I、Q、M、D、L	インクリメントされる値

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、オペランド「Tag_InOut」の値が 1 ずつインクリメントされ、「TagOut」出力がセットされます。

DEC: デクリメント

説明

「デクリメント」命令を使って、IN/OUT パラメータのオペランドの値を次の小さな値に変更し、結果を照会します。「デクリメント」命令は、EN イネーブル入力の信号状態が「1」の場合にのみ実行されます。 処理中に選択したデータタイプの値の範囲を超えなければ、ENO 出力の信号状態も「1」になります。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「デクリメント」命令のパラメータを示します。

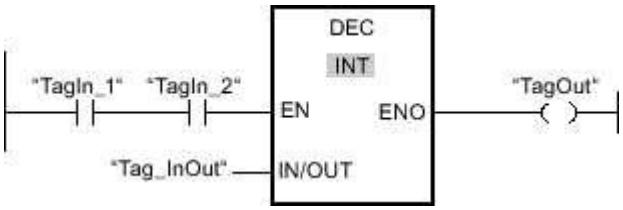
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN/OUT	InOut	整数	I、Q、M、D、L	デクリメントされる値

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、オペランド「Tag_InOut」の値が 1 ずつデクリメントされ、「TagOut」出力がセットされます。

ABS:絶対値の形成

説明

「絶対値の形成」命令を使って、入力 IN で指定した値の絶対値を計算します。 命令の結果は、OUT 出力に送信され、そこから照会できます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「絶対値の形成」命令のパラメータを示します。

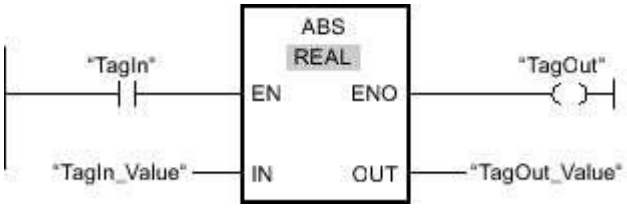
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P または定数	入力値
OUT	Output	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	入力値の絶対値

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

オペランド「TagIn」の信号状態が「1」の場合、「絶対値の形成」命令が実行されます。この命令は、入力「TagIn_Value」の値の絶対値を計算し、結果を出力「TagOut_Value」に送信します。 この命令が工ラーなしで実行された場合、「TagOut」出力がセットされます。

MIN: 最小値の取得

説明

「最小値の取得」命令は、使用可能な入力の値を比較し、最小値を出力 OUT に書き込みます。 入力の数は、追加入力によって、命令ボックスで拡張できます。 入力は、ボックスで昇順に番号付けされます。

命令を実行するには、2 つの入力の最小値と 100 の入力の最大値を指定する必要があります。

次のいずれかの条件が満たされる場合、ENO 許可出力のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「最小値の取得」命令のパラメータを示します。

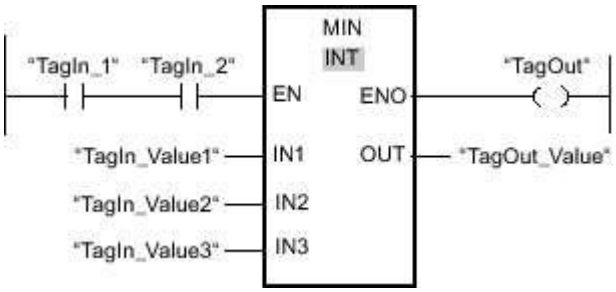
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	2 番目の入力値
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	値が比較される追加挿入された入力
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	結果
IEC チェックが有効でない場合も、等しい長さのビット列または整数を命令のデータタイプとして選択することによって、データタイプ TIME、LTIME、TOD、LTOD、DATE および LDT のタグを使用することができます(TIME => DINT の代わりに、UDINT または DWORD = 32 ビットなど)。				

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、指定されたオペランドの値を比較し、最小値(「TagIn_Value1」)を出力「TagOut_Value」にコピーします。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

MAX: 最大値の取得

説明

「最大値の取得」命令は、使用可能な入力の値を比較し、最大値を出力 OUT に書き込みます。 入力の数は、追加入力によって、命令ボックスで拡張できます。 入力は、ボックスで昇順に番号付けされます。

命令を実行するには、2 つの入力の最小値と 100 の入力の最大値を指定する必要があります。

次のいずれかの条件が満たされる場合、ENO 許可出力のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「最大値の取得」命令のパラメータを示します。

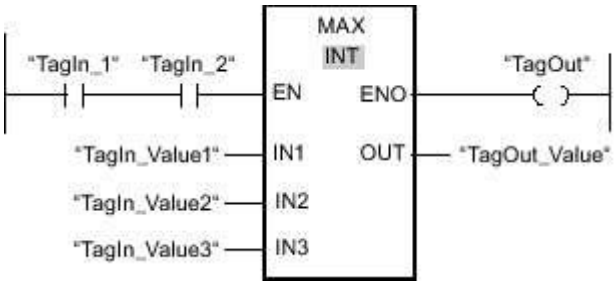
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	2 番目の入力値
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	値が比較される追加挿入された入力
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	結果
IEC チェックが有効でない場合も、等しい長さのビット列または整数を命令のデータタイプとして選択することによって、データタイプ TIME、LTIME、TOD、LTOD、DATE および LDT のタグを使用することができます (TIME => DINT の代わりに、UDINT または DWORD = 32 ビットなど)。				

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、指定されたオペランドの値を比較し、最大値(「TagIn_Value2」)を出力「TagOut_Value」にコピーします。 この命令が工エラーなしで実行された場合、「TagOut」出力がセットされます。

LIMIT: 制限値の設定

説明

「制限値の設定」命令を使用して入力 IN の値を入力 MN および MX の値に制限することができます。 IN 入力の値が条件 MN <= IN MX を満たす場合、この値は OUT 出力にコピーされます。 条件が満たされず入力値 IN が下限値 MN を下回る場合、出力 OUT は MN 入力の値にセットされます。 上限値 MX を超えると、OUT 出力が MX 入力の値にセットされます。

MN 入力の値が MX 入力の値よりも大きい場合、結果は定義されず、イネーブル出力 ENO は「0」となります。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- イネーブル入力 EN の信号状態が「0」であること。
- 指定された複数のタグが同じデータタイプでないこと。
- オペランドの値が無効である。
- MN 入力の値が MX 入力の値よりも大きいこと。

パラメータ

次の表に、「制限値の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	イネーブル出力
MN	Input	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	下限値
IN	Input	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	入力値
MX	Input	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	上限値
OUT	Output	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	結果

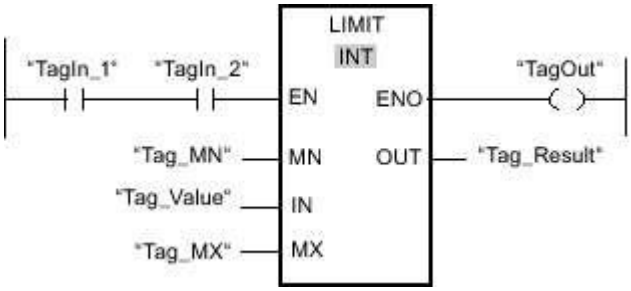
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

オペランド TagIn_1 および TagIn_2 の信号状態が「1」の場合、「制限値の設定」命令が実行されます。 オペランド「Tag_Value」の値が、オペランド「Tag_MN」および「Tag_MX」の値と比較されます。 オペランド「Tag_Value」の値が下限値未満であるため、オペランド「Tag_MN」の値が「Tag_Result」出力にコピーされます。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

SQR: 2 乗値の形成

説明

「2 乗値の形成」命令を使用して、入力 IN の浮動小数点数の値を二乗し、結果を出力 OUT に書き込みます。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「2 乗値の形成」命令のパラメータを示します。

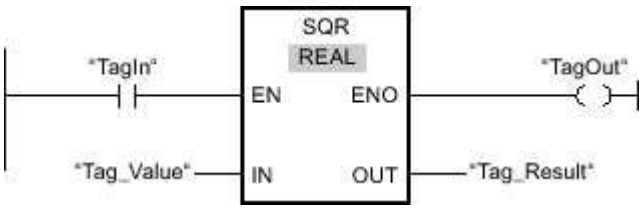
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	入力値の 2 乗値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

オペランド「TagIn」のシグナル状態が「1」の場合、「2 乗値の形成」命令が実行されます。この命令は、オペランド「Tag_Value」の値を 2 乗し、結果を出力「Tag_Result」に送信します。この命令の実行中にエラーが発生しなかった場合、出力「TagOut」が設定されます。

SQRT: 平方根の形成

説明

「平方根の形成」命令を使用して、入力 IN の浮動小数点数の値の平方根を形成し、結果を出力 OUT に書き込みます。この命令は、入力値がゼロよりも大きい場合に正の結果になります。入力値がゼロ未満の場合、出力 OUT が無効な浮動小数点数を返します。入力 IN の値が「0」の場合、結果も「0」になります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。
- 入力 IN の値が負。

パラメータ

次の表に、「平方根の形成」命令のパラメータを示します。

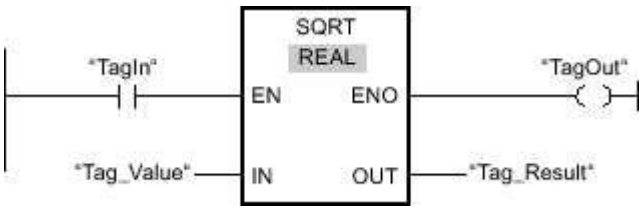
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	入力値の平方根

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

オペランド「TagIn」のシグナル状態が「1」の場合、「平方根の形成」命令が実行されます。この命令は、オペランド「Tag_Value」の平方根を計算し、結果を出力「Tag_Result」に送信します。この命令の実行中にエラーが発生しなかった場合、出力「TagOut」が設定されます。

LN: 自然対数の形成

説明

「自然対数の形成」命令を使用して、入力 IN の値の底 e (e=2.718282)に対する自然対数を計算できます。その結果は、出力 OUT に送信され、そこで照会できます。この命令は、入力値がゼロよりも大きい場合に正の結果になります。入力値がゼロ未満の場合、出力 OUT が無効な浮動小数点数を返します。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。
- 入力 IN の値が負。

パラメータ

次の表に、「自然対数の形成」命令のパラメータを示します。

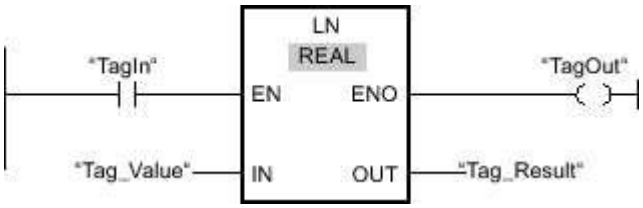
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	入力値の自然対数

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」のシグナル状態が「1」の場合、「自然対数の形成」命令が実行されます。この命令は、入力「Tag_Value」の値の自然対数を形成し、結果を出力「Tag_Result」に送信します。この命令の実行中にエラーが発生しなかった場合、出力「TagOut」が設定されます。

EXP: 指数値の形成

説明

「指数値の形成」命令を使用して、底 e (e=2.718282)および入力 IN で指定された値から指数を計算できます。出力 OUT で提供される結果を照会できます(OUT = e^{IN})。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「指数値の形成」命令のパラメータを示します。

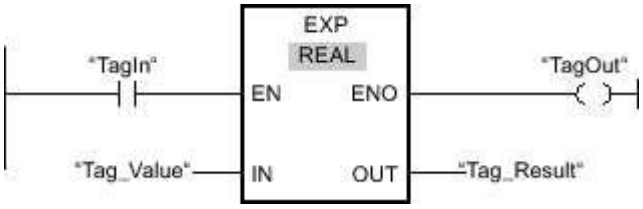
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	入力値 IN の指数値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn」のシグナル状態が「1」の場合、「指数値の形成」命令が実行されます。この命令は、底 e とオペランド「Tag_Value」の値から指数を形成し、結果を出力「Tag_Result」に送信します。この命令の実行中にエラーが発生しなかった場合、出力「TagOut」が設定されます。

SIN: 正弦値の形成

説明

角の正弦を計算するには、「正弦値の形成」命令を使用します。角度のサイズが、IN 入力でラジアン単位で指定されます。命令の結果は、OUT 出力に送信され、そこから照会できます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- IN 入力の値が有効な浮動小数点数でないこと。

パラメータ

次の表に、「正弦値の形成」命令のパラメータを示します。

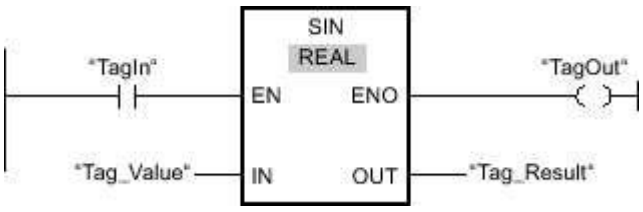
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	浮動小数点数	I、Q、M、D、L、P または定数	ラジアン単位の角度のサイズ
OUT	Output	浮動小数点数	I、Q、M、D、L、P	指定された角度の正弦

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

オペランド「TagIn」の信号状態が「1」の場合、「正弦値の形成」命令が実行されます。 この命令は、「Tag_Value」入力で指定された角度の正弦を計算し、結果を「Tag_Result」出力に保存します。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

COS: 余弦値の形成

説明

角の余弦を計算するには、「余弦値の形成」命令を使用します。角度のサイズが、IN 入力でラジアン単位で指定されます。命令の結果は、OUT 出力に送信され、そこから照会できます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- IN 入力の値が有効な浮動小数点数でないこと。

パラメータ

次の表に、「余弦値の形成」命令のパラメータを示します。

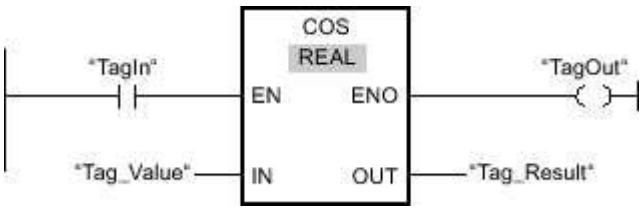
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	浮動小数点数	I、Q、M、D、L、P または定数	ラジアン単位の角度のサイズ
OUT	Output	浮動小数点数	I、Q、M、D、L、P	指定された角度の余弦

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

オペランド「TagIn」の信号状態が「1」の場合、「余弦値の形成」命令が実行されます。 この命令は、「Tag_Value」入力で指定された角度の余弦を計算し、結果を「Tag_Result」出力に保存します。 この命令が エラーなしで実行された場合、「TagOut」出力がセットされます。

TAN: 正接値の形成

説明

角の正接を計算するには、「正接値の形成」命令を使用します。 角度のサイズが、IN 入力でラジアン単位で指定されます。 命令の結果は、OUT 出力に送信され、そこから照会できます。

次のいずれかの条件が満たされる場合、ENO 許可出力のシグナル状態は「0」になります。

- EN 許可入力のシグナル状態が「0」であること。
- IN 入力の値が有効な浮動小数点数でないこと。

パラメータ

次の表に、「正接値の形成」命令のパラメータを示します。

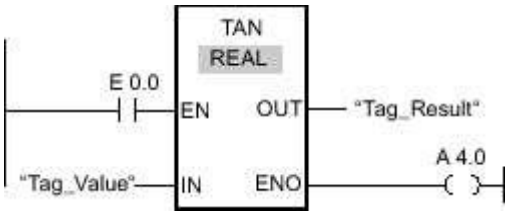
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	ラジアン単位の角度のサイズ
OUT	Output	浮動小数点数	I、Q、M、D、L、P	指定された角度の正接

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	+3.141593 (π)
OUT	Tag_Result	0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、「Tag_Value」入力で指定された角度の正接を計算し、結果を「Tag_Result」出力に保存します。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

ASIN: 逆正弦値の形成

説明

「逆正弦値の形成」命令を使用すると、この値と対応する入力 IN で指定された逆正弦値から角度の大きさを計算できます。 範囲-1 ~ +1 の有効な浮動小数点数のみを IN 入力に指定できます。 計算された角度の大きさは、弧度法を使用して OUT 出力に出力され、-π/2 ~ +π/2 の範囲の値を取ることができます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- IN 入力の値が有効な浮動小数点数でないこと。
- IN 入力の値が許容範囲外であること(-1 ~ +1)。

パラメータ

次の表に、「逆正弦値の形成」命令のパラメータを示します。

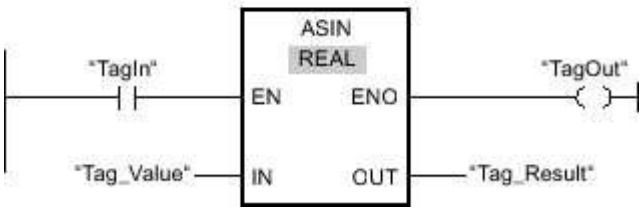
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	浮動小数点数	I、Q、M、D、L、P または定数	正弦値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

オペランド「TagIn」の信号状態が「1」の場合、「逆正弦値の形成」命令が実行されます。 この命令は、「Tag_Value」入力の正弦値に対応する角度のサイズを計算します。 命令の結果は、「Tag_Result」出力に保存されます。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

ACOS: 逆余弦値の形成

説明

「逆余弦値の形成」命令を使用すると、この値と対応する入力 IN で指定された余弦値から角度の大きさを計算できます。 範囲-1～+1 の有効な浮動小数点数のみを IN 入力に指定できます。 計算された角度の大きさは、弧度法を使用して OUT 出力に出力され、0～+π の範囲の値を取ることができます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- IN 入力の値が有効な浮動小数点数でないこと。
- IN 入力の値が許容範囲外であること(-1～+1)。

パラメータ

次の表に、「逆余弦値の形成」命令のパラメータを示します。

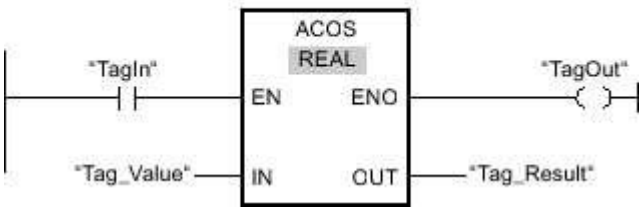
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	浮動小数点数	I、Q、M、D、L、P または定数	余弦値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	0

OUT	Tag_Result	+1.570796 (π/2)

オペランド「TagIn」の信号状態が「1」の場合、「逆余弦値の形成」命令が実行されます。 この命令は、「Tag_Value」入力の余弦値に対応する角度のサイズを計算します。 命令の結果は、「Tag_Result」出力に保存されます。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

ATAN: 逆正接値の形成

説明

「逆正接値の形成」命令を使用すると、この値と対応する入力 IN で指定された正接値から角度の大きさを計算できます。 入力 IN には、有効な浮動小数点数(または-NaN/+NaN)のみを指定することができます。 計算された角度の大きさは、弧度法を使用して OUT 出力に出力され、-π/2 ~ +π/2 の範囲の値を取ることができます。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「逆正接値の形成」命令のパラメータを示します。

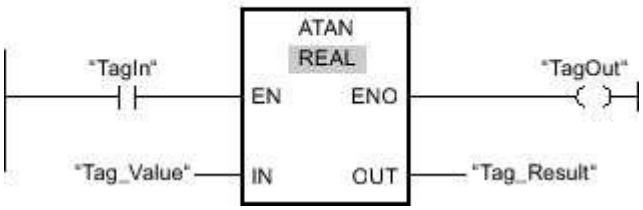
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	正接値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	1.0
OUT	Tag_Result	+0.785398 (π/4)

オペランド「TagIn」のシグナル状態が「1」の場合、「逆正接値の形成」命令が実行されます。この命令は、入力「Tag_Value」に対応する正接値の角度の大きさを計算します。命令の結果は、出力「Tag_Result」に格納されます。この命令の実行中にエラーが発生しなかった場合、出力「TagOut」が設定されます。

FRAC: 小数部を返す

説明

「小数部を返す」命令を使用して、IN 入力の値の小数位を求めることができます。 クエリの結果は OUT 出力に保存され、そこからクエリできます。 たとえば、IN 入力の値が 123.4567 の場合、OUT 出力が値 0.4567 を返します。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「小数部を返す」命令のパラメータを示します。

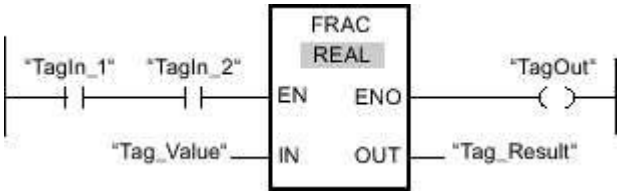
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	小数点以下の桁数を求める値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	IN 入力の値の小数位

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	2.555
OUT	Tag_Result	0.555

オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、「小数部を返す」命令が開始します。 オペランド「Tag_Value」の値の小数位が、オペランド「Tag_Result」にコピーされます。 この命令がエラーなしで実行された場合、ENO 出力は「1」の信号状態になり、「TagOut」出力がセットされます。

EXPT: 累乗

説明

「累乗」命令を使って、IN1 入力の値を IN2 入力の値で指定された指数で累乗することができます。 命令の結果は、OUT 出力に与えられ、そこで照会できます(OUT = $IN1^{IN2}$)。

IN1 入力には、有効な浮動小数点数のみを割り当てることができます。 IN2 入力には、整数も割り当てることができます。

次のいずれかの条件が満たされる場合、ENO イネーブル出力の信号状態は「0」になります。

- EN イネーブル入力の信号状態が「0」であること。
- 命令の実行中に、オーバーフローなどのエラーが発生していること。

パラメータ

次の表に、「累乗」命令のパラメータを示します。

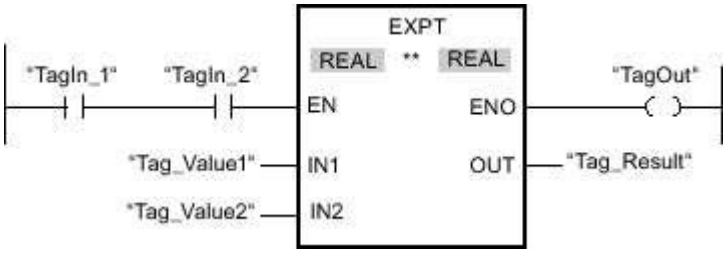
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN1	Input	浮動小数点数	I、Q、M、D、L、P または定数	底の値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	底の値が累乗される値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	結果

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、「累乗」命令が開始します。 オペランド「Tag_Value1」の値をオペランド「Tag_Value2」の値で指定された数で累乗できます。 結果が「Tag_Result」出力に保存されます。この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」となり、「TagOut」出力がセットされます。

移動操作

この章には下記に関する情報が記載されています：

MOVE: 移動値

説明

「値の移動」命令を使用し、IN 入力のオペランドの内容を OUT1 出力のオペランドに転送します。 この転送は、常にアドレスの昇順方向に行われます。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- IN パラメータのデータタイプが、OUT1 パラメータで指定されたデータタイプと一致しません。

次の表に、S7-1200 CPU シリーズで使用可能な転送方法をリストします。

転送元(IN)	宛先(OUT1)	
	IEC チェックあり	IEC チェックなし
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD、CHAR、文字列の文字 [1)]

WCHAR	WCHAR	BYTE、WORD、DWORD、CHAR、WCHAR、文字列の文字 1)
文字列の文字 1)	文字列の文字	CHAR、WCHAR、文字列の文字
ARRAY2)	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC データタイプ(UDT)	PLC データタイプ(UDT)	PLC データタイプ(UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUN-TER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

次の表に、S7-1500 CPU シリーズで使用可能な転送方法をリストします。

転送元(IN)	**宛先(OUT1)**	
	IEC チェックあり	**IEC チェックなし**
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD

UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD、CHAR、文字列の文字 [1)]
WCHAR	WCHAR	BYTE、WORD、DWORD、LWORD、CHAR、WCHAR、文字列の文字 [1)]
文字列の文字 [1)]	文字列の文字	CHAR、WCHAR、文字列の文字
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC データタイプ(UDT)	PLC データタイプ(UDT)	PLC データタイプ(UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER

IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUN-TER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNT-ER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_UL-COUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER

1) 「値のムーブ」命令を使用して、文字列の個々の文字を CHAR または WCHAR データタイプのオペランドに転送することもできます。 転送する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば「MyString[2]」は、「MyString」文字列の 2 番目の文字を転送します。 データタイプ CHAR または WCHAR のオペランドを文字列の個々の文字に転送することも可能です。 文字列の特定の文字を別の文字列の文字と置換することもできます。

2) 配列(ARRAY)全体の転送は、入力 IN および出力 OUT1 のオペランドの配列コンポーネントが同じデータタイプの場合のみ可能です。

IN 入力のデータタイプのビット長が OUT1 出力のデータタイプのビット長を超えると、転送元の値の上位ビットが失われます。入力 IN のデータタイプのビット長が出力 OUT1 のデータタイプのビット長を下回る場合、転送先の値の上位ビットはゼロで上書きされます。

初期状態では、命令ボックスには 1 出力が含まれています(OUT1)。 出力数は拡張できます。 追加された出力は、ボックスで昇順に番号付けされます。 この命令の実行中、入力 IN のオペランドの内容が、使用できるすべての出力に転送されます。 構造体データタイプ(DTL、STRUCT、ARRAY)または文字列の文字が転送される場合、命令ボックスは拡張できません。

「ブロックの移動」(MOVE_BLK)および「割り込み不可能なブロックの移動」(UMOVE_BLK)命令を使用して、ARRAY データタイプのオペランドを移動することもできます。 データタイプ STRING または WSTRING のオペランドは、「文字列のムーブ」(S_MOVE)を使用してコピーすることができます。

パラメータ

次の表に、「値のムーブ」命令のパラメータをリストします。

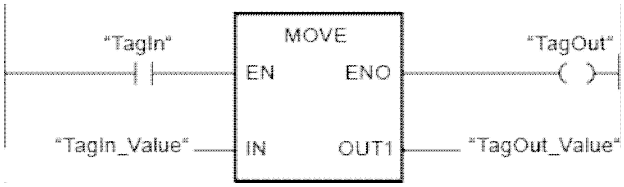
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、	ビット列、整数、浮動小数点数、タイマ、日付と時刻、	I、Q、M、D、L、または定数	ソース値

		CHAR、WCHAR、STRUCT、ARRAY、IEC データタイプ、PLC データタイプ (UDT)	CHAR、WCHAR、STRUCT、ARRAY、TIMER、COUNTER、IEC データタイプ、PLC データタイプ(UDT)		
OUT1	Output	ビット列、整数、浮動小数点数、タイマ、日付と時刻、CHAR、WCHAR、STRUCT、ARRAY、IEC データタイプ、PLC データタイプ (UDT)	ビット列、整数、浮動小数点数、タイマ、日付と時刻、CHAR、WCHAR、STRUCT、ARRAY、TIMER、COUNTER、IEC データタイプ、PLC データタイプ(UDT)	I、Q、M、D、L	ソースで値が転送されるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

オペランド「TagIn」のシグナル状態が「1」の場合、「値の移動」命令が実行されます。 この命令は、オペランド「TagIn_Value」の内容をオペランド「TagOut_Value」にコピーし、出力「TagOut」のシグナル状態を「1」にセットします。

逆シリアル化: 逆シリアル化

説明

「逆シリアル化」命令を使用して、PLC データタイプ(UDT)のシーケンシャル表示を PLC データタイプに変換して戻し、その内容全体を表示することができます。

PLC データタイプのシーケンシャル表示を配置するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。変換の前に、十分なメモリ領域が存在することを確認してください。

この命令では、変換された PLC データタイプの複数のシーケンシャル表示をそれらの元の状態に、順を追って変換し直すことができます。

PLC データタイプ(UDT)の単一のシーケンシャル表示のみを変換して元に戻す場合は、命令「TRCV: 通信接続経由のデータ受信」を指示することもできます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

パラメータ

次の表に、「逆シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC_ARRAY	Input	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるグローバルデータブロック
DEST_VARIABLE	InOut	VARIANT	I、Q、M、L	返された変換済み PLC データタイプ(UDT)が格納されるタグ
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

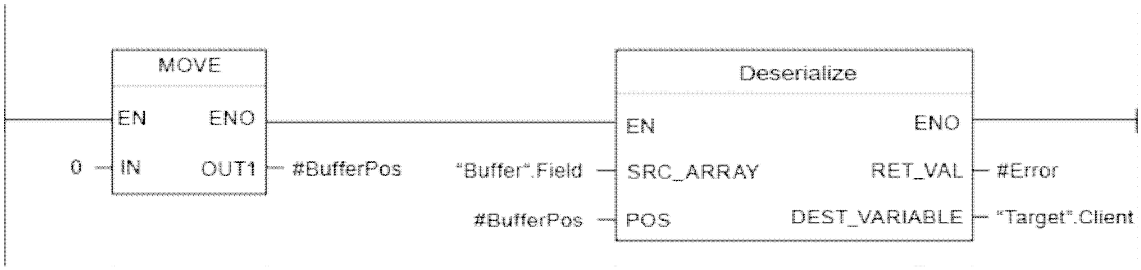
エラーコード*	説明

(W#16#...)	
0000	エラーは発生していません。
80B0	SRC_ARRAY パラメータと DEST_VARIABLE パラメータのためのメモリ領域がオーバーラップしています。
8136	DEST_VARIABLE パラメータのデータブロックが標準アクセスによるブロックではありません。
8150	SRC_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8151	SRC_ARRAY パラメータのコード生成エラー
8153	SRC_ARRAY パラメータのメモリの空き領域が不足しています。
8250	DEST_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8251	DEST_VARIABLE パラメータのコード生成エラー
8254	DEST_VARIABLE パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

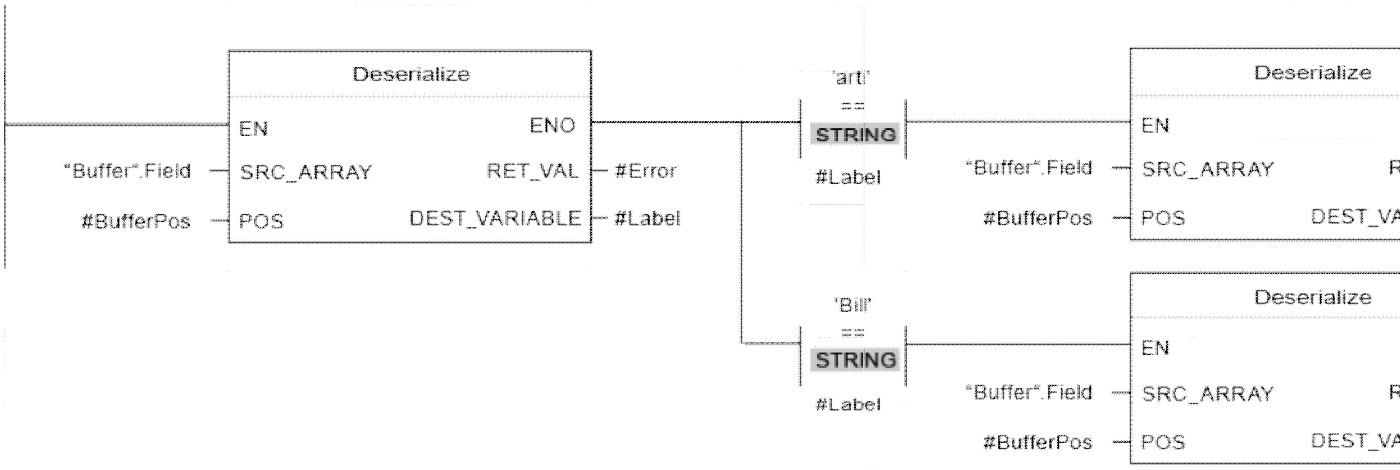
次の例で、命令がどのように動作するかを示します。

ネットワーク 1:



「値のムーブ」命令は、「#BufferPos」オペランドに値「0」を移動します。 「逆シリアル化」命令は、Buffer」データブロックのカスタマデータのシーケンシャル表示を逆シリアル化し、「Target」データブロックに書き込みます。 変換済みカスタマデータで使用されるバイト数は、「#BufferPos」オペランドに格納されます。

ネットワーク 2:



「逆シリアル化」命令は、「Buffer」データブロックの、カスタマデータの後にシーケンシャル表示で保存されたセパレータシートのシーケンシャル表示を逆シリアル化し、これらの文字を「#Label」オペランドに書き込みます。 これらの文字は、比較命令「arti」および「Bill」を使用して比較されます。 「arti」 = TRUE の場合の比較では、データは、逆シリアル化され、「Target」データブロックに書き込まれたアーティクルデータです。 「Bill」 = TRUE の場合の比較では、データは、逆シリアル化され、「Target」データブロックに書き込まれたビリングデータです。

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言を示します。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Target	Client	「クライアント」(PLC データタイプ)

	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
	Bill	INT の配列[0..10]
Buffer	Field	バイトの配列[0..294]

シリアル化: シリアル化:

説明

「シリアル化」命令を使用して、データタイプの構造体の部分を失うことなく、複数の PLC データタイプ(UDT)を 1 つのシーケンシャル表示に変換することができます。

この命令を使用して、ユーザプログラムの複数の構造体データをキャッシュバッファ(グローバルデータブロックを推奨)に格納し、別の CPU に送信することができます。 変換済み PLC データタイプを格納するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。 変換の前に、十分なメモリ領域が存在することを確認してください。

POS パラメータのオペランドには、変換済み PLC データタイプによって使用されるバイト数に関する情報が収納されています。

単一の PLC データタイプ(UDT)を送信する場合、直接に命令「TSEND: 通信接続経由のデータ送信」を呼び出すことができます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセスが付加されているブロックでは、ビットの長さは 1 バイトです。

パラメータ

次の表に、「シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC_VARIABLE	Input	VARIANT	I、Q、M、L	シーケンシャル表示に変換される PLC データタイプ (UDT)。
DEST_ARRAY	InOut	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるデータブロック。
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

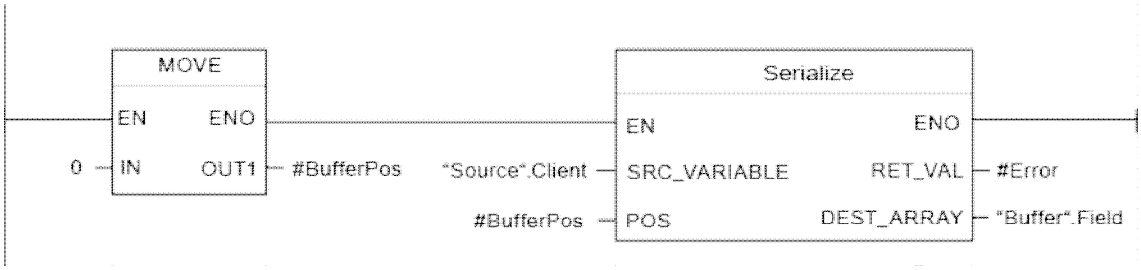
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_VARIABLE パラメータと DEST_ARRAY パラメータのためのメモリ領域がオーバーラップしています。
8150	SRC_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8152	SRC_VARIABLE パラメータのコード生成エラー
8236	DEST_ARRAY パラメータのデータブロックが標準アクセスによるブロックではありません。
8250	DEST_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8252	DEST_ARRAY パラメータのコード生成エラー
8253	DEST_ARRAY パラメータのメモリの空き領域が不足しています。
8254	DEST_ARRAY パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

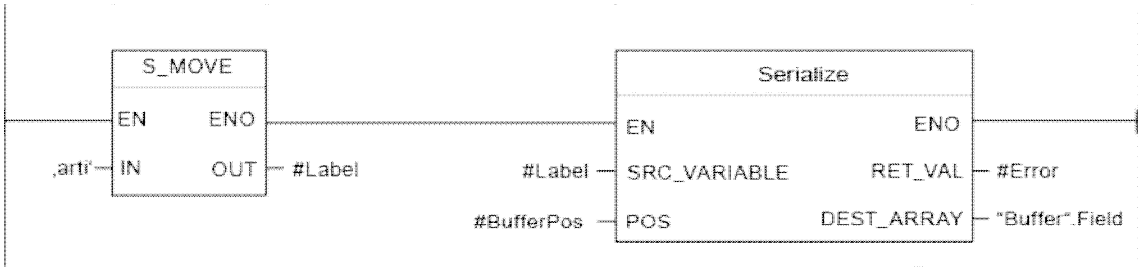
次の例で、命令がどのように動作するかを示します。

ネットワーク 1:



「値のムーブ」命令は、「#BufferPos」オペランドに値「0」を移動します。 「シリアル化」命令は、「Source」データブロックのカスタマデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックに書き込みます。 シーケンシャル表示によって使用されたバイト数は、「#BufferPos」オペランドに格納されます。

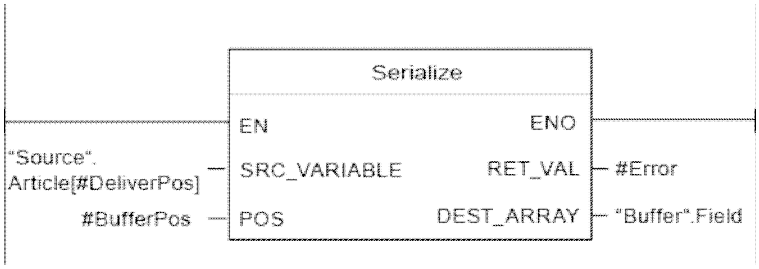
ネットワーク 2:



後でのシーケンシャル表示の逆シリアル化を容易にするために、その時点では、一種のセパレータシートが挿入されます。 「文字列のムーブ」命令は、「arti」文字を「#Label」オペランドに移動しま

す。 「シリアル化」命令は、これらの文字を「Buffer」データブロックのカスタマデータの後に書き込みます。 文字が必要とするバイト数が、「#BufferPos」オペランドに格納済みの数に加算されます。

ネットワーク 3:



「シリアル化」命令は、「Source」データブロックの、ランタイム時に計算される特定のアーティクルのデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックの「arti」文字の後に書き込みます。

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言を示します。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Source	Client	「クライアント」(PLC データタイプ)
	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
Buffer	Field	バイトの配列[0..294]

MOVE_BLK: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 移動先の領域に移動するエレメントの数は、入力 COUNT で指定されます。移動するエレメントの幅は、入力 IN のエレメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

パラメータ

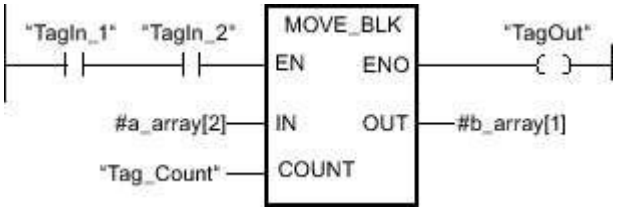
次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	移動元領域から移動先領域に移動するエレメントの数。
OUT [1)]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント
[1)] 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは Array [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは Array [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、「ブロックの移動」命令が実行されます。 この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

MOVE_BLK_VARIANT:ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 配列全体または配列の工レメントを同じデータタイプの別の配列にコピーすることができます。 ソース配列と宛先配列のサイズ(工レメントの数)は異なる場合があります。配列内の複数工レメントまたは単一工レメントをコピーすることができます。

ブロックの作成時にこの命令を使用する場合、VARIANT ごとにソースおよび宛先が転送されるため、配列はまだ既知である必要はありません。

SRC_INDEX および DEST_INDEX パラメータでのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- コピーされるデータが使用可能なデータよりも多いこと。

注記

BOOL データタイプと接続している VARIANT

データタイプ VARIANT のパラメータ(ソース領域または宛先領域)とデータタイプ BOOL のタグまたは ARRAY of BOOL を相互接続する場合、以下のオプションがあります。

1. シンボリックにアドレス指定することができます。

   例: SRC パラメータ: "Data_block".myArray
2. 任意のポインタによってアドレス指定することができます。 ただし、領域で指定された長さは 8 で割り切れるようにする必要があります。割り切れない場合、この命令は実行されません。

   例: SRC パラメータ: P#DB123.DBX456.0 BOOL 1000

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC	Input	VARIANT (配列または個々の配列要素を指します)、<Data_type>の配列	I、Q、M、L	コピー元のソースブロック
COUNT	Input	UDINT	I、Q、M、D、L、または定数	コピーされるエレメント数 ARRAY が SRC パラメータまたは DEST パラメータで指定されていない場合は、

				COUNT パラメータに値「1」を割り当てます。
SRC_INDEX	Input	DINT	I、Q、M、D、L、または定数	• SRC_INDEX パラメータは、ゼロベースで計算されます。 ARRAY がパラメータ SRC で指定されると、パラメータ SRC_INDEX の整数はコピー元のソース領域内の最初の エレメントを指定します。 宣言された配列の限界値には関係ありません。 • ARRAY がパラメータ SRC で指定されないか、または配列の 1 つの単一エレメントのみが指定された場合、パラメータ SRC_INDEX に値「0」を割り当てます。
DEST_INDEX	Input	DINT	I、Q、M、D、L、または定数	• DEST_INDEX パラメータは、ゼロベースで計算されます。 ARRAY がパラメータ DEST で指定されると、パラメータ DEST_INDEX の整数はコピー先の宛先領域内の最初の エレメントを指定します。 宣言された配列の限界値には関係ありません。 • ARRAY が DEST パラメータで指定されていない場合は、DEST_INDEX パラメータに値「0」を割り当てます。
DEST	Output [1)]	VARIANT	I、Q、M、L	ソースブロックの内容がコピーされる宛先領域。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
1) データがタグに流れるため、DEST パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

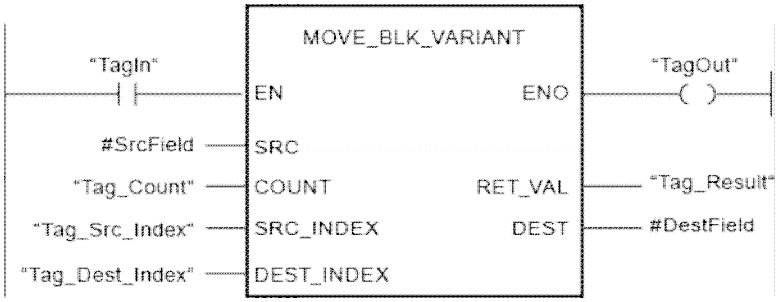
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	データタイプが一致していません
8151	SRC パラメータにアクセスできません。
8152	SRC パラメータのオペランドが入力されていません。
8153	SRC パラメータのコード生成エラー
8154	SRC パラメータのオペランドのデータタイプが BOOL です。
8281	COUNT パラメータの値が無効です
8382	SRC_INDEX パラメータの値が VARIANT の限界値外です。
8383	SRC_INDEX パラメータの値が配列の上限値外です
8482	DEST_INDEX パラメータの値が VARIANT の限界値外です。
8483	DEST_INDEX パラメータの値が配列の上限値外です
8534	DEST パラメータが書き込み保護されています
8551	DEST パラメータにアクセスできません。
8552	DEST パラメータのオペランドが入力されていません。
8553	DEST パラメータのコード生成エラー
8554	DEST パラメータのオペランドのデータタイプが BOOL です。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
SRC	Input	#SrcField	ローカルオペランド #SrcField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。

			("MOVE_UDT"の ARRAY[0..10]
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	ローカルオペランド #DestField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。 ("MOVE_UDT"の ARRAY[10..20]

オペランド「TagIn」のシグナル状態が「1」の場合、「ブロックムーブ」命令が実行されます。 UDT の配列の 4 番目のエレメントから始めて、2 つのエレメントが、ソース領域から宛先領域にコピーされます。 コピーが、UDT の配列の 4 番目のエレメントから始めて、挿入されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

UMOVE_BLK: 割り込みなしブロック転送

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 移動先の領域に移動する工レメントの数は、COUNT パラメータで指定されます。移動する工レメントの幅は、入力 IN の工レメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	コピーされるソース領域の最初の工レメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	移動元領域から移動先領域に移動する工レメントの数。
OUT [1]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、	D、L	ソース領域の内容がコピーされる宛先領

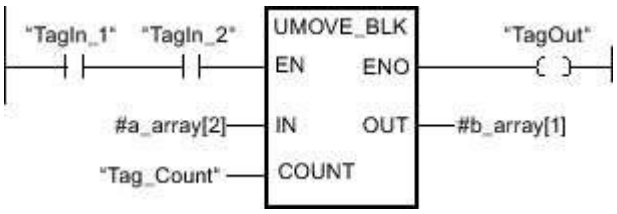
		CHAR、WCHAR、TOD	WCHAR、TOD、LTOD		域の最初のエレメント

[1] 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは Array [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは Array [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、「割り込みなしブロック転送」命令が実行されます。 この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 この命令がエラーなしで実行された場合、ENO 出力は「1」のシグナル状態になり、「TagOut」出力がセットされます。

FILL_BLK: ブロックの塗りつぶし

説明

「フィル」命令を使用して、IN 入力の値でメモリ領域(宛先領域)を埋めることができます。宛先領域は、OUT 出力で指定されたアドレスから開始して埋められます。ムーブ操作の繰り返し回数は、COUNT パラメータの値で指定されます。この命令が実行された場合、入力 IN の値が選択され、COUNT パラメータの値で指定された回数だけ移動先の領域に移動されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

パラメータ

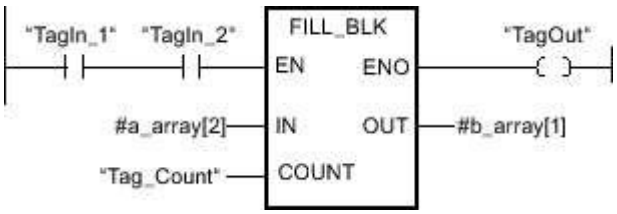
次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	ムーブ操作の繰り返し回数
OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	フィル命令が開始する宛先領域のアドレス
[1] 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。 [2] 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは Array [0..5] of INT です。これは、INT データタイプの 6 つの工レメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは Array [0..6] of INT です。これは、INT データタイプの 7 つの工レメントで構成されます。

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、「フィル」命令が実行されます。 この命令は、#a_array タグの 3 番目の工レメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。 この命令がエラーなしで実行された場合、ENO および "TagOut" 許可出力がシグナル状態「1」にセットされます。

UFILL_BLK: 割り込みなしフィルブロック

説明

「割り込み不可能なフィル」命令を使用して、割り込まれることなく IN 入力の値でメモリ領域(宛先領域)を満たすことができます。宛先領域は、OUT 出力で指定されたアドレスから開始して埋められます。ムーブ操作の繰り返し回数は、COUNT パラメータの値で指定されます。この命令が実行された場合、入力 IN の値が選択され、COUNT パラメータの値で指定された回数だけ移動先の領域に移動されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込み不可能なフィル」命令の実行中には、CPU のアラーム応答時間が長くなります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

「割り込み不可能なフィル」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

パラメータ

次の表に、「割り込み不可能なフィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	ムーブ操作の繰り返し回数
OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、	2 進数、整数、浮動小数点数、タイマ、DATE、	D、L	フィルが開始する宛先領域のアドレス

		CHAR、WCHAR、TOD	CHAR、WCHAR、TOD、LTOD		

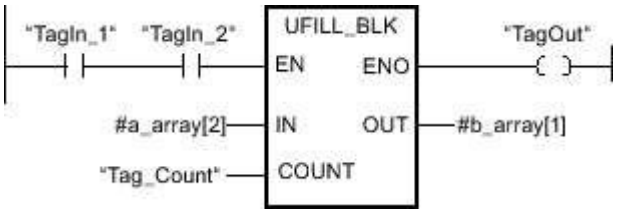
[1] 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。

[2] 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは Array [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは Array [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

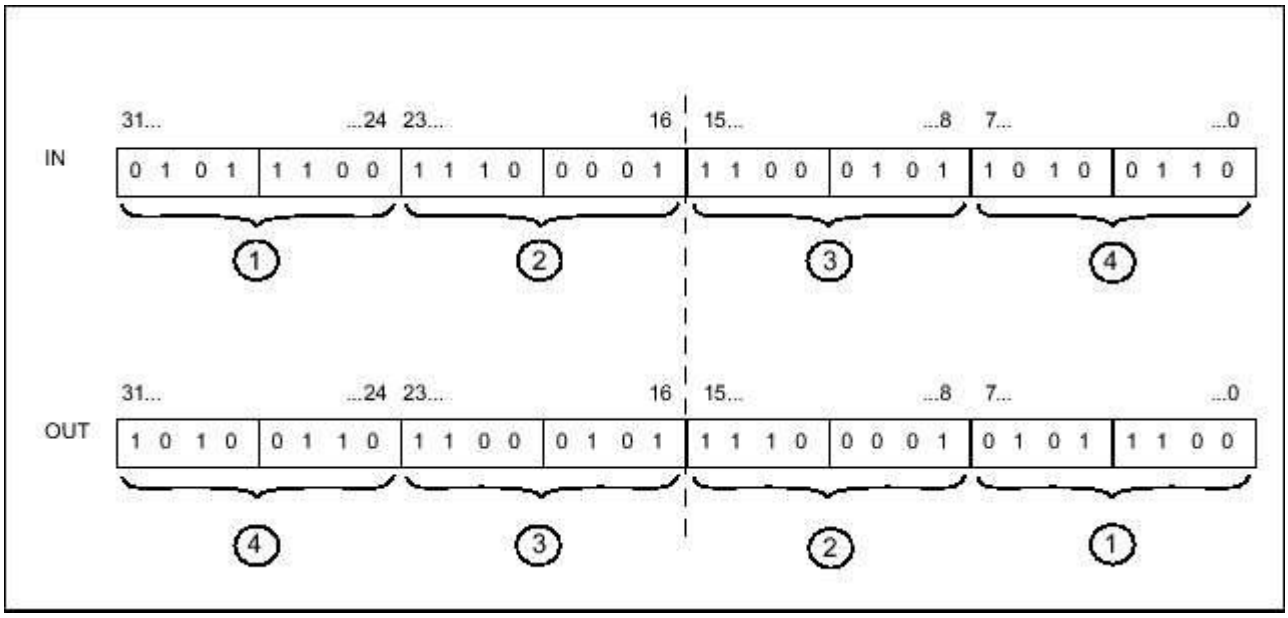
オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、「割り込み不可 n フィル」命令が実行されます。 この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 この命令が工ラーなしで実行された場合、ENO および "Tag-Out" 許可出力がシグナル状態「1」にセットされます。

SWAP: スワップ

説明

「スワップ」命令を使用して、入力 IN のバイトの順番を変更し、出力 OUT で結果を照会することができます。

次の図に、DWORD データタイプのオペランドのバイトが、「スワップ」命令を使用してスワップされる様子を示します。



パラメータ

次の表に、「スワップ」命令のパラメータをリストします。

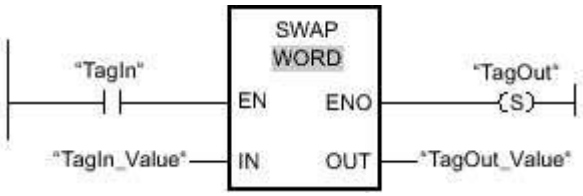
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I、Q、M、D、L、または P 定数	バイトがスワップされるオペランド
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I、Q、M、D、L、P	結果

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

オペランド「TagIn」の信号状態が「1」の場合、「スワップ」命令が実行されます。 バイトの順序が変更され、オペランド「TagOut_Value」に保存されます。

配列 DB

この章には下記に関する情報が記載されています：

- ReadFromArrayDB: 配列データブロックからの読み出し (S7-1500)
- WriteToArrayDB: 配列データブロックへの書き込み (S7-1500)
- ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し (S7-1500)
- WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み (S7-1500)

ReadFromArrayDB: 配列データブロックからの読み出し

説明

「配列データブロックからの読み出し」命令を使用し、配列データブロックからデータを読み出して、宛先領域に書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列での カウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	Output 1)	VARIANT	I、Q、M、L	読み出されて出力される値
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
1) データがタグに流れるため、VALUE パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

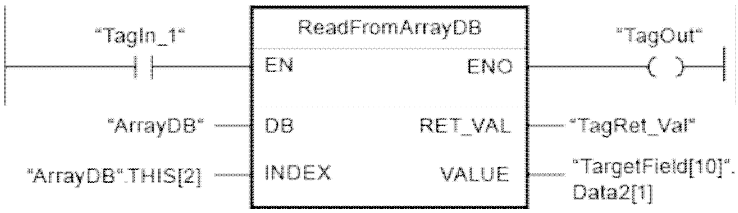
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、書き込み保護されているか、またはロードメモリ内に存在します。
8135	配列データブロックに無効な値が含まれます。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8450	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8452	コード生成エラー
8453	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	TargetField[10].Data2[1]	「TargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[10 to 20]です。

「TagIn1」オペランドのシグナル状態が「1」の場合、「ARRAY データブロックからの読み出し」命令が実行されます。「ArrayDB」から 3 番目のエレメントが読み出され、「TargetField[10].Data2[1]」オペランドに書き込まれます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

WriteToArrayDB: 配列データブロックへの書き込み

説明

「配列データブロックへの書き込み」命令を使用して、配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の工レメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内の工レメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

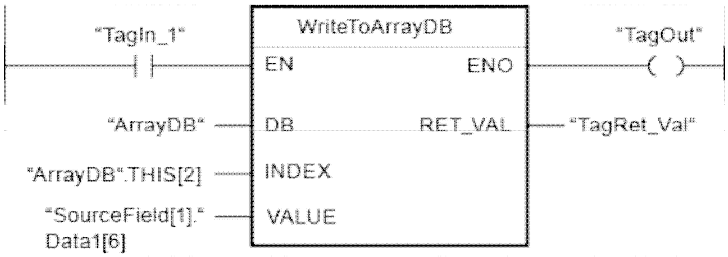
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存された工レメントのデータタイプが、VARIANT で転送される工レメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。

8134	データブロックが書き込み保護されています。
8135	データブロックが配列データブロックではありません。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8350	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8352	コード生成エラー
8353	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	SourceField[1].Data1[6]	「SourceField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。

「TagIn1」オペランドのシグナル状態が「1」の場合、「ARRAY データブロックへの書き込み」命令が実行されます。 「SourceField」オペランドから、2 番目のエレメントの「Data1[6]」エレメントが「ArrayDB」に書き込まれます。 3 番目のエレメントは、「ArrayDB」に書き込まれます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し

説明

「ロードメモリの配列データブロックからの読み出し」命令を使用して、ロードメモリの配列データブロックからデータを読み出します。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 この命令は、BUSY パラメータで信号立ち下がりエッジが検出された場合に終了します。 1 プログラムサイクルの間、DONE パラメータのシグナル状態が「1」になり、このサイクル内に、読み取られた値が VALUE パラメータに出力されます。 他のすべてのプログラムサイクルでは、VALUE パラメータの値は変更されません。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「ロードメモリの配列データブロックからの読み出し」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」: 配列 DB の読み出しから開始
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	InOut	VARIANT	I、Q、M、L	読み出されて出力される値

				TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ読み出し中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

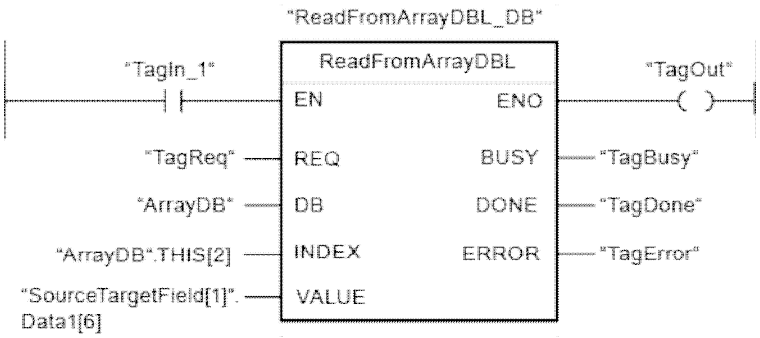
次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	**説明**
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL: ロードメモリのデータブロックからの読み出し」および「WRIT_DBL: データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	SourceTargetField[1].Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagIn1」オペランドのシグナル状態が「1」で、「TagReq」オペランドで立ち上がりエッジが検出されると、「ロードメモリの配列データブロックからの読み出し」命令が実行されます。「ArrayDB」から 3 番目のエレメントが読み出され、「SourceTargetField[1].Data1[6]」オペランドに書き込まれます。「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値は変更されません。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み

説明

「ロードメモリの配列データブロックへの書き込み」命令を使用して、ロードメモリの配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 BUSY パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、VALUE パラメータの値がデータブロックに書き込まれます。 DONE1 パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、パラメータの値がデータブロックに書き込まれます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「ロードメモリの配列データブロックへの書き込み」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」:配列 DB への書き込みを開始
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内のエレメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値

				TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ書き込み中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

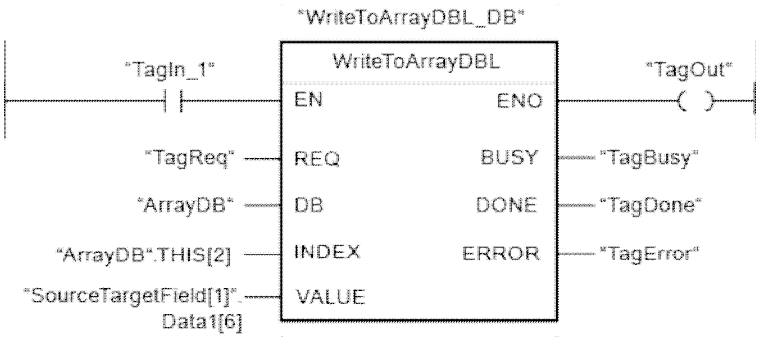
次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存された工レメントのデータタイプが、VARIANT で転送される工レメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8234	データブロックが書き込み保護されています。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックの工レメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL: ロードメモリのデータブロックからの読み出し」および「WRIT_DBL: データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	SourceTargetField[1].Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagIn1」オペランドのシグナル状態が「1」で、「TagReq」オペランドで立ち上がりエッジが検出された場合、「ロードメモリの配列データブロックへの書き込み」命令が実行されます。 「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値が「ArrayDB」の 3 番目のエレメントに書き込まれます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

VARIANT

この章には下記に関する情報が記載されています：

- VariantGet: VARIANT タグ値の読み出し (S7-1200, S7-1500)
- VariantPut: VARIANT タグ値の書き込み (S7-1200, S7-1500)
- CountOfElements: 配列エレメントの数を取得 (S7-1200, S7-1500)

VariantGet: VARIANT タグ値の読み出し

説明

「VARIANT タグ値の読み出し」命令を使用して、SRC パラメータの VARIANT が指すタグの値を読み出し、その値を DST パラメータのタグに書き込むことができます。

SRC パラメータのデータタイプは、VARIANT です。 DST パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

DST パラメータのタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT:ブロックムーブ」命令を使用できます。追加情報については、「関連項目」を参照してください。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- データタイプが一致していません。 (値は転送されません。)

パラメータ

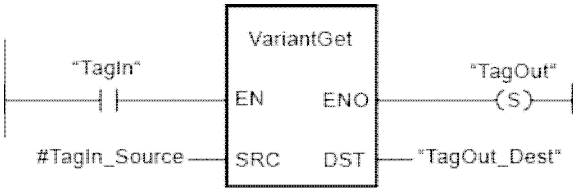
次の表に、「VARIANT タグ値の読み出し」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC	Input	VARIANT	I、Q、M、L	読み出されるタグ
DST	Output	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 「#TagIn_Source」オペランドの VARIANT が指すタグの値が読み出され、「TagOut_Dest」オペランドに書き込まれます。

VariantPut: VARIANT タグ値の書き込み

説明

「VARIANT タグ値の書き込み」命令を使用して、SRC パラメータのタグの値を VARIANT が指す DST パラメータのタグに書き込むことができます。

DST パラメータのデータタイプは、VARIANT です。 SRC パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

SRC パラメータのタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT:ブロックムーブ」命令を使用できます。追加情報については、「関連項目」を参照してください。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- データタイプが一致していません。 (値は転送されません。)

パラメータ

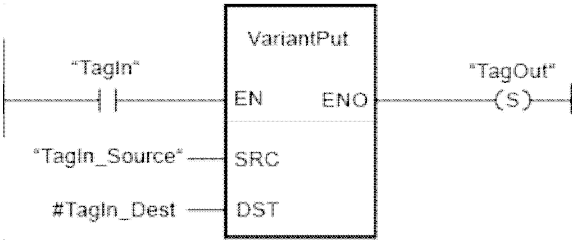
次の表に、「VARIANT タグ値の書き込み」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	読み出されるタグ
DST	Input	VARIANT	I、Q、M、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 「TagIn_Source」オペランドの値が、「#TagIn_Dest」オペランドの VARIANT が指すタグに書き込まれます。

CountOfElements: 配列エレメントの数を取得

説明

「ARRAY エレメント数の取得」命令を使用して、VARIANT を指すタグが所有する ARRAY エレメント数を照会できます。

配列が一次元の ARRAY の場合、上限値と下限値の差 + 1 が結果として出力されます。 配列が多次元の ARRAY の場合、すべての次元の積が結果として出力されます。

IN パラメータのデータタイプは、VARIANT であることが必要です。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- VARIANT タグが ARRAY でないこと。 (結果は「0」です。)

VARIANT がブールの配列を指す場合、FILL エレメントもカウントに含まれます。 (たとえば、ブールの配列[0..1]の場合、8 が返されます)

パラメータ

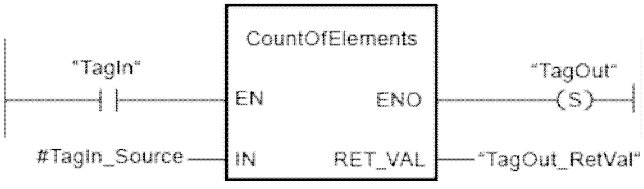
次の表に、「ARRAY エレメント数の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	VARIANT	I、Q、M、L	照会するタグ
RET_VAL	Output	UDINT	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



TagIn」オペランドのシグナル状態が「1」の場合、「ARRAY エレメント数の取得」命令が実行されます。 「#TagIn_Source」オペランドの VARIANT が指すタグの ARRAY エレメント数が読み出され、「TagOut_RetVal」オペランドに出力されます。

レガシー

この章には下記に関する情報が記載されています：

- FieldRead: フィールドの読み出し (S7-1200, S7-1500)
- FieldWrite: フィールドの書き込み (S7-1200, S7-1500)
- BLKMOV: ブロックの移動 (S7-1500)
- UBLKMOV: 割り込みなしブロック転送 (S7-1500)
- FILL: フィル命令 (S7-1500)

FieldRead: フィールドの読み出し

説明

「フィールドの読み出し」命令を使用して、入力 MEMBER で指定されたフィールドから特定のコンポーネントを読み出して、その内容を出力 VALUE でタグに転送することができます。 入力 INDEX で読み取るフィールドコンポーネントのインデックスを指定します。 読み取りが発生するフィールドの最初のコンポーネントを入力 MEMBER で指定します。

パラメータ MEMBER のフィールドコンポーネントのデータタイプ、パラメータ VALUE のインデックスおよびタグは、暗黙的変換が不可能なため「フィールドの読み出し」命令のデータタイプに対応する必要があります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 INDEX で指定されたフィールドコンポーネントが出力 MEMBER で指定されたフィールドで定義されていないこと。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「フィールドの読み出し」命令のパラメータを示します。

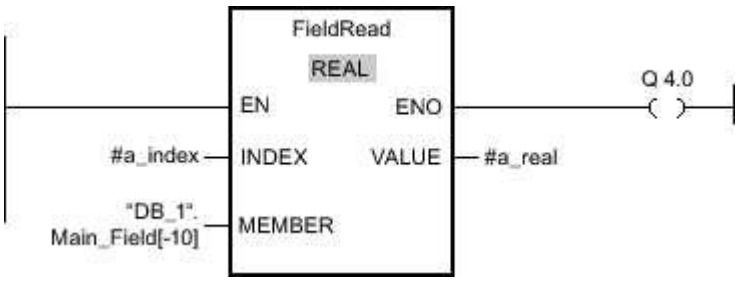
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
INDEX	Input	DINT	DINT	I、Q、M、D、L、P、または定数	内容が読み出されるフィールドコンポーネントのインデックス
MEMBER	Input	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、および WCHAR	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、タイマ、DATE、TOD、LTOD、CHAR、および WCHAR	D、L	読み取りが発生するフィールドの最初のコンポーネント。
VALUE	Output	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、LTOD、CHAR、WCHAR	I、Q、M、D、L、P	フィールドコンポーネントの内容が転送されるオペランド

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	タグ	値
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	"DB_1"データブロックの"Main_Field[-10..10] of REAL"フィールドの最初のコンポーネント
VALUE	a_real	「Main_Field[-10..10] of REAL」フィールドのインデックス 4 のコンポーネント

インデックス 4 のフィールドコンポーネントが「Main_Field[-10...10] of REAL」フィールドから読み取られ、「a_real」タグに書き込まれます。 読み取られるフィールドコンポーネントは、入力 INDEX の値で指定されます。

FieldWrite: フィールドの書き込み

説明

「フィールドの書き込み」命令を使用して、VALUE 入力のタグの内容を MEMBER 出力のフィールドの特定のコンポーネントに転送します。 INDEX 入力の値を使用し、記述されるフィールドコンポーネントのインデックスを指定することができます。 MEMBER 出力で、書き込まれるフィールドの最初のコンポーネントを入力します。

パラメータ MEMBER のフィールドコンポーネントのデータタイプ、パラメータ VALUE のインデックスおよびタグは、暗黙的変換が不可能なため「フィールドの読み出し」命令のデータタイプに対応する必要があります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 INDEX で指定されたフィールドコンポーネントが出力 MEMBER で指定されたフィールドで定義されていないこと。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「フィールドの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
INDEX	Input	DINT	DINT	I、Q、M、D、L、P、または定数	VALUE の内容とともに書き込まれているフィールドコンポーネントのインデックス。
VALUE	Input	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、LTOD、CHAR、WCHAR	I、Q、M、D、L、P、または定数	内容がコピーされるオペランド。
MEMBER	Output	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、タイマ、DATE、TOD、LTOD、	D、L	VALUE の内容が書き込まれるフィールドの最初のコンポーネント。

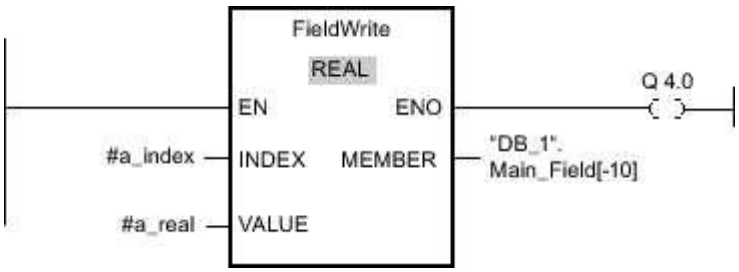
		および WCHAR	CHAR、および WCHAR		

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	"DB_1"データブロックの"Main_Field[-10..10] of REAL"フィールドの最初のコンポーネント

タグ「a_real」の値「10.54」が、Main_Field[-10..10] of REAL フィールドのインデックス 4 のフィールドコンポーネントに書き込まれます。 タグ「a_real」の内容が転送されるフィールドコンポーネントのインデックスは、入力 INDEX の値によって指定されます。

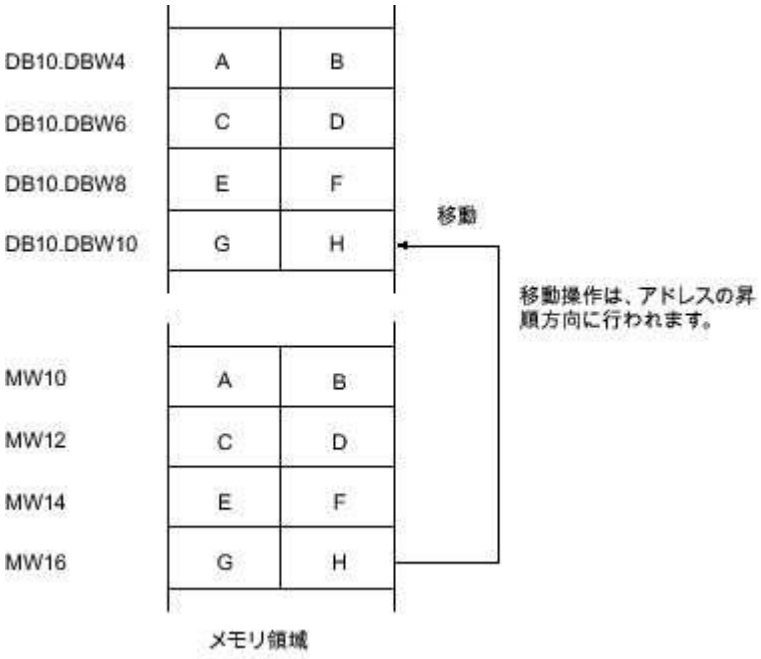
BLKMOV: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

次の図に、ムーブ操作の原理を示します。



ソース領域および宛先領域の整合性

「ブロックムーブ」命令の実行中にソースデータが変更されないことを確認してください。 変更があった場合、宛先データの整合性は保証できません。

割り込み機能

ネストレベルに対する制限はありません。

メモリ領域

「ブロックの移動」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力

- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。ソース領域と宛先領域の長さが異なる場合、小さな領域の長さのみが移動します。

ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

文字列の移動のルール

STRING データタイプの移動元および移動先の領域の移動に「ブロックの移動」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報も、宛先領域に書き込まれます。ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。

文字列の最大長および実際の長さに関する情報を移動するには、SRCBLK および DSTBLK パラメータにバイト単位で領域を指定します。

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output 1)	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

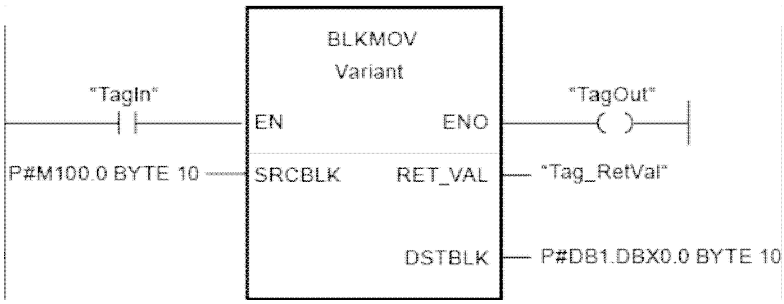
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみで使用できます。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。ムーブ操作中にエラーが発生した場合、「Tag_RetVal」タグにエラーコードが出力されます。

UBLKMOV: 割り込みなしブロック転送

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。その結果、「割り込み不可能なブロックムーブ」命令の実行中に、CPU の割り込み応答時間が長くなる場合があります。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

メモリ領域

「割り込みなしブロック転送」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

「割り込みなしブロック転送」命令の実行中は、ソース領域および宛先領域が重複してはなりません。ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

仮パラメータとして定義されたソース領域または宛先領域が、SRCBLK または DSTBLK パラメータで指定された宛先領域またはソース領域よりも小さい場合、データは転送されません。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

文字列の移動のルール

STRING データタイプのソース領域および宛先領域の移動に「割り込みなしブロック転送」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報は、宛先領域には書き込まれません。ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。 STRING データタイプの領域を移動する場合、領域の長さとして「1」を指定します。

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output [1)]	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

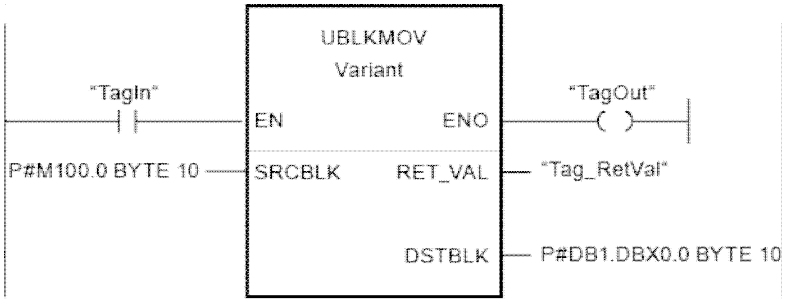
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8091	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。ムーブ操作中にエラーが発生した場合、「Tag_RetVal」タグにエラーコードが出力されます。

FILL: フィル命令

説明

「フィル」命令を使用して、メモリ領域(宛先領域)を別のメモリ領域(ソース領域)の内容によって埋めることができます。 「フィル」命令は、宛先領域がすべて書き込まれるまでソース領域の内容を宛先領域に移動します。 ムーブ操作は、アドレスの昇順方向に行われます。

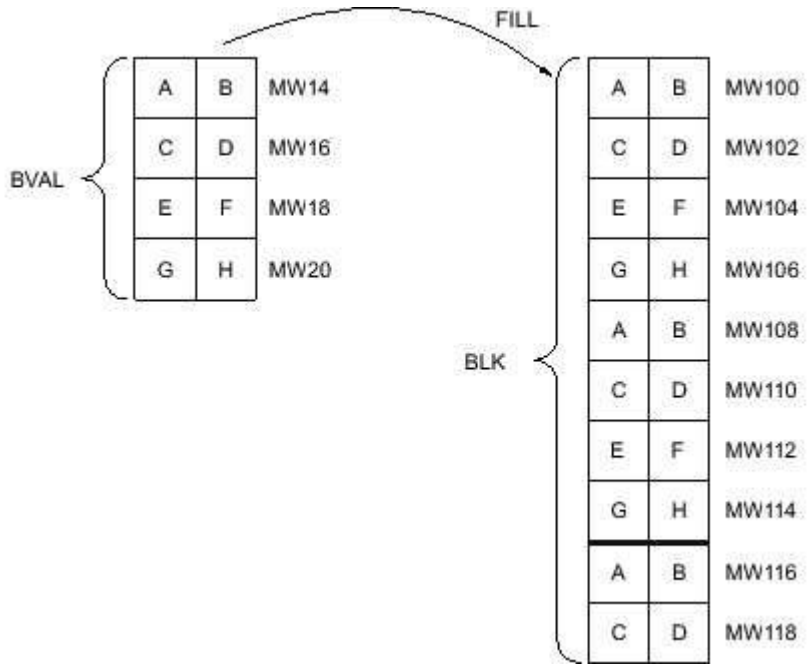
VARIANT を使用して、ソース領域と宛先領域を定義します。

注記

この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

「最適化したブロックアクセス」属性を持つブロックの場合、命令「FILL_BLK: フィル」命令を使用できます。

次の図に、ムーブ操作の原理を示します。



例: MW100～MW118 の範囲の内容は、メモリワード MW14～MW20 の内容で事前割り当てされます。

ソース領域および宛先領域の整合性

「フィル」命令の実行中は、宛先データの整合性を保証するため、ソースデータは変更されないことに注意してください。

メモリ領域

「フィル」命令を使用して、以下のメモリ領域を移動することができます。

- データブロックの領域

- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。 プリセットする宛先領域が入力パラメータ BVAL の長さの整数倍ではなくても、宛先領域は最後のバイトまで書き込まれます。

プリセットする宛先領域がソース領域よりも小さい場合は、ソース領域に含まれているデータのうち、宛先領域に書き込める量のみがコピーされます。

実際に存在する宛先領域またはソース領域がソース領域または宛先領域に割り当てられたメモリ領域よりも小さい場合(BVAL、BLK パラメータ)、データは転送されません。

ANY ポインタ(ソースまたは宛先)がデータタイプ BOOL の場合、これを絶対アドレス指定する必要があり、かつ指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行されません。

宛先領域が STRING データタイプの場合、この命令は管理情報を含む文字列全体を書き込みます。

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
BVAL	Input	VARIANT	I、Q、M、L、P	その内容に内容が BLK パラメータの宛先領域を埋めるのに使用されるメモリ領域(ソース領域)の指定。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
BLK	Output [1)]	VARIANT	I、Q、M、L、P	ソース領域の内容で埋められるメモリ領域の指定。
1) データがタグに流れるため、BLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

RET_VAL パラメータ

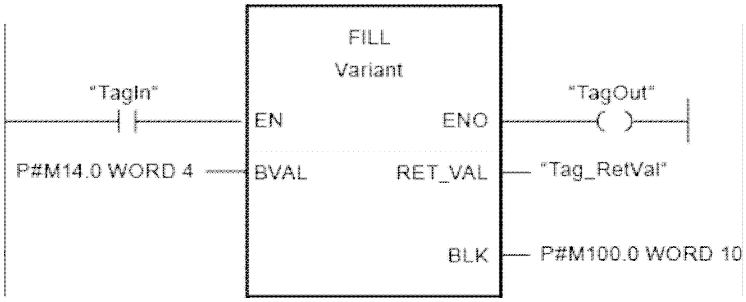
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみに存在します。

8152	WSTRING、WCHAR および BOOL データタイプは BVAL パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは BLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、ソース領域 MW14 ~ MW20 をコピーし、BVAL パラメータのメモリ領域に含まれる 4 ワードの内容で宛先領域 MW100 ~ MW118 を埋めます。

変換操作

この章には下記に関する情報が記載されています：

CONVERT: 値の変換

説明

「データの変換」命令は、IN パラメータの内容を読み出し、命令ボックスで選択されたデータタイプに従って変換します。変換された値は、出力 OUT で提供されます。

実行可能な変換に関する情報は、「関連項目」の「明示的な変換」のセクションを参照してください。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

ビット列の変換オプション

BYTE および WORD ビット列は、命令ボックスで選択できません。 ただし、入力オペランドと出力オペランドの長さが一致する場合は、命令のパラメータでデータタイプ DWORD または LWORD のオペランドを指定できます。 次にオペランドは、入力パラメータまたは出力パラメータのデータタイプに応じてビット列のデータタイプによって解釈され、暗黙的に変換されます。 たとえば、データタイプ DWORD は DINT/UDINT として解釈され、LWORD は LINT/ULINT として解釈されます。 これらの変換オプションは、[IEC チェック]が有効な場合にも使用可能です。

注記
S7-1500 シリーズの CPU の場合: データタイプ DWORD および LWORD は、データタイプ REAL または LREAL との間でのみ変換できます。

パラメータ

次の表に、「値の変換」命令のパラメータを示します。

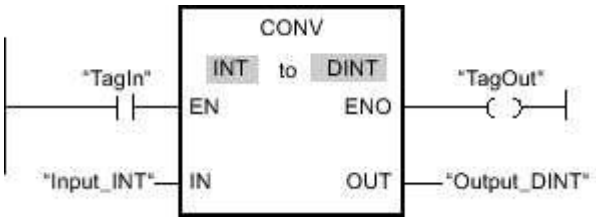
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数、浮動小数点数、CHAR、WCHAR、BCD16、BCD32	I、Q、M、D、L、P、または定数	変換する値
OUT	Output	ビット列、整数、浮動小数点数、CHAR、WCHAR、BCD16、BCD32	I、Q、M、D、L、P	変換の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

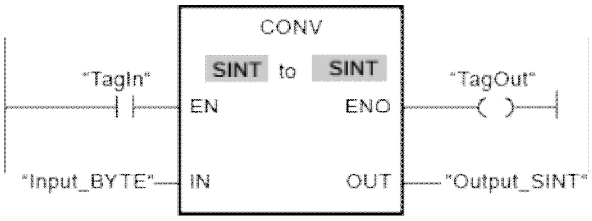
有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

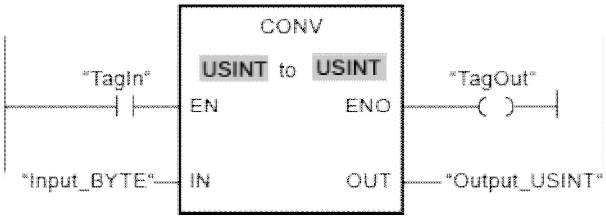
次の例は、整数(16 ビット)から別の整数(32 ビット)への変換を示します。



次の例は、 バイト(8 ビット)から整数 SINT (8 ビット)への変換を示します。



次の例は、バイト(8 ビット)から符号なし整数 USINT (8 ビット)への変換を示します。



これらの変換は、2 つのオペランドの長さが同じであるため可能です。

ROUND: 数値の四捨五入

説明

「数値の四捨五入」命令を使用して、入力 IN の値を近似の整数に四捨五入することができます。 この命令は、入力 IN の値を浮動小数点数として解釈し、データタイプ DINT の整数に変換します。 入力値が偶数と奇数のちょうど間にある場合、偶数が選択されます。 命令の結果は、OUT 出力に送信され、そこから照会できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「数値の四捨五入」命令のパラメータを示します。

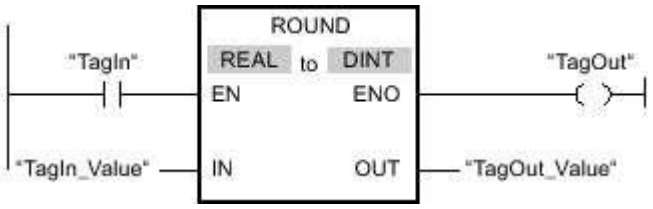
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	四捨五入される入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	丸めの結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	2	-2

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」の浮動小数点数は、最も近い偶数の整数に四捨五入され、出力「TagOut_Value」に送信されます。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

CEIL: 浮動小数点数から次に大きい整数を生成

説明

「浮動小数点数から次に大きい整数を生成」命令を使用して、入力 IN の値を次に大きい整数に切り上げることができます。 この命令は、IN 入力の値を浮動小数点数として解釈し、次に大きい整数に変換します。 命令の結果は、OUT 出力に送信され、そこから照会できます。 出力値は、入力値以上になることがあります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「浮動小数点数から次に大きい整数を生成」命令のパラメータを示します。

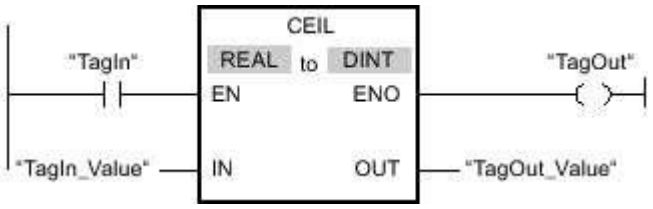
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	次に大きい整数の場合の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	0.50000000	-0.50000000
OUT	TagOut_Value	1	0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。「TagIn_Value」入力の浮動小数点数が、次に大きい整数に四捨五入され、「TagOut_Value」出力に送信されます。 この命令が エラーなしで実行された場合、「TagOut」出力がセットされます。

FLOOR: 浮動小数点数から次に小さい整数を生成

説明

「浮動小数点数から次に小さい整数を生成」命令を使用して、入力 IN の値を次に小さい整数に切り下げることができます。 この命令は、入力 IN の値を浮動小数点数として解釈し、浮動小数点数の値を次に小さい整数に変換します。 命令の結果は、OUT 出力に送信され、そこから照会できます。 出力値は、入力値以下になります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「浮動小数点数から次に小さい整数を生成」命令のパラメータを示します。

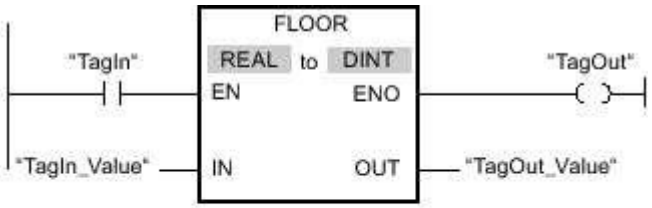
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	次に小さい整数の場合の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	0.50000000	-0.50000000

OUT	TagOut_Value	0	-1

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」の浮動小数点数が、次に小さい整数に四捨五入され、出力「TagOut_Value」に送信されます。 この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

TRUNC: 数値の切り捨て

説明

「数値を切り捨てる」命令を使用して、IN 入力の値から整数を形成することができます。 IN 入力の値は、浮動小数点数として解釈されます。 この命令は、浮動小数点数の整数部のみを選択し、小数位なしで OUT 出力に送信します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「数値を切り捨てる」命令のパラメータを示します。

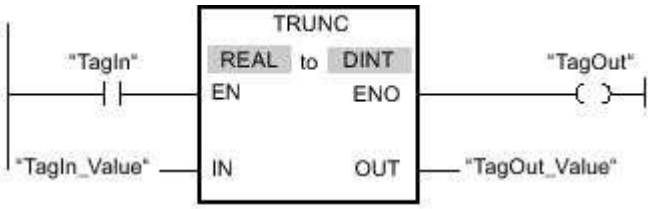
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、または定数	入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L	入力値の整数部

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	1.50000000	-1.50000000

OUT	TagOut_Value	1	-1

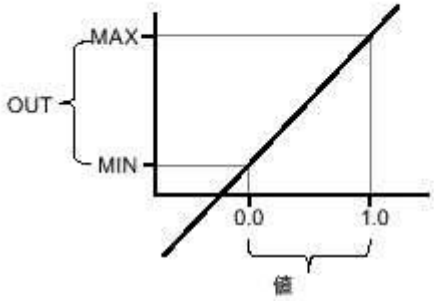
「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。「TagIn_Value」入力の浮動小数点数の整数部が、整数に変換され、「TagOut_Value」出力に送信されます。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

SCALE_X: スケール

説明

「スケール」命令を使用して、VALUE 入力の値を指定した値の範囲にマッピングしてスケーリングすることができます。 「スケール」命令が実行されると、VALUE 入力の浮動小数点値が、MIN および MAX パラメータによって定義された値の範囲にスケーリングされます。 スケーリングの結果は整数になり、OUT 出力に保存されます。

次の図に、値をどのようにスケーリングできるかを示します。



「スケール」命令は、以下の等式で機能します。

OUT = [VALUE * (MAX – MIN)] + MIN

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- MIN 入力の値が MAX 入力の値以上であること。
- 指定した浮動小数点数の値が、IEEE-754 に従い正規化された数の範囲外であること。
- オーバーフローが発生していること。
- VALUE 入力の値が NaN (Not a Number = 無効な算術演算の結果)であること。

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
MIN	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE	Input	浮動小数点数	I、Q、M、D、L、または定数	スケーリングされる値 定数を入力する場合は、それを宣言する必要があります。
MAX	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の上限値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L	スケーリングの結果

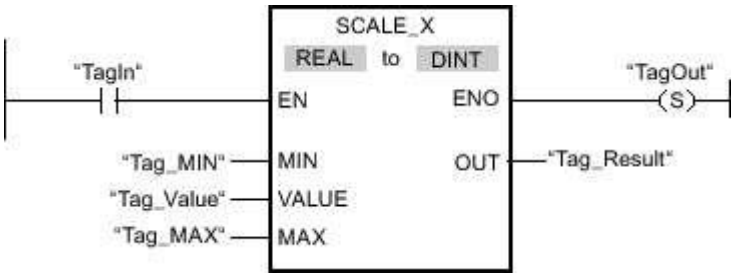
命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

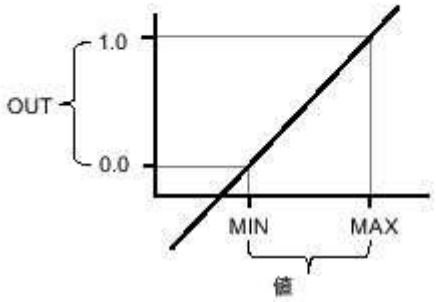
「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 「Tag_Value」入力の値が、「Tag_MIN」および「Tag_MAX」入力の値によって定義された値の範囲にスケーリングされます。結果が「Tag_Result」出力に保存されます。 この命令がエラーなしで実行された場合、ENO許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

NORM_X: 正規化

説明

「正規化」命令を使用して、VALUE 入力のタグの値をリニアスケーリングにマッピングして正規化することができます。 MIN および MAX パラメータを使用し、スケーリングに適用する値の範囲の限界を定義できます。 OUT 出力の結果が計算され、正規化される値のこの値の範囲内の位置に応じて、浮動小数点数として保存されます。 正規化される値が MIN 入力の値に等しい場合、OUT 出力の値は「0.0」になります。 正規化する値が入力 MAX の値と等しい場合、出力 OUT が値「1.0」を返します。

次の図に、値が正規化される例を示します。



「正規化」命令は、以下の等式で機能します。

OUT = (VALUE – MIN) / (MAX – MIN)

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- MIN 入力の値が MAX 入力の値以上であること。
- 指定した浮動小数点数の値が、IEEE-754 に従い正規化された数の範囲外であること。
- VALUE 入力の値が NaN(無効な算術演算の結果)であること。

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

パラメータ

次の表に、「正規化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
MIN [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	正規化する値
MAX [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の上限値
OUT	Output	浮動小数点数	I、Q、M、D、L	正規化の結果
[1)] これらの 3 つのパラメータで定数を使用する場合は、そのいずれかを宣言するのみで済みます。				

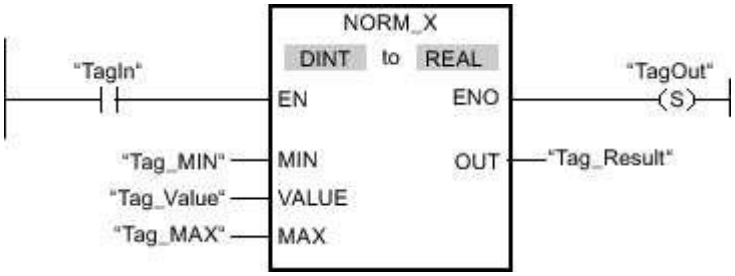
命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 「Tag_Value」入力の値が、「Tag_MIN」および「Tag_MAX」入力の値によって定義された値の範囲にマッピングされます。 「Tag_Value」入力のタグ値が、定義された値の範囲に正規化されます。 結果が、浮動小数点数として「Tag_Result」出力に保存されます。 この命令が エラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

レガシー

この章には下記に関する情報が記載されています：

- SCALE:スケール (S7-1500)
- UNSCALE:スケール解除 (S7-1500)

SCALE:スケール

説明

「スケール」命令は、IN パラメータの整数を下限値と上限値間の物理単位でスケーリングが可能な浮動小数点数に変換します。 LO_LIM および HI_LIM パラメータを使用して、入力値をスケーリングする範囲の下限値と上限値を指定することができます。 命令の結果は、OUT パラメータに出力されます。

「スケール」命令は、以下の等式で機能します。

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

定数「K1」および「K2」の値は、BIPOLAR パラメータの信号状態で決定されます。 BIPOLAR パラメータでは、以下の信号状態が可能です。

- 信号状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の値範囲にあると想定されます。この場合、定数「K1」の値は「-27648.0」となり、定数「K2」の値は「+27648.0」となります。
- 信号状態「0」: IN パラメータの値はユニポーラであり、0～27648 の範囲の値であると想定されます。この場合、定数「K1」の値は「0.0」となり、定数「K2」の値は「+27648.0」となります。

IN パラメータの値が定数値「K2」よりも大きい場合、命令の結果が上限値(HI_LIM)にセットされ、エラーが出力されます。

IN パラメータの値が定数値「K1」未満の場合、命令の結果が下限値(LO_LIM)にセットされ、エラーが出力されます。

示された下限値が上限値よりも大きい(LO_LIM > HI_LIM)場合、結果は入力値に反比例してスケーリングされます。

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	INT	I、Q、M、D、L、P、または定数	スケーリングする入力値。
HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。

				1: バイポーラ 0: ユニポーラ
OUT	Output	REAL	I、Q、M、D、L、P	命令の結果
RET_VAL	Output	WORD	I、Q、M、D、L、P	エラー情報

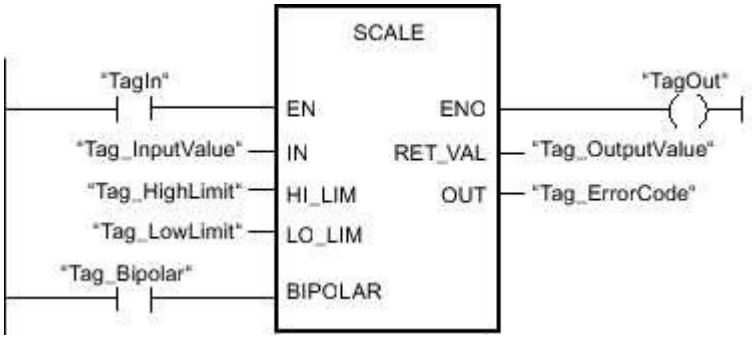
パラメータ RET_VAL

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が定数値「K2」を超えているか、または定数値「K1」未満になっています。
一般エラー情報	関連項目 "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

UNSCALE:スケール解除

説明

「スケール解除」命令を使用して、IN パラメータの浮動小数点数を下限値と上限値間の物理単位にスケーリング解除し、それらを整数に変換します。 LO_LIM および HI_LIM パラメータを使用して、入力値をスケーリング解除する範囲の下限値と上限値を指定することができます。 命令の結果は、OUT パラメータに出力されます。

「スケール解除」命令は、以下の等式で機能します。

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

定数「K1」および「K2」の値は、BIPOLAR パラメータの信号状態で決定されます。 BIPOLAR パラメータでは、以下の信号状態が可能です。

- 信号状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の値範囲にあると想定されます。この場合、定数「K1」の値は「-27648.0」となり、定数「K2」の値は「+27648.0」となります。
- 信号状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の値範囲にあると想定されます。この場合、定数「K1」の値は「0.0」となり、定数「K2」の値は「+27648.0」となります。

IN パラメータの値が定数値「HI_LIM」よりも大きい場合、命令の結果が定数値(K2)にセットされ、エラーが出力されます。

IN パラメータの値が下限値の定数値「LO_LIM」未満の場合、命令の結果が定数値(K1)にセットされ、エラーが出力されます。

パラメータ

次の表に、「スケール解除」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Input	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	REAL	I、Q、M、D、L、P、または定数	整数値にスケーリング解除する入力値。
HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ

				0: ユニポーラ
OUT	Output	INT	I、Q、M、D、L、P	命令の結果
RET_VAL	Output	WORD	I、Q、M、D、L、P	エラー情報

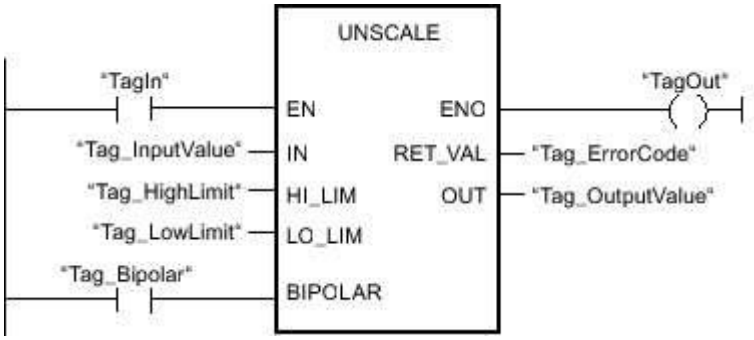
パラメータ RET_VAL

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が上限値(HI_LIM)を超えているか、または下限値(LO_LIM)未満になっています。
一般エラー情報	関連項目 "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

プログラム制御演算

この章には下記に関する情報が記載されています：

---(JMP):RLO = 1 ならばジャンプ

説明

「RLO = 1 ならばジャンプ」命令を使用して、プログラムのリニア実行を中断し、別のネットワークで再開できます。 宛先ネットワークは、ジャンプラベル(LABEL)で識別する必要があります。 このジャンプラベルの名前は、命令の上のプレースホルダに指定されます。

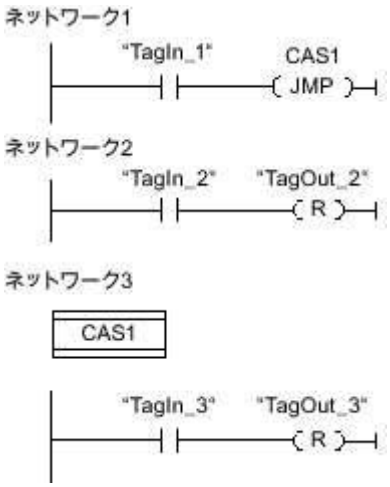
指定したジャンプラベルは、命令が実行されるブロックと同じブロックにあることが必要です。 指定する名前はブロック内で固有でなければなりません。 1 つのネットワークで使用できるジャンピングコイルは 1 つのみです。

命令の入力でのボックス入力の論理演算の結果(RLO)が「1」の場合、指定されたジャンプラベルで識別されたネットワークへのジャンプが実行されます。 ジャンプ方向は、大きいまたは小さいネットワーク番号の方向になります。

命令の入力での条件が満たされない場合(RLO = 0)、プログラムの実行は次のネットワークで続行されます。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」の信号状態が「1」の場合、「RLO = 1 ならばジャンプ」命令が実行されます。 プログラムの線形実行が中断され、ジャンプラベル CAS1 で識別されるネットワーク 3 で続行されます。 "TagIn_3"入力の信号状態が「1」の場合、"TagOut_3"出力が設定されます。

---(JMPN):RLO = 0 ならばジャンプ

説明

「RLO = 0 ならばジャンプ」命令を使用し、命令の入力での論理演算の結果が「0」の場合、プログラムの線形実行に割り込み、それを別のネットワークで再開することができます。 宛先ネットワークは、ジャンプラベル(LABEL)で識別する必要があります。 このジャンプラベルの名前は、命令の上のプレースホルダに指定されます。

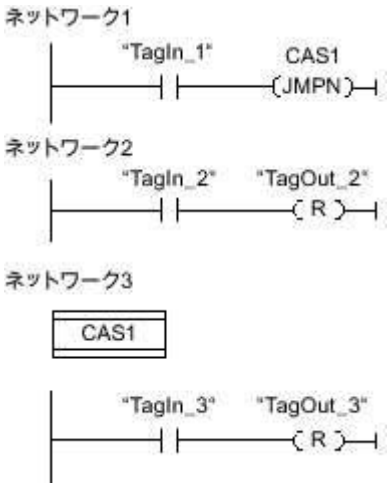
指定したジャンプラベルは、命令が実行されるブロックと同じブロックにあることが必要です。 指定する名前はブロック内で固有でなければなりません。 1 つのネットワークで使用できるジャンピングコイルは 1 つのみです。

命令の入力でのボックス入力の論理演算の結果(RLO)が「0」の場合、指定されたジャンプラベルで識別されたネットワークへのジャンプが実行されます。 ジャンプ方向は、大きいまたは小さいネットワーク番号の方向になります。

命令の入力での論理演算の結果が「1」の場合、プログラムの実行は、次のネットワークで継続します。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」の信号状態が「0」の場合、「RLO = 0 ならばジャンプ」命令が実行されます。 プログラムの線形実行が中断され、ジャンプラベル CAS1 で識別されるネットワーク 3 で続行されます。 "TagIn_3"入力の信号状態が「1」の場合、"TagOut_3"出力が設定されます。

LABEL: ジャンプラベル

説明

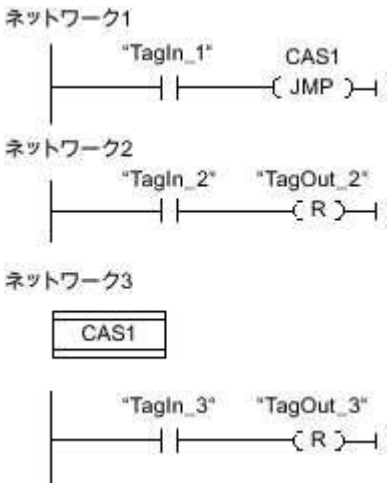
ジャンプラベルを使用して、ジャンプが実行されるときにプログラムの実行が再開される宛先ネットワークを識別することができます。

ジャンプラベルとジャンプラベルを指定する命令は、同じブロックに置く必要があります。 ジャンプラベルの名前は、ブロック内で 1 回のみ割り当てることができます。 CPU S7-1200 を使用している場合は最大で 32 のジャンプラベル、CPU S7-1500 を使用している場合は最大で 256 のジャンプラベルを宣言できます。

ネットワークに置くことのできるジャンプラベルは 1 つのみです。 各ジャンプラベルから複数の位置にジャンプできます。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」の信号状態が「1」の場合、「RLO = 1 ならばジャンプ」命令が実行されます。プログラムの線形実行が中断され、ジャンプラベル CAS1 で識別されるネットワーク 3 で続行されます。"TagIn_3"入力の信号状態が「1」の場合、"TagOut_3"出力が設定されます。

JMP_LIST: ジャンプリストの定義

説明

「ジャンプリストの定義」を使って、複数の条件ジャンプを定義できます。また、K パラメータの値に基づいて、指定したネットワーク内でプログラム実行を継続できます。

ジャンプはジャンプラベル(LABEL)を使用して定義します。ジャンプラベルは、命令ボックスの出力で指定します。 出力の数は、命令ボックスで拡張できます。CPU S7-1200 を使用している場合は最大で 32 の出力、CPU S7-1500 を使用している場合は最大で 99 の出力を宣言できます。

出力の番号付けは、値「0」から開始し、新しい出力ごとに昇順で続きます。 命令の出力では、ジャンプラベルのみを指定できます。 命令またはオペランドを指定することはできません。

K パラメータの値は、出力の番号、およびプログラム実行が再開されるジャンプラベルを指定します。 K パラメータの値が使用可能な出力の数よりも大きい場合、ブロックの次のネットワークでプログラム実行が再開されます。

「ジャンプリストの定義」命令は、EN イネーブル入力の信号状態が「1」の場合にのみ実行されます。

パラメータ

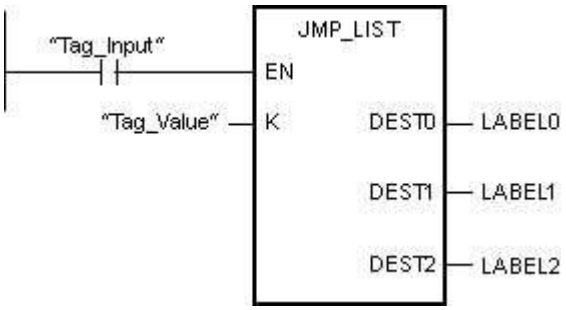
次の表に、「ジャンプリストの定義」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
K	Input	UINT	I、Q、M、D、L、または定数	出力の番号、ひいては実行されるジャンプを指定します。
DEST0	-	-	-	最初のジャンプラベル
DEST1	-	-	-	2 番目のジャンプラベル
DESTn	-	-	-	オプションのジャンプラベル

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド/ジャンプラベル	値
K	Tag_Value	1
Dest0	LABEL0	ジャンプラベル「LABEL0」で識別されるネットワークにジャンプします。
Dest1	LABEL1	ジャンプラベル「LABEL1」で識別されるネットワークにジャンプします。
Dest2	LABEL2	ジャンプラベル「LABEL2」で識別されるネットワークにジャンプします。

オペランド「Tag_Input」の信号状態が「1」の場合、「ジャンプリスト定義」命令が実行されます。ジャンプラベル「LABEL1」で識別されるネットワークのオペランド「Tag_Value」の値に従って、プログラムの実行が再開されます。

SWITCH:ディストリビュータにジャンプ

説明

「ディストリビュータジャンプ」命令を使用して複数のプログラムジャンプを定義し、1 つ以上の比較命令の結果に応じて実行することができます。

K パラメータで比較される値を指定します。 この値は、さまざまな入力によって提供される値と比較されます。 個々の入力に対して、比較方法を選択できます。 さまざまな比較命令の可用性は、命令のデータタイプに応じて異なります。

次の表に、選択されたデータタイプに応じて使用可能な比較命令を示します。

データタイプ		命令	構文
S7-1200	S7-1500		
ビット列	ビット列	等価	==
		不等価	<>
整数、浮動小数点数、TIME、DATE、TOD	整数、浮動小数点数、TIME、LTIME、DATE、TOD、LTOD、LDT	等価	==
		不等価	<>
		以上	>=
		以下	<=
		超過	>
		未満	<

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。 比較命令を選択し、命令のデータタイプがまだ定義されていない場合、「<???>」ドロップダウンリストによって、選択された比較命令に対して許可されているデータタイプのみが提供されます。

命令の実行は、最初の比較で開始し、比較条件が満たされるまで実行します。 比較条件が満たされている場合、その後の比較条件は考慮されません。 指定されたどの比較条件も満たされない場合、ELSE 出力のジャンプが実行されます。 ELSE 出力でプログラムジャンプが定義されていない場合、次のネットワークでプログラムの実行が続行されます。

出力の数は、命令ボックスで拡張できます。 出力の番号付けは、値「0」から開始し、新しい出力ごとに昇順で続きます。命令の出力でジャンプラベル(LABEL)を指定します。 命令の出力で、命令またはオペランドを指定することはできません。

追加出力のそれぞれに対して、入力が自動的に挿入されます。 対応する入力の比較条件が満たされている場合、出力でプログラミングされたジャンプが実行されます。

パラメータ

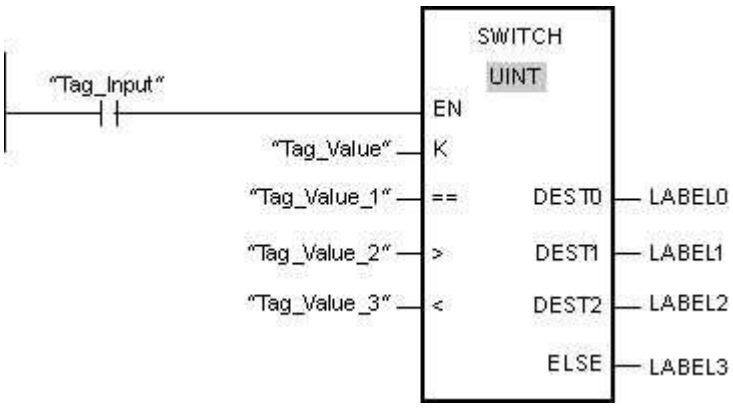
次の表に、「ディストリビュータジャンプ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	イネーブル入力
K	Input	UINT	UINT	I、Q、M、D、L または定数	比較する値を指定します。
<比較値>	Input	ビット列、整数、浮動小数点数、TIME、DATE、TOD	ビット列、整数、浮動小数点数、TIME、LTIME、DATE、TOD、LTOD、LDT	I、Q、M、D、L または定数	K パラメータの値が比較される入力値。
DEST0	-	-	-	-	最初のジャンプラベル
DEST1	-	-	-	-	2 番目のジャンプラベル
DEST(n)	-	-	-	-	オプションのジャンプラベル(n = 2 ~ 99)
ELSE	-	-	-	-	どの比較条件も満たされない場合、実行されるプログラムジャンプ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド/ジャンプラベル	値
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19

Dest 0	LABEL0	K パラメータの値が 20 に等しい場合、ジャンプラベル「LABEL0」にジャンプします。
Dest 1	LABEL1	K パラメータの値が 21 よりも大きい場合、ジャンプラベル「LABEL1」にジャンプします。
Dest 2	LABEL2	K パラメータの値が 19 未満の場合、ジャンプラベル「LABEL2」にジャンプします。
ELSE	LABEL 3	どの比較条件も満たされない場合、ジャンプラベル「LABEL3」にジャンプします。

オペランド「Tag_Input」の信号状態が「1」になった場合、「ディストリビュータジャンプ」命令を実行。ジャンプラベル「LABEL1」で識別されるネットワークでプログラムの実行を継続。

--(RET): 戻り値

説明

「戻り値」命令を使って、ブロックの実行を停止します。 結果は 3 タイプであり、ブロック処理を完了できます。

- 「戻り値」命令の呼び出しなし

  最後のネットワークの実行の後、ブロックが終了されます。 ファンクション呼び出しの ENO の信号状態が「1」にセットされます。

- 先に論理演算を行う「戻り値」命令の呼び出し(例を参照)

  左コネクタの信号状態が「1」の場合、ブロックが終了されます。 ファンクション呼び出しの ENO がオペランドに対応します。

- 先に論理演算を行わない「戻り値」命令の呼び出し

  ブロックが終了されます。 ファンクション呼び出しの ENO がオペランドに対応します。

> **注記**
> 1 つのネットワークで使用できるジャンピングコイルは 1 つのみです(「戻り値」、「RLO=1 ならばジャンプ」、「RLO=0 ならばジャンプ」)。

「戻り値」命令の入力で論理演算(RLO)の結果が「1」の場合、現在呼び出されているブロックでプログラムの実行が終了され、呼び出し元ブロックの呼び出しファンクションの後で再開されます(たとえば、呼び出し元 OB で)。 呼び出しファンクションのステータス(ENO)は、命令のパラメータで決定されます。 以下の値が想定されます。

- [RLO]
- TRUE/FALSE
- <オペランド>

パラメータ値を設定するには、命令をダブルクリックし、ドロップダウンリストで対応する値を選択します。

次の表に、呼び出したブロック内のネットワークで「戻り値」命令がプログラミングされた場合の呼び出しファンクションのステータスを示します。

[RLO]	パラメータ値	呼び出しファンクションの ENO
1	[RLO]	1
	TRUE	1
	FALSE	0
	<オペランド>	<オペランド>
0	[RLO]	プログラムの実行は、呼び出されたブロックの次のネットワークで続行します。
	TRUE	
	FALSE	
	<オペランド>	

OB が完了された場合、優先度クラスシステムによって、別のブロックが選択されて開始されるか、実行されます。

- プログラムサイクル OB が完了された場合、再開されます。
- 別のブロックに割り込んだ OB (たとえば OB)が完了された場合、割り込まれたブロック(たとえばプログラムサイクル OB)が実行されます。

パラメータ

次の表に、「戻り値」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
				RLO = 1 の場合の呼び出し元ファンクションのステータス:
[RLO]	-	-		RLO の信号状態にセットされます。
TRUE	-	-		1
FALSE	-	-		0
<オペランド>	Input	BOOL	I、Q、M、D、L	指定したオペランドの信号状態

例

次の例で、命令がどのように機能するかを示します。

オペランド「TagIn」の信号状態が「1」の場合、「戻り値」命令が実行されます。 呼び出されたブロックでプログラム実行は終了され、呼び出し元のブロックで続行します。 呼び出しファンクションの ENO イネーブル出力は、信号状態「0」にリセットされます。

ランタイム制御

この章には下記に関する情報が記載されています：

- ENDIS_PW: パスワードの適正化の制限および有効化 (S7-1200, S7-1500)
- RE_TRIGR: サイクルタイムモニタのリスタート (S7-1200, S7-1500)
- STP: プログラム終了 (S7-1200, S7-1500)
- GET_ERROR: ローカルでエラーを取得 (S7-1200, S7-1500)
- GET_ERR_ID: ローカルでエラー ID を取得 (S7-1200, S7-1500)
- INIT_RD: すべての維持データを初期化 (S7-1500)
- WAIT:遅延時間の設定 (S7-1500)
- ランタイム:プログラムランタイムの測定 (S7-1200, S7-1500)

ENDIS_PW: パスワードの適正化の制限および有効化

説明

「パスワード正当性の制限および有効化」命令を使用して、設定されたパスワードを CPU に対して正当化できるかどうかを指定します。 正しいパスワードがわかっている場合でも、正規の接続を防止することができます。

コマンド呼び出し時に REQ パラメータの信号状態が「0」の場合、現在設定されている状態のみが、出力パラメータに表示されます。 入力パラメータに変更を加えても、変更は出力パラメータに転送されません。

命令呼び出し時に REQ パラメータの信号状態が「1」の場合、信号状態は、入力パラメータ(F_PWD、FULL_PWD、R_PWD、HMI_PWD)から取得されます。FALSE は、パスワードごとの正当化が許可されていないことを意味します。TRUE は、パスワードを使用できることを意味します。

パスワードの有効化または無効化を個別に許可または禁止することができます。 たとえば、フェールセーフパスワードを除く、すべてのパスワードを禁止することができます。 したがって、アクセスオプションを小さいユーザーグループに制限することができます。 出力パラメータ(F_PWD_ON、FULL_PWD_ON、R_PWD_ON、HMI_PWD_ON)は、REQ パラメータに関係なく、常にパスワード使用の現在のステータスを示します。

設定済みでないパスワードの場合、入力の信号状態が TRUE でなければならず、出力に TRUE 信号状態を返します。 フェールセーフパスワードは、F-CPU の場合のみ設定できるため、標準 CPU では、常に TRUE 信号状態と相互接続する必要があります。 命令がエラーを返す場合、呼び出しは効果がないままです。すなわち、直前のロックが有効なままです。

無効なパスワードは、以下の条件で再度有効にすることができます。

- CPU が工場出荷時設定にリセットされること。
- S7-1500 CPU のフロントパネルが、パスワードを再度有効にすることができる適切なメニューに移動することができるダイアログをサポートしていること。
- 「パスワードの適正化の制限および有効化」命令を呼び出す場合、目的のパスワードの入力パラメータの信号状態が「1」であること。
- 動作モードスイッチを STOP にセットします。 このスイッチの設定を RUN に戻すと直ちに、パスワードの適正化に対する制限が再確立されます。
- S7-1200 CPU への空きメモリカード(転送カードまたはプログラムカード)の挿入。
- S7-1200 CPU では、電源オフから電源オンへの移行によって、保護が無効になります。 この場合、プログラム(たとえば、スタートアップ OB)で、再度、「パスワード正当性の制限および有効化」命令を呼び出す必要があります。

注記
「パスワード正当性の制限および有効化」命令は、HMI パスワードが有効にされていない場合、HMI システムへのアクセスをブロックします。

注記
既存の適正化された接続はアクセス権を保持し、「パスワードの適正化の制限および有効化」命令を使用してそれらを制限することはできません。

S7-1500 CPU での偶発的なロックアウトの防止

S7-1500 CPU のフロントパネルで同じ設定を行なうことができ、CPU は最新の設定を保存します。

偶発的なロックアウトを防止するには、S7-1500 CPU でモードセレクタを STOP に設定することによって保護を無効にすることができます。 モードセレクタを RUN に設定することによって、「パスワ

ードの適正化の制限および有効化」命令を再度呼び出したり、フロントパネルで追加の操作を行う必要なしに、保護が自動的に再度有効にされます。

S7-1200 CPU での偶発的なロックアウトの防止

S7-1200 CPU には動作モードスイッチが存在しないため、電源オフから電源オンへの移行時の保護は無効です。 このため、ユーザープログラム内の特定のプログラムシーケンスを使用して偶発的なロックアウトを防止することをお奨めします。

これを行うには、サイクリック割り込み OB、または、メイン OB (OB 1)のタイマのどちらかを使用して、時間制御をプログラミングします。 これは、電源オフから電源オンへの移行後に、遅延なく、該当 OB (OB 1 または OB 35)で、「パスワード正当性の制限および有効化」命令を再度呼び出すオプションを提供します。 アプリケーションが動作していない時間ウィンドウの表示時間を可能なかぎり短くし、パスワードの正当性に制限が存在しないようにするには、スタートアップ OB (OB 100)でこの命令を呼び出します。 この手順は、未許可のアクセスに対する可能な最大限の保護を提供します。

意図しないロックアウトが発生した場合は、スタートアップ OB での呼び出しをスキップし(たとえば、入力パラメータを照会することによって)、ロックが再び有効になる前に、CPU へ接続確立する時間(たとえば、10 秒～1 分)を設定することができます。

ユーザープログラムコードにタイマを設定していない場合にロックアウトされたときは、空の転送カードまたはプログラムカードを CPU に挿入します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 この後、ユーザープログラムを再び STEP 7 から CPU にロードする必要があります。

S7-1200 CPU でパスワードが失われた場合の手順

パスワード保護された S7-1200 CPU のパスワードを紛失した場合は、空の転送カードまたはプログラムカードを使用して、パスワード保護されたプログラムを削除します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 その後、新しいユーザープログラムを STEP 7 Basic から CPU にロードすることができます。

警告

空の転送カードの挿入

ランタイム中に CPU にトランスファーカードを挿入すると、CPU は STOP モードに切り替わります。 動作ステータスが不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期せぬ動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

転送カードを取り出した後、転送カードの内容は内部ロードメモリに存在します。 この時点に、カードにプログラムが含まれていないことを確認してください。

警告

空のプログラムカードの挿入

ランタイム中に CPU にプログラムカードを挿入すると、CPU は STOP モードに移行します。動作ステータスが不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。これは、オートメーションシステムの予期せぬ動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

プログラムカードが空であることを確認してください。 内部ロードメモリが空のプログラムカードにコピーされます。 以前は空であったプログラムカードを取り出した後、内部ロードメモリは空です。

CPU を RUN モードに切り替える前に、転送カードまたはプログラムカードを取り出す必要があります。

動作モードへのパスワードの使用の影響

次の表に、「パスワードの適正化の制限および有効化」命令によるパスワードの使用が動作モードと対応するユーザーの操作に与える影響を示します。

操作	命令によるパスワードの保護
その後の基本状態 • STOP への動作モードの切り替え • メモリの手動リセット(PG、切り替え、MC (モーションコントロール)の変更) • 工場出荷時設定へのリセット	無効 (制限なし)
電源オン後の基本状態	• S7-1200-CPU: ロックは無効で、プログラム(たとえば、スタートアップ OB)で再びこの命令を呼び出す必要があります。 • S7-1500 CPU: 有効(電源オフの前にロックが有効になった場合). パスワードの不許可のオプションは保持型です。
動作モードの移行 RUN/STARTUP/HOLD -> STOP (命令、エラーまたは通信の終了によって)または STOP -> STARTUP/RUN/HOLD	有効 パスワードはまだ使用できません。

パラメータ

次の表に、「パスワードの適正化の制限および有効化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ パラメータの信号状態が「0」の場合は、現在設定されているパスワードの信号状態が照会されます。
F_PWD	Input	BOOL	I、Q、M、D、L、または定数	フェールセーフなどの読み取り/書き込みアクセス • F_PWD = "0": パスワードを許可しない • F_PWD = "1": パスワードを許可する
FULL_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取り/書き込みアクセス • FULL_PWD = "0": パスワードを許可しない • FULL_PWD = "1": パスワードを許可する
R_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取りアクセス • R_PWD = "0": パスワードを許可しない • R_PWD = "1": パスワードを許可する
HMI_PWD	Input	BOOL	I、Q、M、D、L、または定数	HMI アクセス

				• HMI_PWD = "0": パスワードを許可しない • HMI_PWD = "1": パスワードを許可する
F_PWD_ON	Output	BOOL	I、Q、M、D、L	フェールセーフなどの読み取り/書き込みアクセス • F_PWD_ON = "0": パスワードを許可しない • F_PWD_ON = "1": パスワードを許可する
FULL_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取り/書き込みアクセスステータス • FULL_PWD_ON = "0": パスワードを許可しない • FULL_PWD_ON = "1": パスワードを許可する
R_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取りアクセスステータス • R_PWD_ON = "0": パスワードを許可しない • R_PWD_ON = "1": パスワードを許可する
HMI_PWD_ON	Output	BOOL	I、Q、M、D、L	HMI アクセスステータス • HMI_PWD_ON = "0": パスワードを許可しない • HMI_PWD_ON = "1": パスワードを許可する
RET_VAL	Output	WORD	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	**説明**
0000	エラーは発生していません。
8090	「パスワードの適正化の制限および有効化」命令はサポートされていません
80D0	フェールセーフのパスワードが未設定です。 標準 CPU では、信号状態は TRUE であることが必要です。.
80D1	読み取り/書き込みアクセスが未設定です
80D2	読み取りアクセスが未設定です
80D3	HMI アクセスが未設定です
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

RE_TRIGR: サイクルタイムモニタのリスタート

説明

「サイクルタイムモニタのリスタート」命令を使用して、CPU のサイクルタイムモニタを再起動することができます。 最大サイクルタイムが、CPU の設定で設定した期間中に最初からやり直されます。

「サイクルタイムモニタのリスタート」命令は、すべてのブロック内で優先度に関わらず呼び出し可能です。

ハードウェア割り込み、診断エラー割り込み、周期割り込みなど、さらに優先度の高いブロックで呼び出された場合、この命令は実行されず、ENO 許可出力がシグナル状態「0」にセットされます。

「サイクルタイムモニタのリスタート」命令は、呼び出しの数に関係なく期間内に完全に実行されます (最大プログラムサイクルの 10 倍)。 時間が期限切れになると、プログラムサイクルは延長できなくなります。

パラメータ

「サイクルタイムモニタのリスタート」命令には、パラメータはありません。

STP: プログラム終了

説明

「プログラムの終了」命令を使用し、CPU を STOP モードに設定してプログラムの実行を終了します。 RUN から STOP への変更の影響は CPU の構成によって異なります。

命令の入力での論理演算の結果(RLO)が「1」の場合、CPU は「STOP」モードに切り替わり、プログラムの実行が終了されます。 命令の出力の信号状態は評価されません。

命令の入力の RLO が「0」の場合、命令は実行されません。

パラメータ

「プログラムの終了」命令にはパラメータはありません。

GET_ERROR: ローカルでエラーを取得

説明

「ローカルでエラーを取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 システムがブロックの処理中にエラーを報告した場合、命令の最後の実行の後、このエラーがブロックの実行中に発生した最初の該当エラーの場合、詳細情報が、ERROR 出力のオペランドに格納されます。

ERROR 出力では、「ErrorStruct」システムデータタイプのオペランドのみを指定できます。 「ErrorStruct」システムデータタイプは、エラー情報が保存されている正確な構造体を指定します。 追加の命令を使用することで、この構造体とプログラムが適切な応答をするかを評価できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーに関するエラー情報が出力されます。

注記
ERROR 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。
- イネーブル出力 ENO を照会する。

「ローカルでエラーを取得」命令のイネーブル出力 ENO は、イネーブル入力 EN が信号状態「1」を返し、エラー情報が存在する場合のみ設定されます。 これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラーを取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラーを取得」命令は、ブロック内のローカルエラー処理を可能にします。 「ローカルでエラーを取得」がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

パラメータ

次の表に、「ローカルでエラーを取得」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
ERROR	Output	ErrorStruct	D、L	エラー情報

データタイプ「ErrorStruct」

次の表に、「ErrorStruct」データタイプの構造体をリストします。

構造体コンポーネント	データタイプ	説明
ERROR_ID	WORD	エラー ID
FLAGS	BYTE	エラーがブロック呼び出し中に発生したのかどうかを示します。

			16#01: ブロック呼び出し中のエラー 16#00: ブロック呼び出し中にエラーなし
REACTION		BYTE	デフォルトの応答 0: 無視(書き込みエラー) 1: 代替値「0」で続行(読み取りエラー) 2: 命令をスキップ(システムエラー)
CODE_ADDRESS		CREF	ブロックのアドレスおよびタイプに関する情報
	BLOCK_TYPE	BYTE	エラーが発生したブロックのタイプ: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	プログラムブロックの番号
	OFFSET	UDINT	内部メモリへの参照
MODE		BYTE	オペランドのアドレスに関する情報
OPERAND_NUMBER		UINT	マシンコマンドのオペランド番号
POINTER_NUMBER_LOCATION		UINT	(A) 内部ポインタ
SLOT_NUMBER_SCOPE		UINT	(B) 内部メモリの記憶領域
DATA_ADDRESS		NREF	オペランドのアドレスに関する情報
	AREA	BYTE	(C) メモリ領域 L: 16#40～4E、86、87、8E、8F、C0～CE I:16#81 Q: 16#82 M: 16#83 DB: 16#84、85、8A、8B データタイプ DINT の直接編集可能なタグの範囲違反: 16#04
	DB_NUMBER	UINT	(D) データブロックの番号
	OFFSET	UDINT	(E) オペランドの相対アドレス

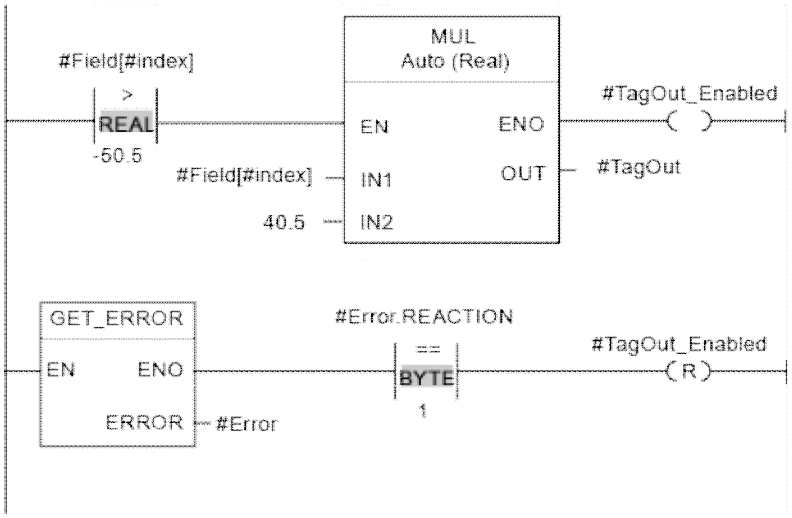
構造体コンポーネント「ERROR_ID」

次の表に、構造体コンポーネント「ERROR_ID」で出力可能な値をリストします。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー:無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー:データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、ファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、ファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー:DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
2576	9590	ローカルデータ配布のエラー
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー: 入力
2943	10563	書き込みエラー: 出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

例

次の例で、命令がどのように機能するかを示します。



「#Field[#index]」タグへのアクセス時にエラーが発生しました。 読み取り/アクセスエラーにも関わらず、「乗算」命令のイネーブル出力 ENO と「#TagOut_Enabled」オペランドの信号状態が「1」で、値「0.0」を使用して乗算が実行されます。 このエラーシナリオが発生した場合は、エラーを取得するために、「乗算」命令の後に「ローカルでエラーを取得」命令をプログラミングすることを推奨します。 「ローカルでエラーを取得」命令によって供給されるエラー情報は、比較命令「等しい」を使用して評価されます。 「#Error.REACTION」構造体コンポーネントの値が「1」の場合、読み取り/アクセスエラーが伴い、「#TagOut_Enabled」出力がリセットされます。

GET_ERR_ID: ローカルでエラー ID を取得

説明

「ローカルでエラー ID を取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 この命令の最後の実行の後に、システムがブロック処理中にブロック実行に関するエラーを報告すると、タグで発生した最初のエラーのエラー ID が ID 出力に格納されます。

ID 出力では、WORD データタイプのオペランドのみを指定できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーのエラー ID が出力されます。

注記
ID 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。
- イネーブル出力 ENO を照会する。

命令「ローカルでエラー ID を取得」のイネーブル出力 ENO は、イネーブル入力 EN が信号状態「1」を返し、エラー情報が存在する場合のみ設定されます。 これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラー ID を取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラー ID を取得」命令は、ブロック内のローカルエラー処理を可能にします。「ローカルでエラー ID を取得」命令がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

パラメータ

次の表に、「ローカルでエラー ID を取得」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
ID	Output	WORD	I、Q、M、D、L	エラー ID

ID パラメータ

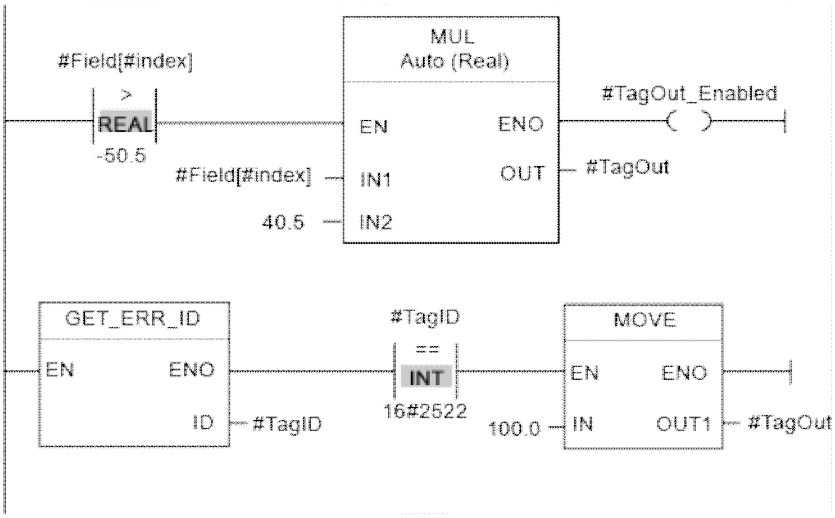
次の表に、ID パラメータで出力できる値をリストします。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外

2523	9507	書き込みエラー:オペランドが有効な範囲外
2524	9508	読み取りエラー:無効なオペランド
2525	9509	書き込みエラー:無効なオペランド
2528	9512	読み取りエラー:データのアラインメント
2529	9513	書き込みエラー:データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー:データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー:DB が存在しない
2539	9529	アクセスエラー:不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、ファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、ファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー:DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー:入力
2943	10563	書き込みエラー:出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

例

次の例で、命令がどのように機能するかを示します。



「#Field[#index]」タグへのアクセス時にエラーが発生しました。 読み取り/アクセスエラーにも関わらず、「乗算」命令のイネーブル出力 ENO と「#TagOut_Enabled」オペランドの信号状態が「1」で、値「0.0」を使用して乗算が実行されます。 このエラーシナリオが発生した場合は、エラーを取得するために、「乗算」命令の後に「ローカルでエラー ID を取得」命令をプログラミングすることを推奨します。「ローカルでエラー ID を取得」命令によって供給されるエラー情報は、比較命令「等しい」を使用して評価されます。 オペランド「#TagID」が ID 2522 を返す場合、読み取り/アクセスエラーが伴い、値「100.0」が「#TagOut」出力に書き込まれます。

INIT_RD: すべての維持データを初期化

説明

「すべての保持データを初期化」命令を使用して、すべてのデータブロック、ビットメモリ、および SIMATIC タイマとカウンタの保持データを同時にリセットできます。 この命令は、実行するとプログラムサイクル持続時間を超過する場合があるため、スタートアップ OB 内でのみ実行が可能です。

パラメータ

次の表に、「すべての保持データを初期化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数	入力「REQ」の信号状態が「1」の場合、すべての保持データがリセットされます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、エラーコードが RET_VAL パラメータに出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

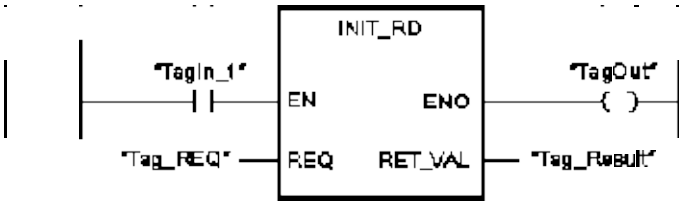
パラメータ RET_VAL

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
80B5	この命令はスタートアップ OB 内にプログラムされていないため、実行できません。
一般エラー情報	関連項目 "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「Tag_REQ」の信号状態が「1」の場合、命令が実行されます。 すべてのデータブロック、ビットメモリ、および SIMATIC タイマとカウンタの保持データがリセットされます。命令がエラーなく実行されると、ENO イネーブル出力の信号状態が「1」となります。

WAIT:遅延時間の設定

説明

「遅延時間の設定」命令は、指定された時間の間プログラム実行を一時停止します。 命令の WT パラメータに、マイクロ秒で時間を示します。

遅延時間は、-32768 から最大+32767 マイクロ秒 (µs)が設定可能です。 設定可能な最短の遅延時間は個々の CPU に依存し、「遅延時間の設定」命令の実行時間に対応します。

この命令の実行には、さらに優先度の高いイベントが割り込む可能性があります。

「遅延時間の設定」命令はエラー情報を返しません。

パラメータ

次の表に、「遅延時間の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
WT	Input	INT	I、Q、M、D、L、P、または定数	遅延時間(µs)

ランタイム:プログラムランタイムの測定

説明

「プログラムランタイムの測定」命令を使用して、プログラム全体、個々のブロック、またはコマンドシーケンスのランタイムを測定します。

プログラム全体のランタイムを測定する場合は、「プログラムランタイムの測定」命令を OB1 で呼び出します。 最初の呼び出しでランタイムの測定が開始され、2 回目の呼び出し後に出力 RET_VAL がプログラムのランタイムを返します。 測定されたランタイムには、たとえば上位レベルのイベントや通信による割り込みなど、プログラム実行中に発生する可能性のあるすべての CPU プロセスが含まれます。 「プログラムランタイムの測定」命令は CPU の内部カウンタを読み取り、値を IN-OUT パラメータ MEM に書き込みます。 この命令は、内部カウンタ周波数に基づいて現在のプログラムランタイムを計算し、出力 RET_VAL に書き込みます。

個々のブロック、または個々のコマンドシーケンスのランタイムを測定する場合、3 つの個別のネットワークが必要です。 プログラム内の個々のネットワークで、「プログラムランタイムの測定」命令を呼び出します。 命令のこの最初の呼び出しでランタイム測定の開始位置を設定します。 その後、次のネットワーク内で目的のプログラムブロック、またはコマンドシーケンスを呼び出します。 他のネットワーク内で、「プログラムランタイムの測定」命令の 2 回目の呼び出しを行い、命令の最初の呼び出しで行ったように同じメモリを IN-OUT パラメータ MEM に割り当てます。 3 番目のネットワークの「プログラムランタイムの測定」命令は内部 CPU カウンタを読み取り、内部カウンタ周波数に基づいてプログラムブロック、またはコマンドシーケンスの現在のランタイムを計算し、それを出力 RET_VAL に書き込みます。

以下は、ファームウェアバージョン V4.1 未満の S7-1200 が対象です。 「プログラムランタイムの測定」命令は、内部的な高周波数カウンタを使用して時間を計算します。 カウンタがオーバーフローする場合(これは 1 分当たり 1 回発生する可能性があります)、命令は値<= 0.0 を返します。 これらのランタイム値は無視してください。

注記
コマンドシーケンスのランタイムは、正確に測定することはできません。これは、コマンドシーケンス内の命令のシーケンスが、プログラムを最適にコンパイルする過程で変更されるためです。

パラメータ

次の表に、「プログラムランタイムの測定」命令のパラメータを示します。

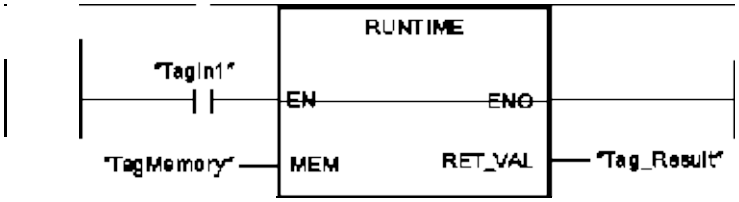
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
MEM	InOut	LREAL	I、Q、M、D、L	ランタイム測定の開始位置を保存します。
RET_VAL	Output	LREAL	I、Q、M、D、L	測定したランタイムを秒単位で返します

有効なデータタイプの追加情報については、「関連項目」を参照してください。

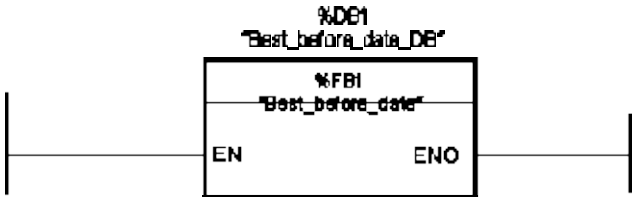
例

次の例は、命令がプログラムブロックのランタイム計算に基づいてどのように動作するかを示します。

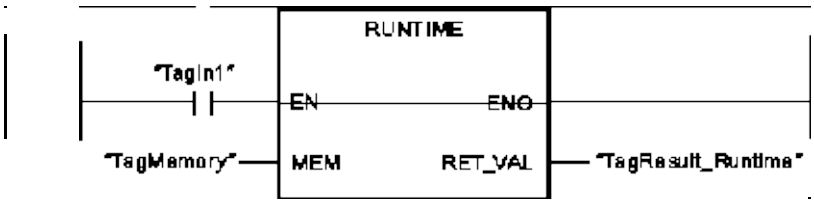
ネットワーク 1:



ネットワーク 2:



ネットワーク 3:



ネットワーク 1 の「TagIn1」オペランドのシグナル状態が「1」の場合、この命令が実行されます。この命令の最初の呼び出しでランタイムの測定が開始され、TagMemory オペランドでの命令の 2 回目の呼び出しの参照先としてバッファリングされます。

ネットワーク 2 で、「Best_before_date」プログラムブロック FB1 が呼び出されます。

FB1 プログラムブロックの実行が終了し、「TagIn1」オペランドのシグナル状態が「1」の場合、ネットワーク 3 の命令が実行されます。 命令の 2 回目の呼び出しにより、プログラムブロックのランタイムが計算され、結果が出力 RET_VAL に書き込まれます。

ワード論理演算

この章には下記に関する情報が記載されています：

AND: AND 演算

説明

「論理積(AND)演算」命令を使用して、IN1 入力の値と IN2 入力の値を 1 ビットずつ AND 論理によって結合し、結果を OUT 出力で照会できます。

命令が実行されると、IN1 入力の値のビット 0 と IN2 入力の値のビット 0 が論理的に AND 演算されます。結果は出力 OUT のビット 0 に格納されます。 同じ論理演算が、指定された値の他のすべてのビットに実行されます。

入力の数は、命令ボックスで拡張できます。 追加された入力は、ボックスで昇順に番号付けされます。 命令が実行されると、すべての使用可能な入力パラメータの値が、AND 論理で結合(AND 演算)されます。 結果が、OUT 出力に保存されます。

論理演算の両方のビットのシグナル状態が「1」になる場合のみ、結果ビットのシグナル状態が「1」になります。 論理演算の 2 つのビットの 1 つのシグナル状態が「0」の場合、対応する結果ビットはリセットされます。

パラメータ

次の表に、「AND 演算」命令のパラメータを示します。

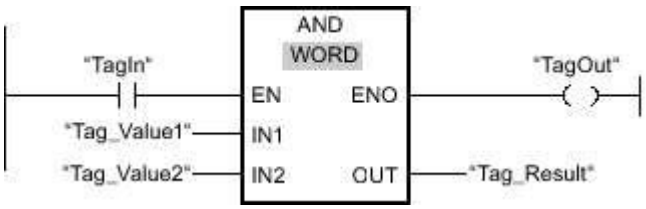
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN1	Input	ビット列	I、Q、M、D、L、P、または定数	論理演算の最初の値
IN2	Input	ビット列	I、Q、M、D、L、P、または定数	論理演算の 2 番目の値
INn	Input	ビット列	I、Q、M、D、L、P、または定数	値が論理的に結合されるその他の入力
OUT	Output	ビット列	I、Q、M、D、L、P	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値とオペランド「Tag_Value2」の値が AND 演算されます。結果は 1 ビットずつマッピングされ、オペランド「Tag_Result」に出力されます。 許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

OR: OR 論理和演算

説明

「論理和(OR)演算」命令を使用して、IN1 入力の値と IN2 入力の値を 1 ビットずつ OR 論理によって結合し、結果を OUT 出力で照会できます。

命令が実行されると、IN1 入力の値のビット 0 と IN2 入力の値のビット 0 が OR 論理和演算によって結合されます。 結果は出力 OUT のビット 0 に格納されます。 同じ論理演算が、指定されたタグのすべてのビットについて実行されます。

入力の数は、命令ボックスで拡張できます。 追加された入力は、ボックスで
昇順に番号付けされます。 命令が実行されると、すべての使用可能な入力パラメータの値が、OR 論理で結合(OR 演算)されます。 結果が、OUT 出力に保存されます。

論理演算の 2 つのビットのうち少なくとも 1 つのシグナル状態が「1」ならば、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が「0」の場合、対応する結果ビットがリセットされます。

パラメータ

次の表に、「OR 演算」命令のパラメータを示します。

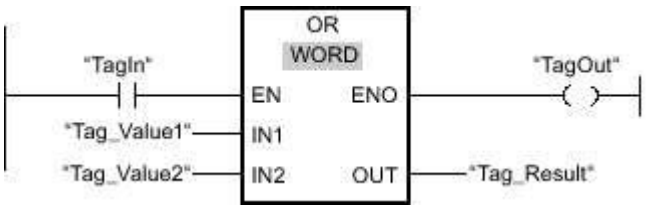
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN1	Input	ビット列	I、Q、M、D、L、P、または定数	論理演算の最初の値
IN2	Input	ビット列	I、Q、M、D、L、P、または定数	論理演算の 2 番目の値
INn	Input	ビット列	I、Q、M、D、L、P、または定数	値が論理的に結合されるその他の入力
OUT	Output	ビット列	I、Q、M、D、L、P	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値とオペランド「Tag_Value2」の値が OR 演算されます。結果は 1 ビットずつマッピングされ、オペランド「Tag_Result」に出力されます。許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

XOR: EXCLUSIVE OR 排他的論理和演算

説明

「排他的論理和演算」命令を使用して、IN1 入力の値と IN2 入力の値を 1 ビットずつ排他的 OR 論理によって結合し、結果を OUT 出力で照会することができます。

命令が実行されると、IN1 入力の値のビット 0 と IN2 入力の値のビット 0 が排他的 OR 論理和演算によって結合されます。 結果は出力 OUT のビット 0 に格納されます。 同じ論理演算が、指定された値のすべての他のビットについて実行されます。

入力の数は、命令ボックスで拡張できます。 追加された入力は、ボックスで昇順に番号付けされます。 命令が実行されると、すべての使用可能な入力パラメータの値が、排他的 OR 論理で結合されます。結果が、OUT 出力に保存されます。

論理演算の 2 つのビットのうち 1 つのシグナル状態が「1」の場合、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が「1」または「0」の場合、対応する結果ビットがリセットされます。

パラメータ

次の表に、「排他的論理和演算」命令のパラメータを示します。

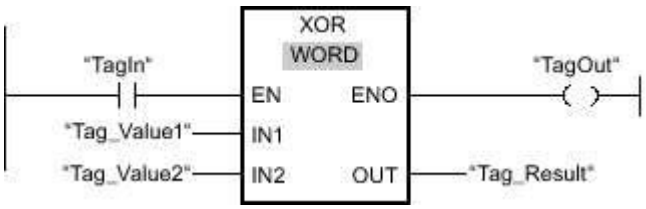
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN1	Input	ビット列	I、Q、M、D、L、P、または定数	論理演算の最初の値
IN2	Input	ビット列	I、Q、M、D、L、P、または定数	論理演算の 2 番目の値
INn	Input	ビット列	I、Q、M、D、L、P、または定数	値が論理的に結合されるその他の入力
OUT	Output	ビット列	I、Q、M、D、L、P	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値とオペランド「Tag_Value2」の値が、排他的 OR 論理によって結合されます。 結果は 1 ビットずつマッピングされ、オペランド「Tag_Result」に出力されます。許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

INVERT: 1 の補数の作成

説明

「1 の補数の作成」命令を使用して、入力 IN のビットのシグナル状態を反転します。 この命令が処理されると、16 進数マスク(16 ビットの数では W#16#FFFF、または 32 ビットの数では DW#16#FFFF FFFF)によって入力 IN の値が排他的 OR にリンクされます。 これによって、個々のビットのシグナル状態を反転させ、出力 OUT に格納します。

パラメータ

次の表に、「1 の補数の作成」命令のパラメータを示します。

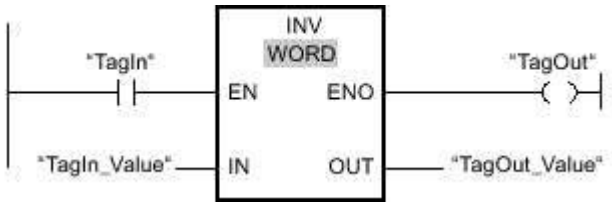
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	ビット列、整数	I、Q、M、D、L、P	入力 IN の値の 1 の補数

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「TagIn_Value」の個々のビットのシグナル状態を反転させ、結果を出力「TagOut_Value」に書き込みます。許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

DECO: デコード

説明

「デコード」命令を使用して、入力値で指定されたビットを出力値にセットできます。

「デコード」命令は、IN 入力の値を読み出し、そのビットを出力値にセットします。そのビット位置は、読み出された値に対応します。 出力値の他のビットはゼロで埋められます。 IN 入力の値が 31 よりも大きい場合、剰余 32 命令が実行されます。

パラメータ

次の表に、「デコード」命令のパラメータを示します。

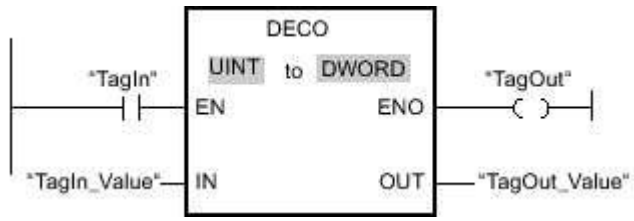
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	UINT	I、Q、M、D、L、P、または定数	設定される出力値のビット位置
OUT	Output	ビット列	I、Q、M、D、L、P	出力値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

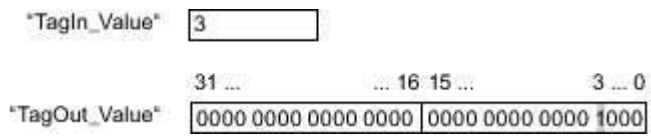
有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令はオペランド「TagIn_Value」の値からビット番号「3」を入力で読み取り、出力で 3 番目のビットをオペランド「TagOut_Value」の値にセットします。

この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

ENCO: エンコード

説明

「エンコード」命令を使用して、入力値の最下位ビットのビット番号を読み出し、OUT 出力に送信できます。

「エンコード」命令は、IN 入力の値の最下位ビットを選択し、そのビット番号を OUT 出力のタグに書き込みます。

パラメータ

次の表に、「エンコード」命令のパラメータを示します。

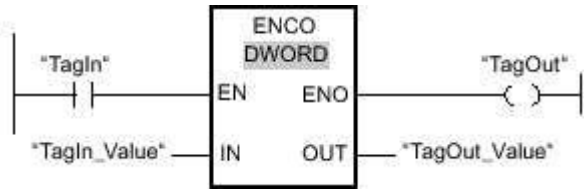
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列	I、Q、M、D、L、P、または定数	入力値
OUT	Output	INT	I、Q、M、D、L、P	出力値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

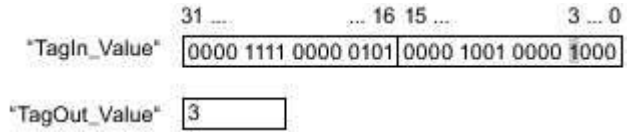
有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、「TagIn_Value」入力の最下位ビットを選択し、ビット位置「3」を「TagOut_Value」出力のタグに書き込みます。

この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

SEL: 選択

説明

スイッチ(G 入力)に応じて、「選択」命令は、入力 IN0 または IN1 の 1 つを選択し、その内容を OUT 出力にコピーします。 入力 G のシグナル状態が「0」の場合、入力 IN0 の値が移動されます。 入力 G のシグナル状態が「1」の場合、入力 IN1 の値が出力 OUT に移動されます。

すべてのパラメータのすべてのタグは、同じデータタイプであることが必要です。

パラメータ

次の表に、「選択」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
G	Input	BOOL	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、T、C、または定数	スイッチ
IN0	Input	ビット列、整数、浮動小数点数、タイマ、TOD、CHAR、WCHAR、DATE	ビット列、整数、浮動小数点数、タイマ、TOD、LTOD、LDT、CHAR、WCHAR、DATE	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	最初の入力値
IN1	Input	ビット列、整数、浮動小数点数、タイマ、TOD、CHAR、WCHAR、DATE	ビット列、整数、浮動小数点数、タイマ、TOD、LTOD、LDT、CHAR、WCHAR、DATE	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	2 番目の入力値
OUT	Output	ビット列、整数、浮動小数点数、タイマ、TOD、CHAR、WCHAR、DATE	ビット列、整数、浮動小数点数、タイマ、TOD、LTOD、	I、Q、M、D、L、P	I、Q、M、D、L、P	結果

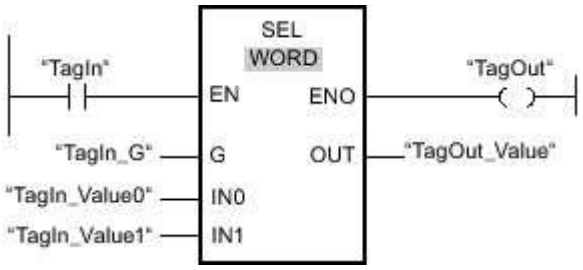
			LDT、CHAR、WCHAR、DATE			

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。「TagIn_G」入力のシグナル状態に基づいて、「TagIn_Value0」または「TagIn_Value1」入力の値が選択され、「TagOut_Value」出力に移動されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」になり、「TagOut」出力がセットされます。

MUX: 多重化

説明

「多重化」命令を使用して、選択した入力の内容を出力 OUT にコピーします。 命令ボックスの選択可能な入力の数は拡張できます。 最大 32 の入力を宣言できます。

入力はボックスで自動的に割り当てられます。 番号の割り当ては IN0 から開始され、新しい入力ごとに継続的にインクリメントされます。K パラメータを使用して、その内容が OUT 出力にコピーされる入力を定義できます。 K パラメータの値が使用可能な入力の数よりも大きい場合、ELSE パラメータの内容が OUT 出力にコピーされ、ENO 許可出力にシグナル状態「0」が割り当てられます。

「多重化」命令は、すべての入力と OUT 出力のタグが同じデータタイプの場合にのみ実行できます。K パラメータは、指定できるのが整数のみであるため、例外です。

次のいずれかの条件が満たされると、許可出力 ENO がリセットされます。

- 許可入力 EN のシグナル状態が「0」であること。
- K パラメータの入力は、使用可能な入力の外側にあります。 これは、ELSE 入力が使用されるかどうかに関わらず適用されます。 OUT 出力の値は変更されたままになります。
- 命令の実行中にエラーが発生していること。

パラメータ

次の表に、「多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
K	Input	整数	整数	I、Q、M、D、L、P、または定数	内容がコピーされる入力を指定します。 • K = 0 ならば、パラメータ IN0 • K = 1 ならば、パラメータ IN1 など
IN0	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	最初の入力値
IN1	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、	2 進数、整数、浮動小数点数、タイマ、CHAR、	I、Q、M、D、L、P、または定数	2 番目の入力値

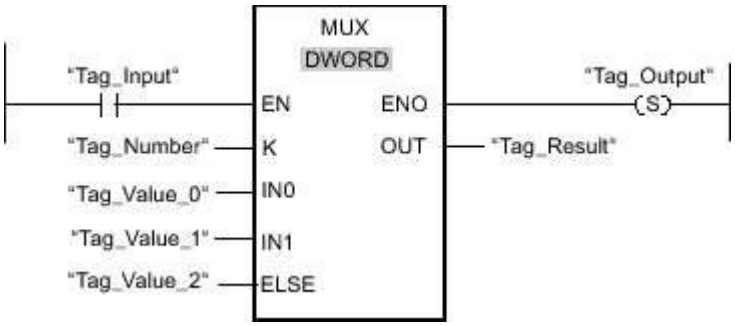
		WCHAR、TOD、DATE	WCHAR、TOD、LTOD、DATE、LDT		
INn	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	オプションの入力値
ELSE	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	K > n の場合コピーする値を指定します。
OUT	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	値がコピーされる出力

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
K	Tag_Number	1
IN0	Tag_Value_0	DW#16#00000000

IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

「Tag_Input」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Number」の値に応じて、入力「Tag_Value_1」の値がコピーされ、出力「Tag_Result」のオペランドに割り当てられます。 この命令がエラーなしで実行された場合、「ENO」および「Tag_Output」許可出力がセットされます。

DEMUX: 逆多重化

説明

「逆多重化」命令を使用して、選択した出力に入力 IN の内容をコピーできます。 選択可能な出力数は命令ボックスで拡張できます。 このボックスで、出力は自動的に番号付けされます。 番号の割り当ては OUT0 から開始され、新しい出力ごとに継続します。K パラメータを使用して、IN 入力の内容がコピーされる出力を定義できます。 その他の出力は、変更されません。 K パラメータの値が使用可能な出力の数よりも大きい場合、ELSE パラメータの入力の内容 IN および許可出力 ENO にシグナル状態「0」が割り当てられます。

「逆多重化」命令は、入力 IN とすべての出力のタグが同じデータタイプの場合のみ実行できます。 K パラメータは、指定できるのが整数のみであるため、例外です。

次のいずれかの条件が満たされると、許可出力 ENO がリセットされます。

- 許可入力 EN のシグナル状態が「0」であること。
- K パラメータの値が、使用可能な出力の数を超えていること。
- 命令の実行中にエラーが発生していること。

パラメータ

次の表に、「逆多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
K	Input	整数	整数	I、Q、M、D、L、P、または定数	入力値(IN)がコピーされる出力を指定します。 • K = 0 ならば、パラメータ OUT0 • K = 1 ならば、パラメータ OUT1 など
IN	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	入力値
OUT0	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、	I、Q、M、D、L、P	最初の出力

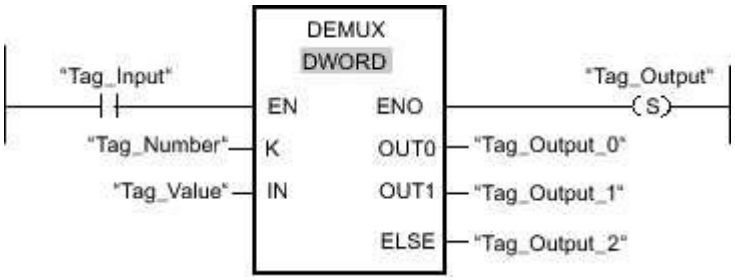
		TOD、DATE	LTOD、DATE、LDT		
OUT1	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	2 番目の出力
OUTn	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	オプションの出力
ELSE	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	K > n の場合に入力値(IN)がコピーされる出力。

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

使用可能なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

ネットワーク実行前の「逆多重化」命令の入力値

パラメータ	オペランド	値	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

ネットワーク実行後の「逆多重化」命令の出力値

パラメータ	オペランド	値	
OUT0	Tag_Output_0	不変	不変
OUT1	Tag_Output_1	DW#16#FFFFFFFF	不変
ELSE	Tag_Output_2	不変	DW#16#003E4A7D

「Tag_Input」のシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Number」の値に応じて、入力「IN」の値が対応する出力にコピーされます。

シフトとローテーション

この章には下記に関する情報が記載されています：

- SHR: 右へシフト (S7-1200, S7-1500)
- SHL: 左へシフト (S7-1200, S7-1500)
- ROR: 右へローテーション (S7-1200, S7-1500)
- ROL: 左へローテーション (S7-1200, S7-1500)

SHR: 右へシフト

説明

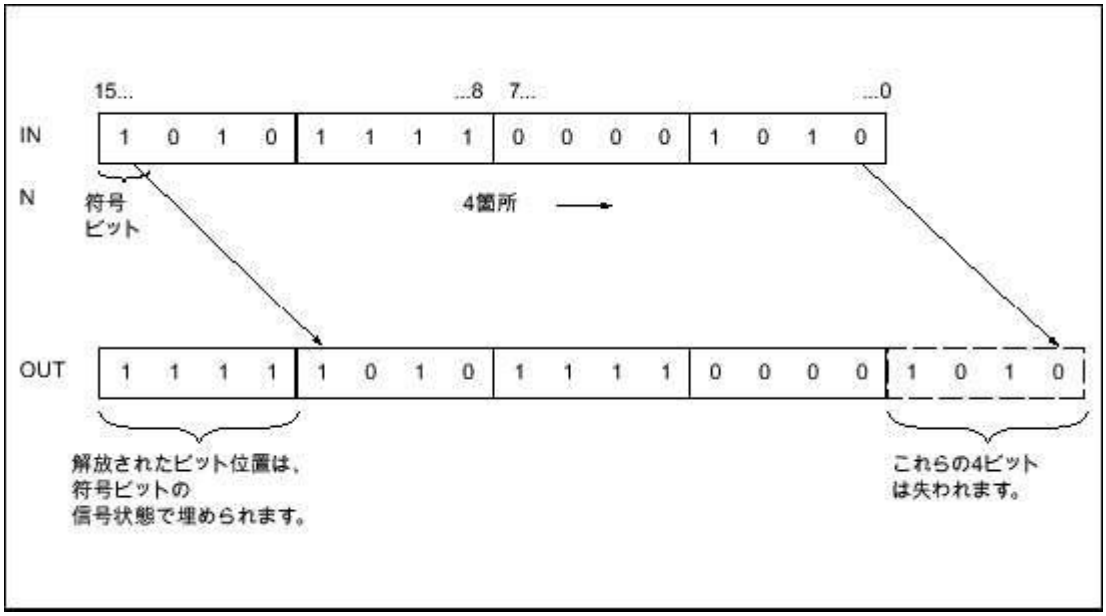
「右へシフト」命令を使用して、入力 IN のオペランドの内容を 1 ビットずつ右へシフトし、結果を OUT 出力で照会できます。 N パラメータを使用し、指定された値をシフトするビット位置の数を指定します。

N パラメータの値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりも N パラメータの値が大きい場合、使用可能なビット位置の数だけ IN 入力のオペランド値が右にシフトされます。

符号なしの値がシフトされると、オペランド領域の左側の解放されたビット位置はゼロで埋められます。 指定された値に符号が存在する場合、解放されたビット位置は符号ビットのシグナル状態で埋められます。

次の図に、整数データタイプのオペランドの内容が、どのように 4 ビット位置右へシフトされるかを示します。



パラメータ

次の表に、「右へシフト」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	シフトされる値

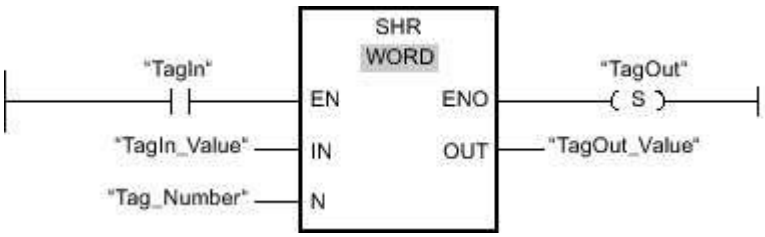
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がシフトされるビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

オペランド「TagIn」のシグナル状態が「1」の場合、「右へシフト」命令が実行されます。 オペランド「TagIn_Value」の内容が、3 ビット位置右へシフトされます。 結果が「TagOut_Value」出力に送信されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

SHL: 左へシフト

説明

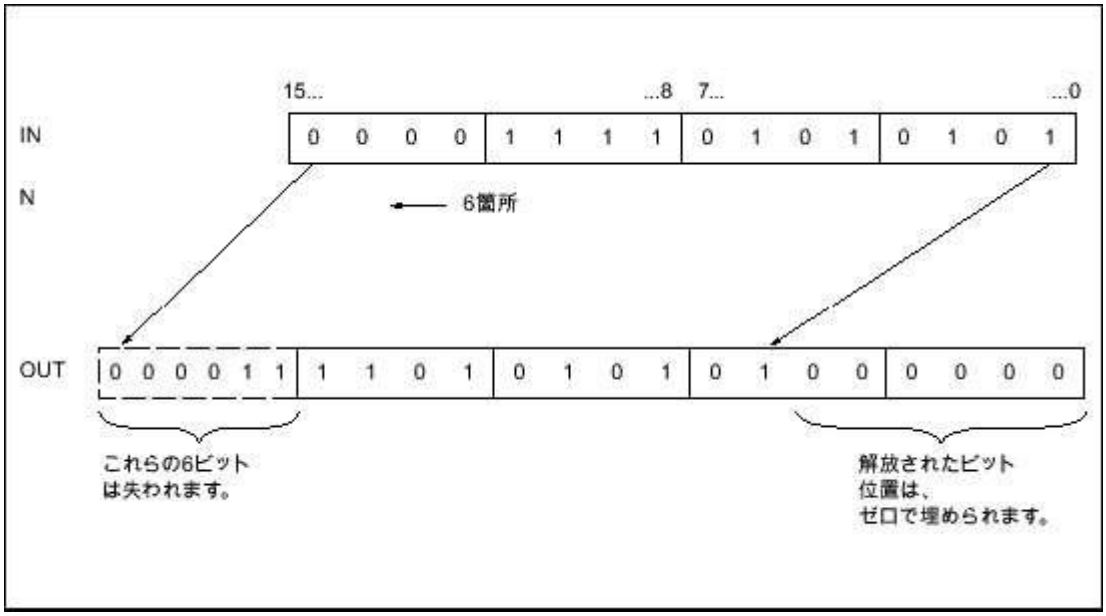
「左へシフト」命令を使用して、入力 IN のオペランドの内容を 1 ビットずつ左へシフトし、結果を OUT 出力で照会できます。 N パラメータを使用し、指定された値をシフトするビット位置の数を指定します。

N パラメータの値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりも N パラメータの値が大きい場合、使用可能なビット位置の数だけ IN 入力のオペランド値が左へシフトされます。

シフトによって解放されたオペランド領域の右側のビット位置は、ゼロで埋められます。

次の図に、WORD データタイプのオペランドの内容が、どのように 6 ビット位置左へシフトされるかを示します。



パラメータ

次の表に、「左へシフト」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	シフトされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がシフトされるビット位置の数

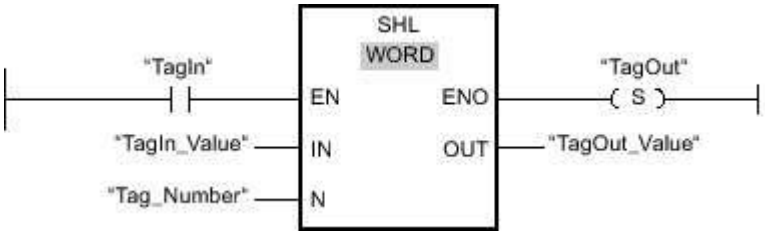
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

オペランド「TagIn」のシグナル状態が「1」の場合、「左へシフト」命令が実行されます。 オペランド「TagIn_Value」の内容は、4 ビット位置左へシフトされます。 結果が「TagOut_Value」出力に送信されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

ROR: 右へローテーション

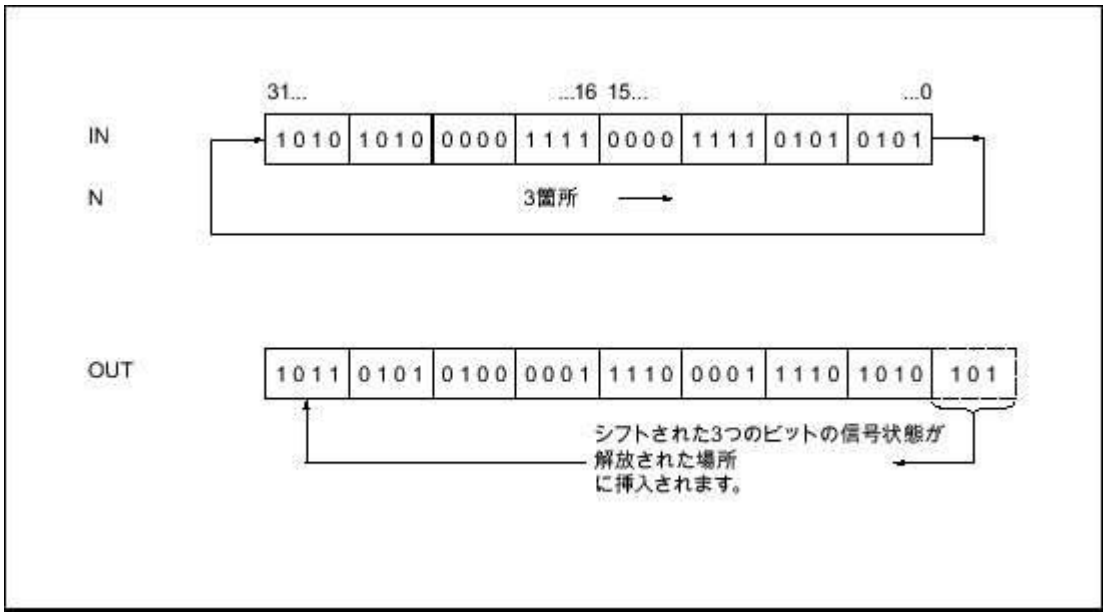
説明

「右へローテーション」命令を使用して、IN 入力のオペランドの内容を 1 ビットずつ右へローテーションし、結果を OUT 出力で照会できます。 N パラメータを使用して、指定された値をローテーションするビット位置の数を指定します。 ローテーションによって解放されたビット位置は、押し出されたビット位置で埋められます。

N パラメータの値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりもパラメータ N の値が大きい場合でも、指定されたビット位置の数だけ IN 入力のオペランド値がローテーションされます。

次の図に、DWORD データタイプのオペランドの内容が、どのように 3 ビット位置右へローテーションされるかを示します。



パラメータ

次の表に、「右へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	ローテーションされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がローテーションされる

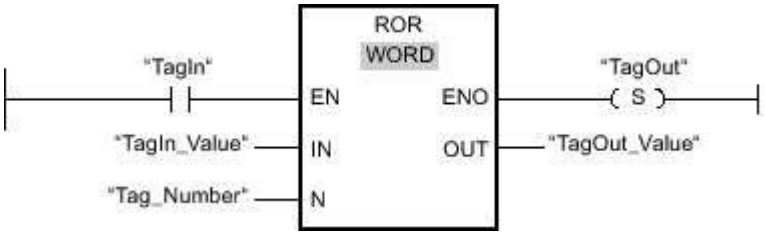
					ビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

オペランド「TagIn」のシグナル状態が「1」の場合、「右へローテーション」命令が実行されます。 オペランド「TagIn_Value」の内容が、5 ビット位置右へローテーションされます。 結果が「TagOut_Value」出力に送信されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

ROL: 左へローテーション

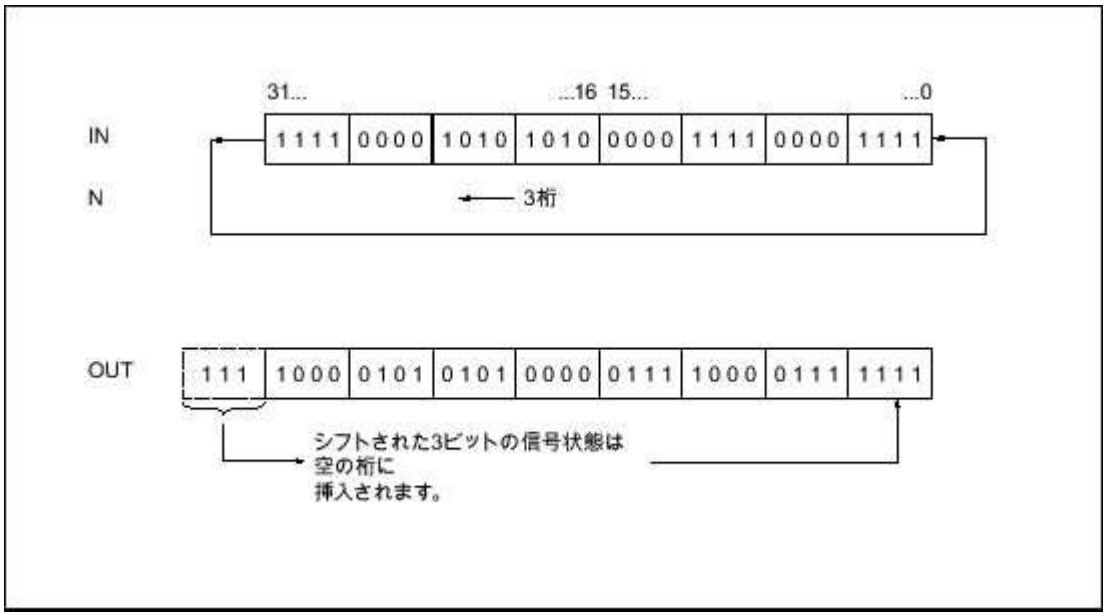
説明

「左へローテーション」命令を使用して、IN 入力のオペランドの内容を 1 ビットずつ左へローテーションし、結果を OUT 出力で照会できます。N パラメータを使用して、指定された値をローテーションするビット位置の数を指定します。ローテーションによって解放されたビット位置は、押し出されたビット位置で埋められます。

N パラメータの値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりもパラメータ N の値が大きい場合でも、指定されたビット位置の数だけ IN 入力のオペランド値がローテーションされます。

次の図に、DWORD データタイプのオペランドの内容が、どのように 3 ビット位置左へローテーションされるかを示します。



パラメータ

次の表に、「左へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	ローテーションされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がローテーションされる

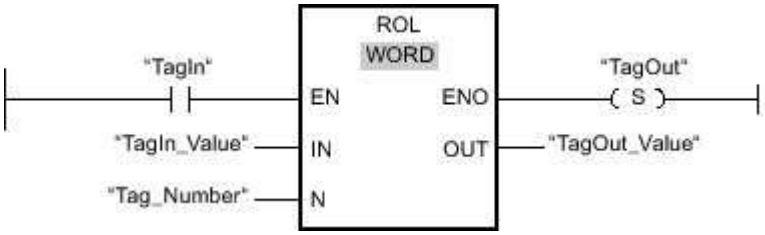
					ビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

入力「TagIn」のシグナル状態が「1」の場合、「左へローテーション」命令が実行されます。 オペランド「TagIn_Value」の内容が、5ビット位置左へローテーションされます。 結果が「TagOut_Value」出力に送信されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

レガシー

この章には下記に関する情報が記載されています：

DRUM: PLC 実行

説明

「PLC の実装」命令を使用して、対応するステップの OUT_VAL パラメータのプログラム済みの値をプログラム済み出力ビット(OUT1～OUT16)および出力ワード(OUT_WORD)に割り当てることができます。 そのため、この命令が特定のステップに留まっている間は、このステップは S_MASK パラメータでプログラムされた許可マスクの条件を満たす必要があります。このステップのイベントが真(TRUE)になっていて現在のステップのプログラムされた時間が経過した場合、または JOG パラメータの値が「0」から「1」に切り替わった場合、この命令は次のステップに進みます。 RESET パラメータのシグナル状態が「1」に切り替わると、この命令はリセットされます。 これによって、プリセットされたステップ(DSP)と現在のステップが等しくなります。

このステップで消費された時間の量は、プリセットタイムベース(DTBP)と各ステップのプリセットカウンタ値(S_PRESET)の積によって計算されます。 新しいステップの開始時に、この計算値が DCC パラメータにロードされます。このパラメータには、現在のステップの残り時間が含まれています。たとえば、DTBP パラメータの値が「2」で、最初のステップのプリセット値が「100」(100 ミリ秒)の場合、DCC パラメータの値は「200」(200 ミリ秒)になります。

ステップは、タイマ値、イベント、またはその両方を使用してプログラム可能です。 イベントビットおよびタイマ値「0」を持つステップは、イベントビットのシグナル状態が「1」になると直ちに次のステップに進みます。 タイマ値のみによってプログラムされたステップは時間を直ちに開始します。「0」よりも大きいイベントビットおよびタイマ値を使用してプログラムされたステップは、イベントビットのシグナル状態が「1」になると時間を開始します。 イベントビットは、シグナル状態「1」で初期化されます。

プログラムされた最後のステップ(LST_STEP)の実行中にこのステップに割り当てられた時間が経過した場合、Q パラメータのシグナル状態は「1」にセットされ、それ以外の場合は「0」にセットされます。 Q パラメータがセットされると、この命令はリセットされるまでこのステップに留まります。

設定可能なマスク(S_MASK)で、出力ワード(OUT_WORD)の個々のビットを選択し、出力値(OUT_VAL)によって出力ビット(OUT1～OUT16)をセットまたはリセットできます。設定可能なマスクのビットのシグナル状態が「1」の場合、OUT_VAL 値は対応するビットをセットまたはリセットします。 設定可能なマスクビットのシグナル状態が「0」の場合、対応するビットは変更されません。16 のステップすべてに設定可能なマスクのすべてのビットは、シグナル状態「1」で初期化されます。

OUT1 パラメータの出力ビットは、出力ワード(OUT_WORD)の最下位ビットに対応します。 OUT16 パラメータの出力ビットは、出力ワード(OUT_WORD)の最上位ビットに対応します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「PLC 実装」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力

RESET	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、リセット条件を示します。
JOG	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「0」から「1」に切り替わると、命令が次のステップに進みます。
DRUM_EN	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「1」になると、PLC はイベントおよび時間条件に基づいて処理を進めます。
LST_STEP	Input	BYTE	I、Q、M、D、L、または定数	プログラムされた最後のステップの番号。
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I、Q、M、D、L、または定数	イベントビット(i); 初期シグナル状態は「1」。
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I、Q、M、D、L	出力ビット(j)
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
OUT_WORD	Output	WORD	I、Q、M、D、L、P	PLC が出力値を書き込むワードアドレス。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
JOG_HIS	Static	BOOL	I、Q、M、D、L	JOG パラメータの履歴ビット
EOD	Static	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
DSP	Static	BYTE	I、Q、M、D、L、P、または定数	PLC のプリセットされたステップ
DSC	Static	BYTE	I、Q、M、D、L、P、または定数	PLC の現在のステップ
DCC	Static	DWORD	I、Q、M、D、L、P、または定数	PLC の現在の数値
DTBP	Static	WORD	I、Q、M、D、L、P、または定数	PLC のプリセットタイムベース
PrevTime	Static	TIME	I、Q、M、D、L、または定数	前のシステム時間
S_PRESET	Static	ARRAY[1..16] of WORD	I、Q、M、D、L、または定数	1 クロックパルス = 1 ミリ秒として、各ステップ[1 ~ 16]のカウントをプリセットします。
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの出力値[1 ~ 16, 0 ~ 15]。
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの設定可能なマスク[1 ~ 16, 0 ~ 15]。 初期シグナル状態は「1」。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

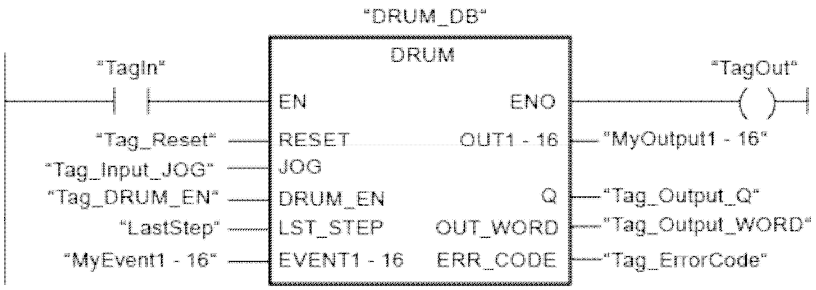
次の表に、ERR_CODE パラメータの値の意味を示します。

ERR_CODE*	説明
W#16#0000	エラーは発生していません。
W#16#000B	LST_STEP パラメータの値が 1 未満であるか、または 16 を超えています。
W#16#000C	DSC パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
W#16#000D	DSP パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例で命令はステップ 1 からステップ 2 に進みます。出力ビット(OUT1～OUT16)および出力ワード(OUT_WORD)はステップ 2 に構成されたマスクおよび OUT_VAL パラメータの値によってセットされます。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータの初期化には以下の値が使用されます。

パラメータ	オペランド	アドレス	値
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE

EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

以下の値が命令の「DRUM_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE

OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

出力パラメータは、命令の実行前に以下の値にセットされます。

パラメータ	オペランド	アドレス	値
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE

OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	アドレス	値
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

命令の実行後、以下の値が命令のインスタンスデータブロック「DRUM_DB」で変更されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002

DCC	DBD16	DW#16#000000C8

DCAT: ディスクリート制御タイマアラーム

説明

「ディスクリート制御タイマアラーム」命令は、CMD パラメータが開くまたは閉じるためのコマンドを発行した時点からの時間を累積するために使用されます。 時間は、指定した時間内でプリセット時間(PT)を超えるか、またはデバイスの開閉に関する情報を受信するまで蓄積します(O_FB または C_FB)。 デバイスの開閉に関する情報を受信する前にプリセット済み時間が経過すると、対応するアラームが有効になります。 プリセット済み時間の前にコマンド入力の信号状態が切り替わると、この時間経過が再開します。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「ディスクリート制御タイマアラーム」命令は入力条件に対して以下のように応答します。

- CMD パラメータの信号状態が「0」から「1」に切り替わると、パラメータ Q、CMD_HIS、ET (ET < PT の場合のみ)、OA、および CA の信号状態は、以下のような影響を受けます。
  - Q および CMD_HIS パラメータが「1」にセットされます。
  - ET、OA および CA パラメータが「0」にリセットされます。
- CMD パラメータの信号状態が「1」から「0」に切り替わると、Q、ET (ET < PT の場合のみ)、OA、CA、および CMD_HIS パラメータが「0」にリセットされます。
- パラメータ CMD および CMD_HIS の信号状態が「1」で、パラメータ O_FB が「0」にセットされると、命令が最後に実行されてからの時間差(ms)がパラメータ ET の値に加算されます。 パラメータ ET の値がパラメータ PT の値を超えると、パラメータ OA の信号状態が「1」にセットされます。 ET パラメータの値が PT パラメータの値を超えなければ、OA パラメータの信号状態が「0」にリセットされます。 CMD_HIS パラメータの値が、CMD パラメータの値にリセットされます。
- パラメータ CMD、CMD_HIS、および O_FB の信号状態が「1」にセットされ、パラメータ C_FB の値が「0」の場合、パラメータ OA の信号状態が「0」にセットされます。 パラメータ ET の値が、パラメータ PT の値にセットされます。O_FB パラメータの信号状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 パラメータ CMD_HIS の値が、パラメータ CMD の値にセットされます。
- CMD、CMD_HIS および C_FB パラメータの値が「0」の場合、命令が最後に実行されてからの時間差(ms)が ET パラメータの値に加算されます。 ETPT パラメータの値がパラメータの値を超えると、パラメータの信号状態が CA「1」にリセットされます。 PT パラメータの値を超えなければ、CA パラメータの信号状態は「0」です。 パラメータ CMD_HIS の値が、パラメータ CMD の値にセットされます。
- CMD、CMD_HIS および O_FB パラメータの信号状態が「0」のときに C_FB パラメータが「1」にセットされると、CA パラメータが「0」にセットされます。 パラメータ ET の値が、パラメータ PT の値にセットされます。C_FB パラメータの信号状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 パラメータ CMD_HIS の値が、パラメータ CMD の値にセットされます。
- O_FB および C_FB パラメータの信号状態が同時に「1」の場合、両方のアラーム出力の信号状態が「1」にセットされます。

「ディスクリート制御タイマアラーム」命令は、エラー情報を返しません。

パラメータ

次の表に、「ディスクリート制御タイマアラーム」命令のパラメータを示します。

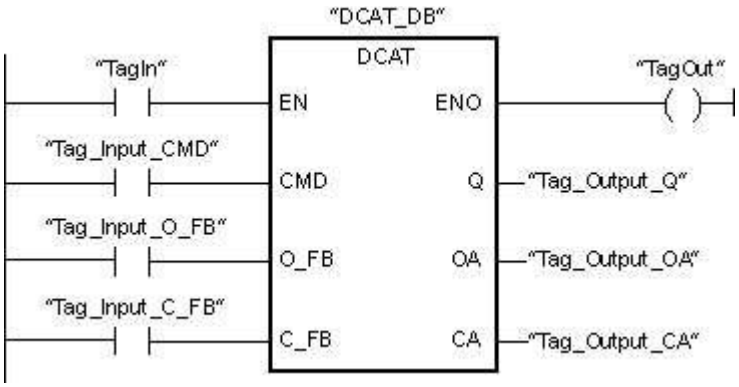
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
CMD	Input	BOOL	I、Q、M、D、L、または定数	信号状態「0」は、「閉じる」コマンドを示します。 信号状態「1」は、「開く」コマンドを示します。
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
Q	Output	BOOL	I、Q、M、D、L	CMD パラメータのステータスを示します
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒
PT	Static	DINT	D、L、または定数	プリセットタイマ値(1 クロックパルス = 1 ミリ秒)
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
CMD_HIS	Static	BOOL	D、L、または定数	CMD の履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例では、パラメータ CMD が「0」から「1」に切り替わります。 命令の実行後、パラメータ Q が「1」にセットされ、2 つのアラーム出力 OA および CA の信号状態は「0」になります。 インスタンスデータブロックのパラメータ CMD_HIS が信号状態「1」にセットされ、パラメータ ET が「0」にリセットされます。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令のインスタンスデータブロック「DCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令のインスタンスデータブロック「DCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

MCAT: モータ制御タイマアラーム

説明

命令「モータ制御タイマアラーム」を使用して、コマンド入力(開または閉)の 1 つが有効になる時点からの時間を累積します。 時間はプリセット済み時間が経過するまで、または関連のフィードバック入力が、デバイスが指定された時間内に要求された操作を実行したことを示す場合に蓄積されます。フィードバックが受信される前にプリセット済み時間が経過すると、対応するアラームがトリガされます。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「モータ制御タイマアラーム」命令の実行

次の表に、さまざまな入力条件に対する「モータ制御タイマアラーム」命令の応答を示します。

入力パラメータ								出力パラメータ								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	ステータス
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening (開き始める)
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open (開く)
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened (開いた)
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm (アラームを開いている)
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing

															(閉じ始める)
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close (閉じる)
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed (閉じた)
>= PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm (アラームを閉じている)
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped (停止)

凡例:	
INC	FB から ET への最後に処理されてからの時間差(ms)を追加します。
PT	PT は ET と同じ値にセットされます。
X	使用不可
<PT	ET < PT
>=PT	ET >= PT

入力パラメータ O_HIS および C_HIS のシグナル状態が両方とも「1」の場合、これらのパラメータのシグナル状態が直ちに「0」にセットされます。 この場合、上記の表の最後の行(X)が有効になります。 入力パラメータ O_HIS および C_HIS のシグナル状態が「1」であるかどうかをチェックすることができなくなるため、この場合の出力パラメータは以下のようにセットされます。

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

「モータ制御タイマアラーム」命令は、エラー情報を返しません。

パラメータ

次の表に、「モータ制御タイマアラーム」命令の応答を示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
O_CMD	Input	BOOL	I、Q、M、D、L、または定数	[開]コマンド入力
C_CMD	Input	BOOL	I、Q、M、D、L、または定数	[閉]コマンド入力
S_CMD	Input	BOOL	I、Q、M、D、L、または定数	[停止]コマンド入力

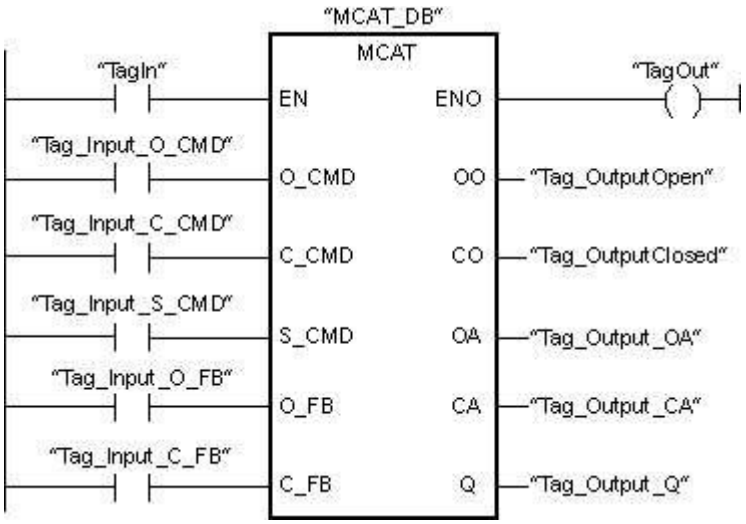
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
OO	Output	BOOL	I、Q、M、D、L	[開]出力
CO	Output	BOOL	I、Q、M、D、L	[閉]出力
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「0」は、エラー条件を示します。
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒
PT	Static	DINT	D、L、または定数	プリセットタイマ値(1 クロックパルス = 1 ミリ秒)
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
O_HIS	Static	BOOL	D、L、または定数	[開]履歴ビット
C_HIS	Static	BOOL	D、L、または定数	[閉]履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

以下の値が命令のインスタンスデータブロック「MCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

以下の値が命令のインスタンスデータブロック「MCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

IMC: 入力ビットをマスクビットと比較

説明

「入力ビットをマスクビットと比較」命令を使用して、最大 16 のプログラムされた入力ビット(IN_BIT0 ~ IN_BIT15)の信号状態とマスクの対応するビットを比較します。 最大 16 のマスクされたステップをプログラム可能です。 IN_BIT0 パラメータの値が、マスク CMP_VAL[x,0]の値と比較されます。「x」はステップ番号を示しています。 CMP_STEP パラメータで、比較に使用するマスクのステップ番号を指定します。 プログラムされた値は、すべて同じ方法で比較されます。 プログラムされていない入力ビットやマスクビットは、デフォルトの信号状態である FALSE になります。

比較で一致が検出されると、OUT パラメータの信号状態が「1」にセットされます。 それ以外の場合、OUT パラメータが「0」にセットされます。

CMP_STEP パラメータの値が 15 よりも大きい場合、この命令は実行されません。 ERR_CODE パラメータでエラーメッセージが出力されます。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「入力ビットをマスクビットと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。
IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。

IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
CMP_STEP	Input	BYTE	I、Q、M、D、L、P、または定数	比較に使用されるマスクのステップ番号。
OUT	Output	BOOL	I、Q、M、D、L	信号状態「1」は、一致が検出されたことを示します。 信号状態「0」は、一致が検出されたことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	**説明**
0000	エラーは発生していません。
000A	CMP_STEP パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

SMC: スキャンマトリックスの比較

説明

「スキャンマトリックスの比較」命令を使用して、最大 16 のプログラム入力ビット(IN_BIT0 ~ IN_BIT15)と比較マスクの対応するビットの信号状態を各ステップで比較します。 処理はステップ 1 から開始し、プログラムされた最後のステップ(LAST)まで、または一致が検出されるまで続行されます。 IN_BIT0 パラメータの入力ビットが、マスク CMP_VAL[x,0]の値と比較されます。「x」はステップ番号を示しています。 プログラムされた値は、すべて同じ方法で比較されます。 一致が検出されると、OUT パラメータの信号状態が「1」にセットされ、一致するマスクのステップ番号が OUT_STEP パラメータに書き込まれます。 プログラムされていない入力ビットやマスクビットは、デフォルトの信号状態である FALSE になります。 複数のステップに一致するマスクが存在する場合、最初に検出されたマスクのみが OUT_STEP パラメータに示されます。 一致が検出されないと、OUT パラメータの信号状態が「0」にセットされます。 この場合、OUT_STEP パラメータの値は LAST パラメータの値よりも「1」だけ大きくなります。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「スキャンマトリックスの比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。
IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。

IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
OUT	Output	BOOL	I、Q、M、D、L	信号状態「1」は、一致が検出されたことを示します。 信号状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT_STEP	Output	BYTE	I、Q、M、D、L、P	一致するマスクを持つステップの番号が含まれているか、または一致が検出されない場合は、LAST パラメータの値よりも「1」だけ大きいステップ番号が含まれています。
LAST	Static	BYTE	I、Q、M、D、L、P、または定数	一致するマスクをスキャンする最後のステップのステップ番号を指定します。
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]:インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	**説明**
0000	エラーは発生していません。
000E	LAST パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

LEAD_LAG:リード/ラグアルゴリズム

説明

「リード/ラグアルゴリズム」命令を使用して、アナログタグの信号を処理します。 GAIN パラメータのゲイン値はゼロよりも大きくなければなりません。「リード/ラグアルゴリズム」命令の結果は、以下の等式を使用して計算されます。

$$OUT = \left[\frac{LG\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_OUT + GAIN \left[\frac{LD\_TIME + SAMPLE\_T}{LG\_TIME + SAMPLE\_T}\right] IN - GAIN \left[\frac{LD\_TIME}{LG\_TIME + SAMPLE\_T}\right] * PREV\_IN$$

「リード/ラグアルゴリズム」命令は、処理を固定プログラムサイクルで行う場合のみ正しい結果が得られます。 LD_TIME、LG_TIME、および SAMPLE_T パラメータには同じ単位を指定する必要があります。LG_TIME > 4 + SAMPLE_T の場合、この命令は以下の関数を実行します。

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

GAIN パラメータの値がゼロ以下の場合、計算は実行されず、ERR_CODE パラメータでエラー情報が出力されます。

「リード/ラグアルゴリズム」命令をダイナミックフィードフォワード制御でループと共に補償器として使用できます。 この命令は、2 つの操作から成ります。「リード」操作は OUT 出力のフェーズをシフトして、出力が入力の前に来るようにします。 一方「ラグ」操作は出力をシフトして、出力が入力の後になるようにします。「ラグ」演算は積分に相当するため、ノイズサプレッサやローパスフィルタとして使用できます。「リード」演算は微分と同等なため、ハイパスフィルタとして使用できます。 2 つの演算(リードおよびラグ)をまとめて実行すると、低周波域では出力フェーズが入力フェーズよりも遅れ、高周波域では出力フェーズが入力フェーズに先行します。 つまり「リード/ラグアルゴリズム」命令は帯域フィルタとして使用できます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「リード/ラグアルゴリズム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	REAL	I、Q、M、D、L、P、または定数	処理対象の現在のサンプル時間(サイクルタイム)の入力値。 IN パラメータには定数も指定できます。

SAMPLE_T	Input	INT	I、Q、M、D、L、P、または定数	サンプル時間 SAMPLE_T パラメータには定数も指定できます。
OUT	Output	REAL	I、Q、M、D、L	命令の結果
ERR_CODE	Output	WORD	I、Q、M、D、L	エラー情報
LD_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のリードタイム。
LG_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のラグタイム。
GAIN	Static	REAL	I、Q、M、D、L、P、または定数	% / %で表されるゲイン(定常状態時の出力の切り替えと入力の切り替えとの比率)。
PREV_IN	Static	REAL	I、Q、M、D、L、P、または定数	前の入力
PREV_OUT	Static	REAL	I、Q、M、D、L、P、または定数	前の出力

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パラメータ ERR_CODE

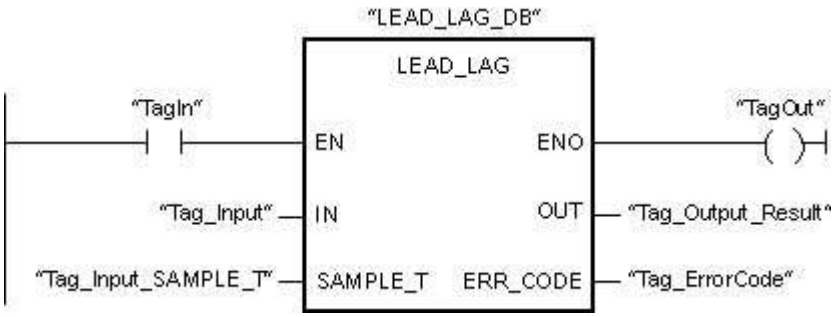
次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	**説明**
0000	エラーは発生していません。
0009	GAIN パラメータの値がゼロ以下である。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

以下の値が命令のインスタンスデータブロック「LEAD_LAG_DB」に保存されます。

パラメータ	アドレス	値
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output_Result	2.0

以下の値が命令のインスタンスデータブロック「LEAD_LAD_DB」に保存されます。

パラメータ	オペランド	値
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

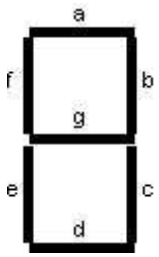
SEG:7 セグ表示のビットパターン作成

説明

「7 セグメント表示のビットパターンの作成」命令を使用して、指定されたソースワードの 4 つの各 16 進数(IN)を 7 セグメント表示の相当するビットパターンに変換します。 命令の結果は、OUT パラメータにダブルワードで出力されます。

16 進数と 7 セグメント(a、b、c、d、e、f、g)の割り当ての間には、以下の関係があります。

入力する数値 (バイナリ)	セグメントの割り当て -gfedcba	表示 (16 進数)	7 セグメント表示
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	



パラメータ

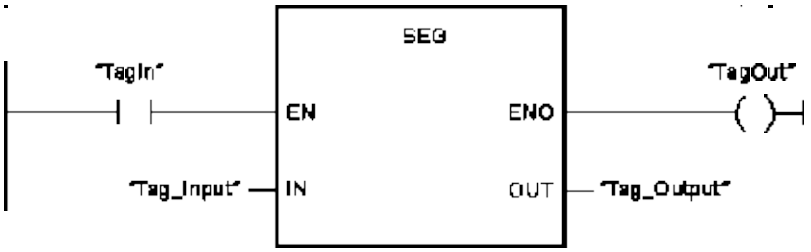
次の表に、「7 セグメント表示のビットパターンの作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	WORD	I、Q、M、D、L、P または定数	4 つの 16 進数のソースワード

OUT	Output	DWORD	I、Q、M、D、L、P	7 セグメント表示のビットパターン

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値	
		16 進数	2 進数
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 表示: 1234

BCDCPL: 10 の補数の作成

説明

「10 の補数の作成」命令を使用して、IN パラメータで指定された 7 桁の BCD 数の 10 の補数を作成できます。 この命令は、次の数式を使用して計算します。

10000000 (BCD として)

– 7 桁の BCD 数

---

10 の補数(BCD として)

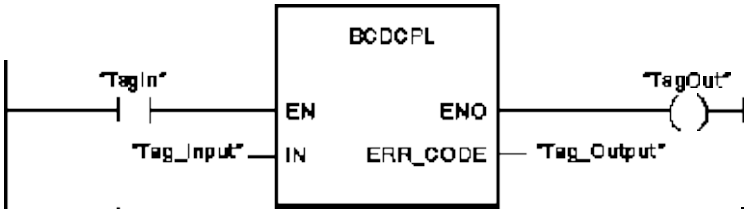
パラメータ

次の表に、「10 の補数の作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	ビット列	I、Q、M、D、L、P、または定数	7 桁の BCD 数
ERR_CODE	Output	DWORD	I、Q、M、D、L、P	命令の結果

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

BITSUM:セットビット数カウント

説明

「セットビット数カウント」命令を使用して、信号状態「1」にセットされるオペランドのビット数をカウントします。 ビットをカウントするオペランドは、 IN パラメータで指定されます。 命令の結果は、RET_VAL パラメータに出力されます。

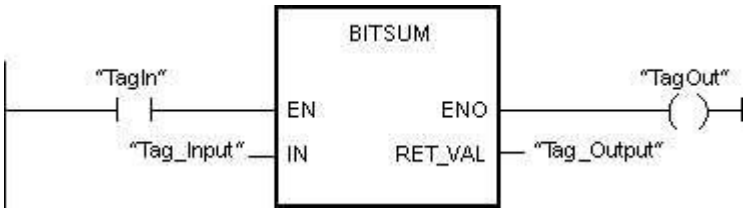
パラメータ

次の表に、「セットビット数カウント」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	DWORD	I、Q、M、D、L、P または定数	セットビット数をカウントするオペランド
RET_VAL	Output	INT	I、Q、M、D、L、P	セットするビットの数

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 ビット)
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

FBD

この章には下記に関する情報が記載されています：

ビット論理演算

この章には下記に関する情報が記載されています：

&: AND 演算

説明

「AND 演算」命令を使って、複数の指定されたオペランドのシグナル状態を照会でき、これらを AND 真理値表に基づいて評価できます。

すべてのオペランドのシグナル状態が「1」の場合、条件は満たされ、この命令が結果「1」を返します。 いずれかのオペランドのシグナル状態が「0」の場合、条件は満たされず、この命令が結果「0」を生成します。

パラメータ

次の表に、「AND 演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	オペランドはシグナル状態がクエリされるビットを示しています。

例

次の例で、命令がどのように動作するかを示します。

出力「TagOut」は、オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合にセットされ、オペランド「TagIn_1」および「TagIn_2」が「0」の場合にリセットされます。

論理積の真理値表

論理演算の結果

次の表に、2 つのオペランドの論理積演算から得られる結果を示します。

最初のオペランドのシグナル状態	2 番目のオペランドのシグナル状態	論理演算の結果
1	1	1
0	1	0
1	0	0
0	0	0

>=1: OR 論理和演算

説明

「OR 演算」命令を使って、複数の指定されたオペランドのシグナル状態を照会でき、これらを OR 真理値表に基づいて評価できます。

いずれかのオペランドのシグナル状態が「1」の場合、条件は満たされ、この命令が結果「1」を返します。 すべてのオペランドのシグナル状態が「0」の場合、条件は満たされず、この命令が結果「0」を生成します。

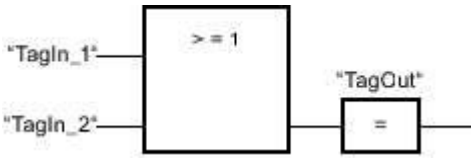
パラメータ

次の表に、「OR 演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	オペランドはシグナル状態がクエリされるビットを示しています。

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」または「TagIn_2」のシグナル状態が「1」のとき、出力「TagOut」が設定されます。

論理和の真理値表

論理演算の結果

次の表に、2 つのオペランドの論理和演算から得られる結果を示します。

最初のオペランドのシグナル状態	2 番目のオペランドのシグナル状態	論理演算の結果
1	0	1
0	1	1
1	1	1
0	0	0

X: EXCLUSIVE OR 排他的論理和演算

説明

「排他的論理和演算」命令を使って、排他的論理和演算真理値表に基づいてシグナル状態照会の結果を照会することができます。

「排他的論理和演算」命令では、2 つの指定されたオペランドのいずれかのシグナル状態が「1」の場合、シグナル状態は「1」です。 3 つ以上のオペランドを照会した場合、奇数の照会されたオペランドが結果「1」を返すと共通の RLO が「1」となります。

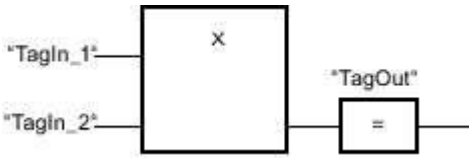
パラメータ

次の表に、「排他的論理和演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	オペランドはシグナル状態がクエリされるビットを示しています。

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」のとき、出力「TagOut」が設定されます。 両方のオペランドがシグナル状態「1」または「0」を返した場合、出力「TagOut」がリセットされます。

排他的論理和の真理値表

論理演算の結果

次の表に、2 つのオペランドの排他的論理和演算から得られる結果を示します。

最初のオペランドのシグナル状態	2 番目のオペランドのシグナル状態	論理演算の結果
1	0	1
0	1	1
1	1	0
0	0	0

次の表に、3 つのオペランドの排他的論理和演算から得られる結果を示します。

最初のオペランドのシグナル状態	2 番目のオペランドのシグナル状態	3 番目のオペランドのシグナル状態	論理演算の結果
1	0	0	1
0	1	1	0
0	1	0	1
1	0	1	0
0	0	1	1
1	1	0	0
1	1	1	1
0	0	0	0

入力の挿入

説明

「入力の挿入」命令を使って、次の命令のいずれかのボックスに入力を追加します。

- "論理積演算
- "論理和演算
- "排他的論理和演算

命令ボックスの拡張によって、複数のオペランドのシグナル状態を照会できます。

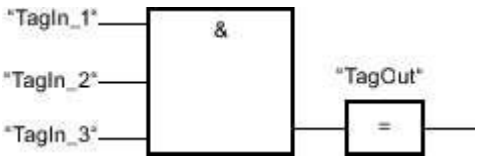
パラメータ

次の表に、「入力の挿入」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	オペランドはシグナル状態がクエリされるビットを示しています。

例

次の例で、命令がどのように動作するかを示します。



「AND 演算」命令のボックスがオペランド「TagIn_3」のシグナル状態が照会される追加の入力によって拡張されています。「TagOut」出力が設定されていると、オペランド「TagIn_1」および「TagIn_2」および「TagIn_3」がシグナル状態「1」を返します。

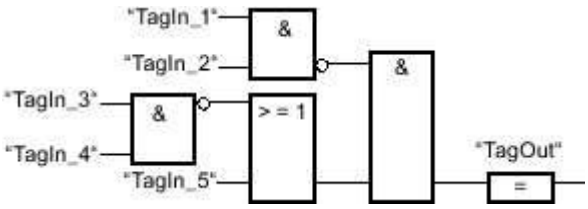
RLO の反転

説明

「RLO の反転」命令を使用し、論理演算の結果(RLO)のシグナル状態を反転します。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- 入力「TagIn_1」および/または「TagIn_2」のシグナル状態が「0」であること。
- 入力「TagIn_3」および/または「TagIn_4」のシグナル状態が「0」であるか、または入力「TagIn_5」のシグナル状態が「1」であること。

=: 割り当て

説明

「割り当て」命令を使って、指定したオペランドのビットを設定できます。 ボックス入力の論理演算(RLO)の結果がシグナル状態「1」の場合、指定したオペランドがシグナル状態「1」にセットされます。 ボックス入力のシグナル状態が「0」の場合、指定したオペランドのビットが「0」にリセットされます。

この命令は、RLO に影響を与えません。 ボックス入力の RLO は、割り当てボックスの上のオペランドに直接割り当てられます。

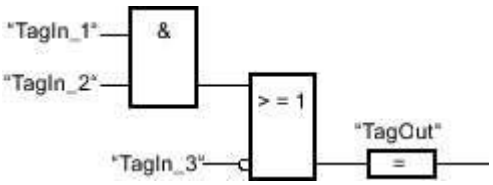
パラメータ

次の表に、「割り当て」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	RLO が割り当てられるオペランド。

例

次の例で、命令がどのように動作するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」が「割り当て」命令の出力に設定されます。

- 入力「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- 入力「TagIn_3」のシグナル状態が「0」であること。

/=: 否定割り当て

説明

「否定割り当て」命令は、論理演算(RLO)の結果を反転し、これをボックスの上のオペランドに割り当てます。 ボックス入力の RLO が「1」の場合、バイナリオペランドはリセットされます。 ボックス入力における RLO が「0」の場合、オペランドは「1」にセットされます。

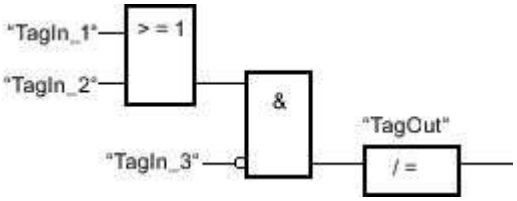
パラメータ

次の表に、「割り当てを無効にする」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	否定された RLO が割り当てられるオペランド。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、オペランド「TagOut」はリセットされます。

- オペランド「TagIn_1」または「TagIn_2」のシグナル状態が「1」であること。
- オペランド「TagIn_3」のシグナル状態が「0」であること。

R: 出力のリセット

説明

「出力のリセット」命令を使って、指定したオペランドのシグナル状態を「0」にリセットできます。

この命令は、ボックス入力の論理演算の結果(RLO)が「1」の場合のみ実行されます。 ボックス入力のシグナル状態が「1」の場合、指定したオペランドが「0」にリセットされます。 ボックス入力の RLO が「0」の場合、指定したオペランドのシグナル状態は変更されません。

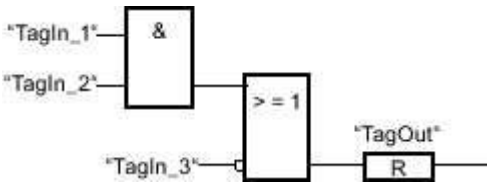
パラメータ

次の表に、「出力のリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand>	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	RLO =「1」の場合にリセットされるオペランド。

例

次の例で、命令がどのように動作するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がリセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- オペランド「TagIn_3」のシグナル状態が「0」であること。

S: 出力設定

説明

「出力のセット」命令を使用すると、指定したオペランドのシグナル状態を「1」にセットすることができます。

この命令は、ボックス入力の論理演算の結果(RLO)が「1」の場合のみ実行されます。 ボックス入力のシグナル状態が「1」の場合、指定したオペランドが「1」にセットされます。 ボックス入力の RLO が「0」の場合、指定したオペランドのシグナル状態は変更されません。

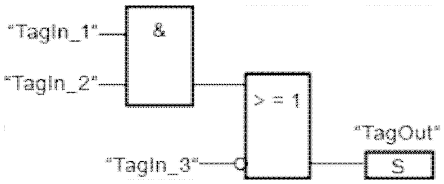
パラメータ

次の表に、「出力のセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	RLO =「1」を使用してセットされるオペランド

例

次の例で、命令がどのように動作するかを示します。



次のいずれかの条件が満たされている場合、オペランド「TagOut」がセットされます。

- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。
- オペランド「TagIn_3」のシグナル状態が「0」であること。

SET_BF: ビットフィールドのセット

説明

命令「ビットフィールドのセット」を使用して、特定のアドレスから始まる複数ビットをセットできます。

入力「N」を使用して、セットするビット数を定義できます。 セットする最初のビットのアドレスは、(<Operand>)で定義されます。 N 入力の値が選択したバイトのビット数よりも大きい場合、次のバイトのビットがセットされます。 これらのビットは、別の命令などで明示的にリセットされるまで、セットされた状態が継続します。

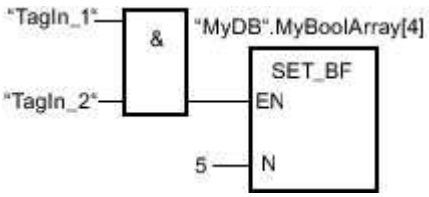
パラメータ

次の表に、「ビットフィールドのセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
N	Input	UINT	定数	定数	セットするビットの数
<Operand>	Output	BOOL	I、Q、M DB または IDB の場合、ブールの配列[..]のエレメント。	I、Q、M DB または IDB の場合、ブールの配列[..]のエレメント。	セットする最初のビットへのポインタ。

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、オペランド "MyDB".MyBoolArray[4]のアドレスで始まる 5 つのビットがセットされます。

RESET_BF: ビットフィールドのリセット

説明

命令「ビットフィールドのリセット」を使用して、特定のアドレスから始まる複数のビットをリセットできます。

N 入力の値を使用して、リセットするビット数を定義できます。 リセットする最初のビットのアドレスは、(<Operand>)で定義されます。 N 入力の値が選択したバイトのビット数よりも大きい場合、次のバイトのビットがリセットされます。 これらのビットは、別の命令などで明示的にセットされるまで、リセットされた状態が継続します。

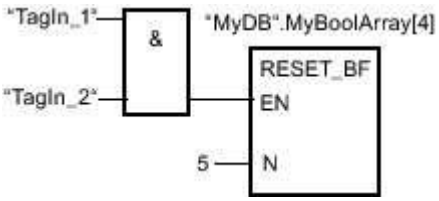
パラメータ

次の表に、「ビットフィールドのリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
N	Input	UINT	定数	定数	リセットするビットの数。
<Operand>	Output	BOOL	I、Q、M DB または IDB の場合、ブールの配列[..]のエレメント。	I、Q、M DB または IDB の場合、ブールの配列[..]のエレメント。	リセットする最初のビットへのポインタ。

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、オペランド "MyDB".MyBoolArray[4]のアドレスで始まる 5 つのビットがリセットされます。

SR: フリップフロップのセット/リセット

説明

「フリップフロップのセット/リセット」命令を使って、入力 S および R1 のシグナル状態に基づいて指定されたオペランドのビットをセットまたはリセットします。 入力 S のシグナル状態が「1」で、入力 R1 のシグナル状態が「0」の場合、指定したオペランドが「1」にセットされます。 入力 S のシグナル状態が「0」で、入力 R1 のシグナル状態が「1」の場合、指定したオペランドが「0」にリセットされます。

入力 R1 が入力 S よりも優先されます。 入力 S および R1 の両方でシグナル状態が「1」の場合、指定したオペランドのシグナル状態が「0」にリセットされます。

2 つの入力 S と R1 のシグナル状態が「0」の場合、この命令は実行されません。 この場合、オペランドのシグナル状態は変更されません。

オペランドの現在のシグナル状態は出力 Q に転送されるため、そこで照会できます。

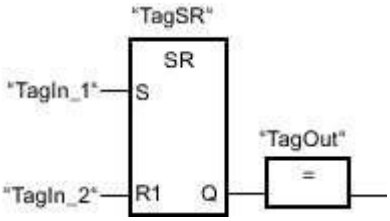
パラメータ

次の表に、「フリップフロップのセット/リセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
S	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	セットの有効化
R1	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	リセットの有効化
<Operand>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	セットまたはリセットされるオペランド
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	オペランドのシグナル状態

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、オペランド「TagSR」および「TagOut」がセットされます。

- オペランド「TagIn_1」のシグナル状態が「1」であること。
- オペランド「TagIn_2」のシグナル状態が「0」であること。

次のいずれかの条件が満たされている場合、オペランド「TagSR」および「TagOut」がリセットされます。

- オペランド「TagIn_1」のシグナル状態が「0」であり、オペランド「TagIn_2」のシグナル状態が「1」であること。
- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。

RS: フリップフロップのリセット/セット

説明

「フリップフロップのリセット/セット」命令を使って、入力 R および S1 のシグナル状態に基づいて指定されたオペランドのビットをセットまたはリセットします。 入力 R のシグナル状態が「1」で、入力 S1 のシグナル状態が「0」の場合、指定したオペランドが「0」にリセットされます。 入力 R のシグナル状態が「0」で、入力 S1 のシグナル状態が「1」の場合、指定したオペランドが「1」にセットされます。

入力 S1 が入力 R よりも優先されます。 入力 R および S1 の両方でシグナル状態が「1」の場合、指定したオペランドのシグナル状態が「1」にセットされます。

2 つの入力 R と S1 のシグナル状態が「0」の場合、この命令は実行されません。 この場合、オペランドのシグナル状態は変更されません。

オペランドの現在のシグナル状態は出力 Q に転送されるため、そこで照会できます。

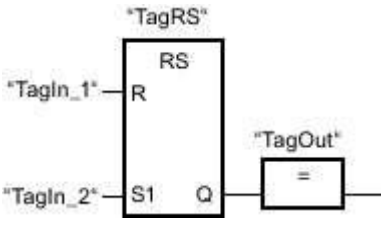
パラメータ

次の表に、「フリップフロップのリセット/セット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
R	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	リセットの有効化
S1	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	セットの有効化
<Operand>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	リセットまたはセットされるオペランド。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	オペランドのシグナル状態

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、オペランド「TagRS」および「TagOut」がリセットされます。

- オペランド「TagIn_1」のシグナル状態が「1」であること。
- オペランド「TagIn_2」のシグナル状態が「0」であること。

次の条件が満たされている場合、オペランド「TagRS」および「TagOut」がセットされます。

- オペランド「TagIn_1」のシグナル状態が「0」であり、オペランド「TagIn_2」のシグナル状態が「1」であること。
- オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」であること。

P: 立ち上がりエッジのオペランドスキャン

説明

「立ち上がりエッジのオペランドスキャン」命令を使用して、指定したオペランド (<Operand1>)のシグナル状態が「0」から「1」に切り替わったかどうかを識別できます。 この命令は、エッジメモリビット(<Operand2>)に保存されている以前のスキャンのシグナル状態と、<Operand1> の現在のシグナル状態を比較します。 この命令が論理演算(RLO)の結果の「0」から「1」への切り替えを検出した場合、正の信号立ち上がりエッジがあります。

立ち上がりエッジを検出した場合、命令の出力のシグナル状態は「1」です。 それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

命令の上のオペランドのプレースホルダで、照会されるオペランド(<Operand1>)を指定します。 命令の下のオペランドのプレースホルダで、エッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、メモリビットが上書きされます。 これがエッジ評価に影響を与え、結果が不定になります。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

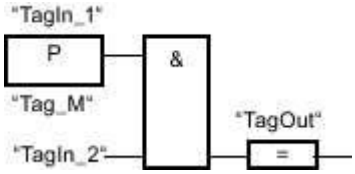
パラメータ

次の表に、「立ち上がりエッジのオペランドスキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	スキャン対象の信号
<Operand2>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	前のスキャンのシグナル状態が保存されるエッジメモリビット。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- 入力「TagIn_1」に信号立ち上がりエッジがあります。
- オペランド「TagIn_2」のシグナル状態が「1」であること。

N: 立ち下がりエッジのオペランドスキャン

説明

「立ち下がりエッジのオペランドスキャン」命令を使用して、指定したオペランド (<Operand1>)のシグナル状態が「1」から「0」に切り替わったかどうかを識別できます。 この命令は、エッジメモリビット(<Operand2>)に保存されている以前のスキャンのシグナル状態と、<Operand1> の現在のシグナル状態を比較します。 この命令が論理演算(RLO)の結果の「1」から「0」への切り替えを検出した場合、負の信号立ち下がりエッジがあります。

立ち下がりエッジを検出した場合、命令の出力のシグナル状態は「1」です。 それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

命令の上のオペランドのプレースホルダで、照会されるオペランド(<Operand1>)を指定します。 命令の下のオペランドのプレースホルダで、エッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、メモリビットが上書きされます。 これがエッジ評価に影響を与え、結果が不定になります。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

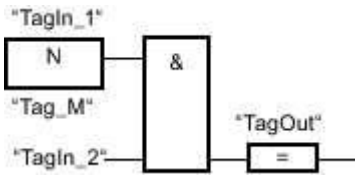
パラメータ

次の表に、「立ち下がりエッジのオペランドスキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
<Operand1>	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	スキャン対象の信号
<Operand2>	InOut	BOOL	I、Q、M、D、L	I、Q、M、D、L	前のスキャンのシグナル状態が保存されるエッジメモリビット。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- 入力「TagIn_1」に信号立ち下がりエッジがあります。
- オペランド「TagIn_2」のシグナル状態が「1」であること。

P=: 立ち上がりエッジのオペランド設定

説明

「立ち上がりエッジのオペランド設定」命令を使用して、論理演算の結果(RLO)が「0」から「1」に切り替わった場合、指定されたオペランド(<Operand2>)を設定します。 この命令は現在の RLO を、エッジメモリビット(<Operand1>)に保存された前の照会からの RLO と比較します。この命令が論理演算(RLO)の結果の「0」から「1」への切り替えを検出した場合、正の信号立ち上がりエッジがあります。

立ち上がりエッジが検出された場合、1 プログラムサイクルの間<Operand2>がシグナル状態「1」にセットされます。それ以外のすべての場合、オペランドのシグナル状態は「0」です。

命令の上のオペランドのプレースホルダで、設定されるオペランド(<Operand2>)を指定します。 命令の下のオペランドのプレースホルダで、エッジメモリビット(<Operand1>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、メモリビットが上書きされます。これがエッジ評価に影響を与え、結果が不定になります。エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

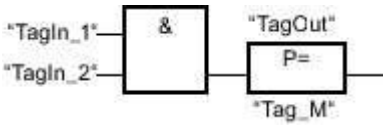
パラメータ

次の表に、「立ち上がりエッジのオペランド設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand2>	Output	BOOL	I、Q、M、D、L	立ち上がりエッジがある場合に設定されるオペランド。
<Operand1>	InOut	BOOL	I、Q、M、D、L	エッジメモリビット

例

次の例で、ブロックパラメータのタイプを示します。



命令ボックスの入力のシグナル状態が「0」から「1」に変わる(立ち上がりエッジ)と、「TagOut」出力が 1 プログラムサイクルの間に設定されます。 それ以外のすべての場合、「TagOut」出力のシグナル状態は「0」になります。

N=: 信号立ち下がりエッジのオペランドの設定

説明

「信号立ち下がりエッジのオペランド設定」命令を使用して、論理演算の結果(RLO)が「1」から「0」に切り替わった場合、指定されたオペランド(<Operand1>)を設定します。 この命令は現在の RLO を、エッジメモリビット(<Operand2>)に保存された前の照会からの RLO と比較します。この命令が論理演算(RLO)の結果の「1」から「0」への切り替えを検出した場合、負の信号立ち下がりエッジがあります。

立ち下がりエッジが検出された場合、1 プログラムサイクルの間<Operand1>がシグナル状態「1」にセットされます。それ以外のすべての場合、オペランドのシグナル状態は「0」です。

命令の上のオペランドのプレースホルダで、設定されるオペランド(<Operand1>)を指定します。 命令の下のオペランドのプレースホルダで、エッジメモリビット(<Operand2>)を指定します。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、メモリビットが上書きされます。これがエッジ評価に影響を与え、結果が不定になります。エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

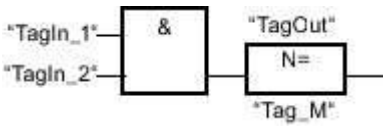
パラメータ

次の表に、「信号立ち下がりエッジのオペランドの設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Output	BOOL	I、Q、M、D、L	立ち下がりエッジがある場合に設定されるオペランド。
<Operand2>	InOut	BOOL	I、Q、M、D、L	エッジメモリビット

例

次の例で、命令がどのように動作するかを示します。



命令ボックスの入力のシグナル状態が「1」から「0」に切り替わった場合(立ち下がりエッジ)、オペランド「TagOut」が 1 プログラムサイクルの間に設定されます。 それ以外のすべての場合、オペランド「TagOut」のシグナル状態は「0」です。

P_TRIG:立ち上がりエッジの RLO スキャン

説明

「立ち上がりエッジの RLO スキャン」命令を使用して、論理演算の結果(RLO)の信号状態での「0」から「1」への切り替えを照会します。 この命令は、RLO の現在の信号状態と、エッジメモリビット(<Operand>)に保存された前に照会された信号状態を比較します。 この命令が論理演算(RLO)の結果の「0」から「1」への切り替えを検出した場合、正の信号立ち上がりエッジがあります。

立ち上がりエッジを検出した場合、命令の出力の信号状態は「1」です。 それ以外のすべての場合、この命令の出力の信号状態は「0」です。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、メモリビットが上書きされます。これがエッジ評価に影響を与え、結果が不定になります。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

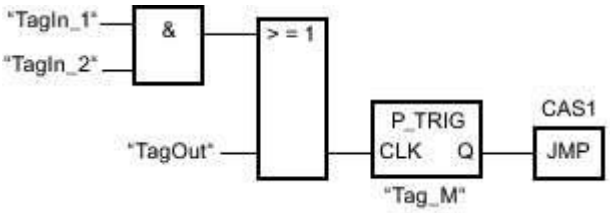
パラメータ

次の表に、「立ち上がりエッジの RLO スキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L	現在の RLO
<Operand>	InOut	BOOL	M、D	以前の照会の RLO が保存されているエッジメモリビット。
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように機能するかを示します。



前のビット論理演算の RLO は、エッジメモリビット「Tag_M」に保存されます。 RLO の信号状態で「0」から「1」の切り替えが検出された場合、プログラムがジャンプラベル CAS1 にジャンプします。

N_TRIG:立ち下がりエッジの RLO スキャン

説明

「立ち下がりエッジの RLO スキャン」命令を使用して、論理演算の結果(RLO)の信号状態での「1」から「0」への切り替えを照会します。 この命令は、RLO の現在の信号状態と、エッジメモリビット(<Operand>)に保存された前に照会された信号状態を比較します。 この命令が論理演算(RLO)の結果の「1」から「0」への切り替えを検出した場合、負の信号立ち下がりエッジがあります。

立ち下がりエッジを検出した場合、命令の出力の信号状態は「1」です。 それ以外のすべての場合、この命令の出力の信号状態は「0」です。

注記
プログラム内でエッジメモリビットのアドレスを複数回使用しないでください。複数回使用した場合、メモリビットが上書きされます。 これがエッジ評価に影響を与え、結果が不定になります。 エッジメモリビットのメモリ領域は、DB (FB の静的領域)またはビットメモリ領域にあることが必要です。

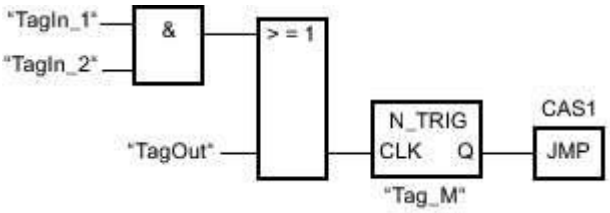
パラメータ

次の表に、「立ち下がりエッジの RLO スキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L	現在の RLO
<Operand>	InOut	BOOL	M、D	以前の照会の RLO が保存されているエッジメモリビット。
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように機能するかを示します。



前のビット論理演算の RLO は、エッジメモリビット「Tag_M」に保存されます。 RLO の信号状態で「1」から「0」の切り替えが検出された場合、プログラムがジャンプラベル CAS1 にジャンプします。

R_TRIG:信号立ち上がりエッジの検出

説明

「信号立ち上がりエッジの検出」命令を使用して、CLK 入力で「0」から「1」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「0」から「1」への状態変化を検出すると、Q 出力で信号立ち上がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

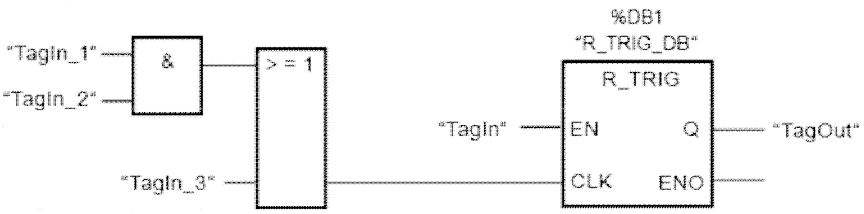
パラメータ

次の表に、「信号立ち上がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
CLK	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、T、C、または定数	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。



CLK 入力のタグでの前の状態は、「R_TRIG_DB」タグに保存されます。 「0」から「1」へのシグナル状態の変化が「TagIn_1」および「TagIn_2」オペランド または「TagIn_3」オペランドで検出されると、「TagOut_Q」出力のシグナル状態は 1 サイクルについて「1」になります。

F_TRIG:信号立ち下がりエッジの検出

説明

「信号立ち下がりエッジの検出」命令を使用して、CLK 入力で「1」から「0」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「1」から「0」への状態変化を検出すると、Q 出力で信号立ち下がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

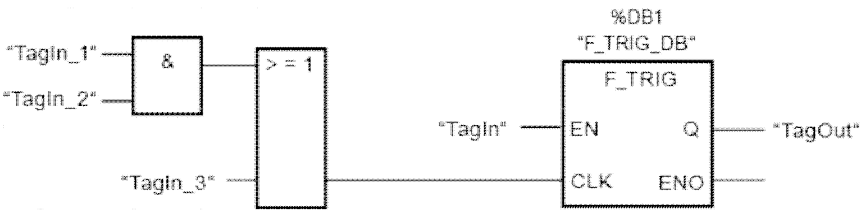
パラメータ

次の表に、「信号立ち下がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
CLK	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、T、C、または定数	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。



CLK 入力のタグでの前の状態は、「F_TRIG_DB」タグに保存されます。 「1」から「0」へのシグナル状態の変化が「TagIn_1」および「TagIn_2」オペランド または「TagIn_3」オペランドで検出されると、「TagOut_Q」出力のシグナル状態は 1 サイクルについて「1」になります。

タイマの動作

この章には下記に関する情報が記載されています：

TP: パルスの生成

説明

「パルス生成」命令を使用して、持続時間 PT の出力 Q を設定できます。 入力 IN で論理演算の結果(RLO)が「0」から「1」(立ち上がりエッジ)に切り替わると、命令が開始します。 設定した持続時間 PT は、命令の開始とともに開始します。 その後の入力信号(立ち上がりエッジ)に関係なく、出力 Q が持続時間 PT の間セットされます。 新しい立ち上がりエッジが検出されても、PT の持続時間内は Q 出力のシグナル状態は影響されません。

現在の時間値は、出力 ET で照会できます。 時間値は T#0s で開始し、持続時間 PT の値に達すると終了します。 設定済みの持続時間 PT に達し、入力 IN のシグナル状態が「0」の場合、ET 出力はリセットされます。

「パルス生成」命令の各呼び出しは、命令データが保存される IEC タイマに割り当てる必要があります。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、出力 ET はタイマの期限が切れるとすぐに定数値を返します。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TP_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TP_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TP_TIME、TP_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「パルス生成」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

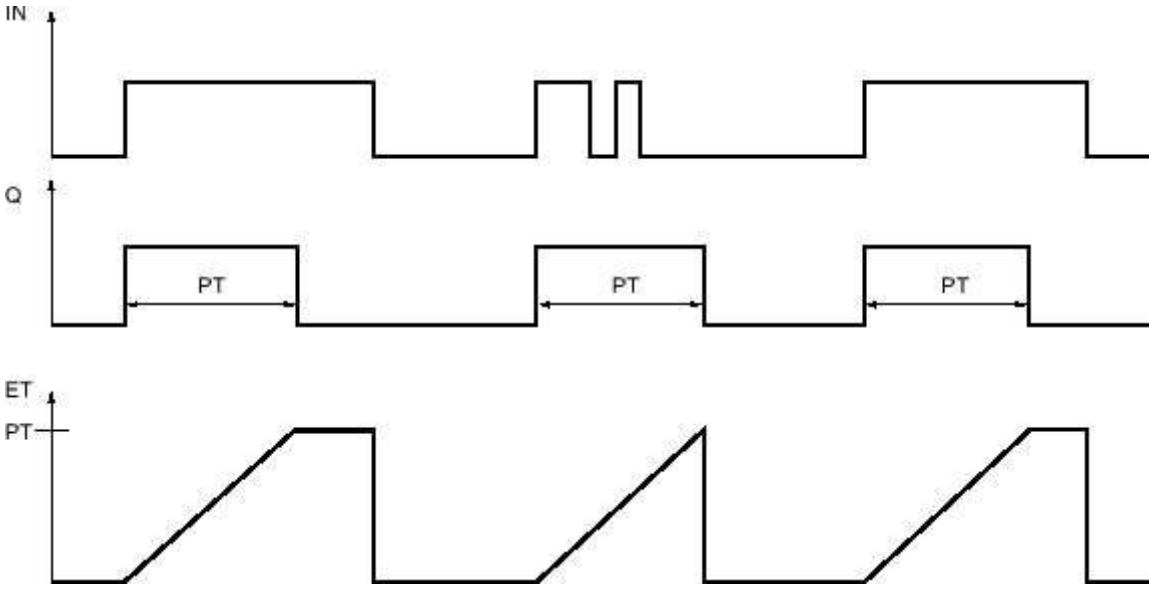
次の表に、「パルス生成」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	パルスの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	パルス出力
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「パルスタイマ」命令のパルスタイミング図を示します。



TON: オンディレータイマ

説明

「オンディレータイマ」命令を使用し、PT 時間で設定された持続時間によって Q 出力のセットを遅延することができます。 入力 IN で論理演算の結果(RLO)が「0」から「1」(立ち上がりエッジ)に切り替わると、命令が開始します。 命令が開始されると、プログラムされた時間 PT が開始します。 長さの PT の期限が切れると、出力 Q のシグナル状態は「1」です。 出力 Q は、開始入力が「1」の間はセットされた状態が続きます。 開始入力のシグナル状態が「1」から「0」に切り替わると、Q 出力がリセットされます。 開始入力で信号の新しい立ち上がりエッジが検出されると、タイマファンクションが再開されます。

現在の時間値は、出力 ET で照会できます。 時間値は T#0s で開始し、持続時間 PT の値に達すると終了します。 IN 入力がシグナル状態「0」に切り替わると、ET 出力が直ちにリセットされます。

「オンディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、出力 ET はタイマの期限が切れるとすぐに定数値を返します。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TON_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TON_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TON_TIME、TON_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「オンディレータイマ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

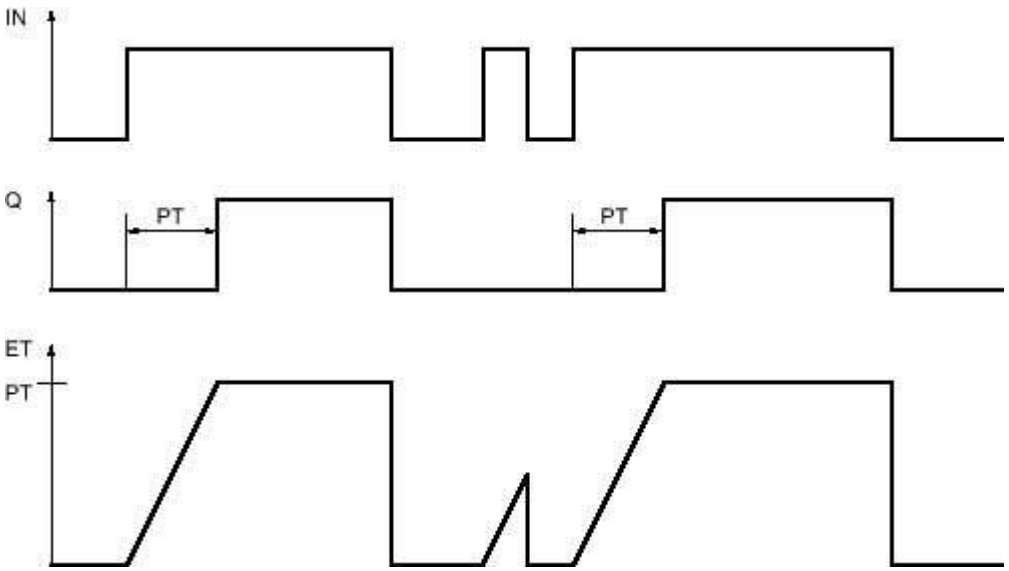
次の表に、「オンディレータイマ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	オンディレーの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	時間 PT の経過時にセットされる出力。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オンディレータイマ」命令のパルスタイミング図を示します。



TOF: オフディレータイマ

説明

「オフディレータイマ」命令を使用し、PT 時間で設定された持続時間によって Q 出力のセットを遅延することができます。 Q 出力は、入力 IN で論理演算の結果(RLO)が「0」から「1」に切り替わった場合(立ち上がりエッジ)に設定されます。 入力 IN で信号状態が「0」に戻る(立ち上がりエッジ)と、設定された持続時間 PT が開始します。 持続時間 PT の間、出力 Q がセットされた状態が継続します。 PT 持続時間が経過すると、Q 出力がリセットされます。 持続時間 PT が経過する前に入力 IN の信号状態が「1」に切り替わると、タイマがリセットされます。 出力 Q の信号状態は、「1」のままになります。

現在の時間値は、出力 ET で照会できます。時間値は T#0s で開始し、持続時間 PT の値に達すると終了します。 持続時間 PT が経過すると、入力 IN が「1」に戻るまで ET 出力は現在値にセットされた状態が継続します。 持続時間 PT が経過する前に入力 IN が「1」に切り替わると、ET 出力が値 T#0s にリセットされます。

「オフディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、出力 ET はタイマの期限が切れるとすぐに定数値を返します。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TOF_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TOF_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TOF_TIME、TOF_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「オフディレータイマ」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

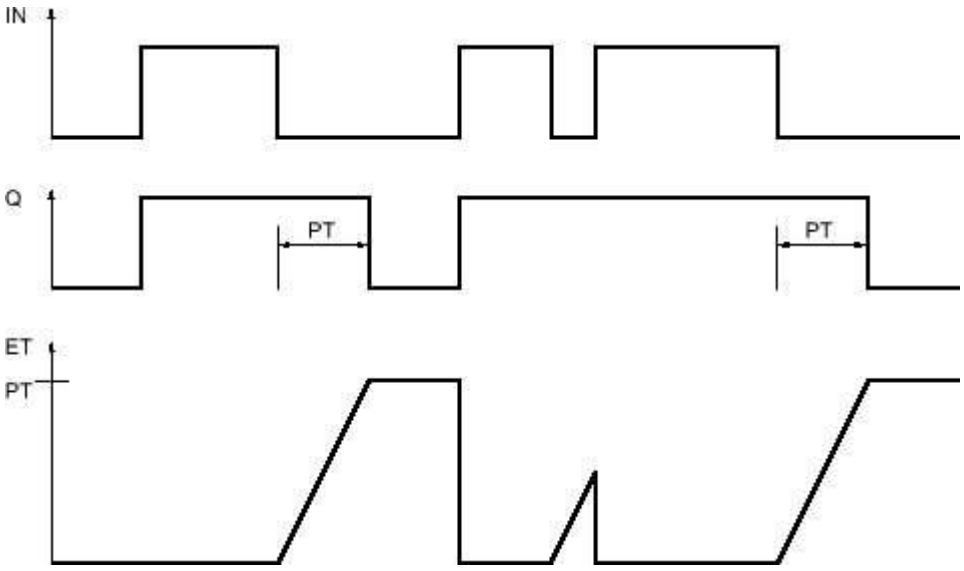
次の表に、「オフディレータイマ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	オフディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	タイマ PT の期限が切れたときにリセットされる出力。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オフディレータイマ」命令のパルスタイミング図を示します。



TONR: タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用して、PT パラメータによってセットされた時間内の時間値を累積します。 この命令は、入力 IN の論理演算の結果(RLO)が「0」から「1」に変わる(立ち上がり工ッジ)と実行され、設定された持続時間 PT が開始します。PT の設定された時間が経過している間、入力 IN で信号状態「1」で記録された時間値が累積されます。 累積時間は、出力 ET に書き込まれ、そこで照会できます。現在の時間値 PT に到達すると、出力 Q の信号状態が「1」になります。 入力 IN が信号状態「0」に変わっても、出力 Q は「1」にセットされた状態が継続します。

開始入力の信号状態に関係なく、R 入力が出力 ET と Q をリセットします。

「タイムアキュムレータ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TONR_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TONR_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TONR_TIME、TONR_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「タイムアキュムレータ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

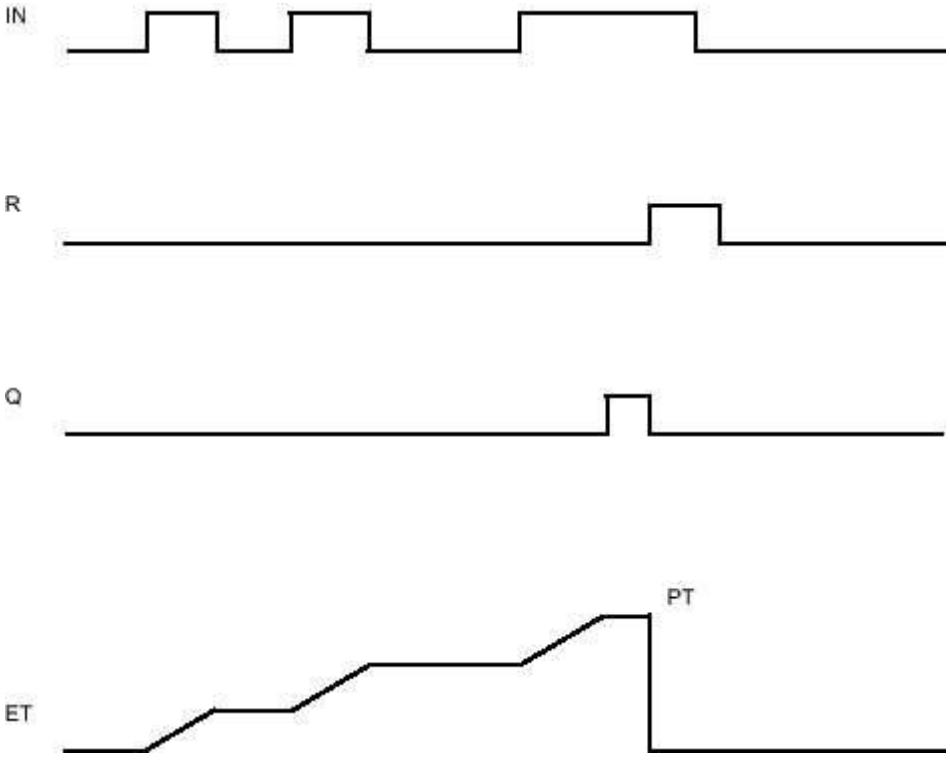
次の表に、「タイムアキュムレータ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
IN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C、P	開始入力
R	Input	BOOL	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	リセット入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	I、Q、M、D、L、P、または定数	時間記録の最大継続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、P	時間 PT の経過時にセットされる出力。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「タイムアキュムレータ」命令のパルスタイミング図を示します。



TP: パルスタイマの起動

説明

「パルスタイマの起動」命令を使用して、パルスとして指定された持続時間分、IEC タイマを起動します。 IEC タイマは、論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がりエッジ)に起動されます。 RLO がその後に変化するかどうかに関係なく、IEC タイマは指定された持続時間分、実行されます。 IEC タイマの期限切れも、新しい信号立ち上がりエッジの検出には影響されません。 IEC タイマを実行している限り、タイマステータスが「1」であるか問い合わせると、シグナル状態が「1」に戻ります。 IEC タイマの期限が切れると、タイマステータスがシグナル状態「0」に戻ります。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、IEC タイマの開始と照会の期限切れのレベルが異なる可能性があります。

S7-1200 CPU の場合

「パルスタイマの起動」命令は、データタイプ IEC_TIMER または TP_TIME の構造体にデータを保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TP_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「パルスタイマの起動」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体にデータを保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TP_TIME、TP_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および指定された IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは、IEC タイマの構造体コンポーネント Q に保存されます。 タイマステータスは、バイナリ論理演算を使って照会できます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「パルスタイマの起動」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「パルスタイマの起動」命令のパラメータを示します。

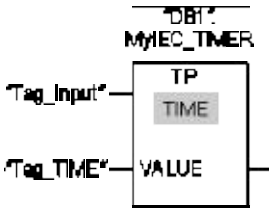
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	IEC タイマが動作する時間。

<IEC timer>	InOut	IEC_TIMER, TP_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME	D、L	開始される IEC タイマ。

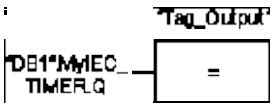
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「パルスタイマの起動」命令は、オペランド Tag_Input のシグナル状態が「0」から「1」に切り替わると実行されます。 タイマ "DB1".MyIEC_TIMER は、オペランド「TagTime」内に格納された時間で開始されます。



タイマ "DB1".MyIEC_TIMER が実行中である限り、タイマステータス("DB1".MyIEC_TIMER.Q)のシグナル状態は「1」であり、オペランド「Tag_Output」がセットされます。 IEC タイマの期限が切れると、時間ステータスがシグナル状態「0」に戻り Tag_Output、オペランドがリセットされます。

TON: オンディレータイマの起動

説明

「オンディレータイマの起動」命令を使用して、オンディレーとして指定された持続時間分、IEC タイマを起動します。 IEC タイマは、論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がりエッジ)に起動されます。 IEC タイマが、指定された持続時間の間、作動します。 命令の入力時の RLO のシグナル状態が「1」の場合、出力はシグナル状態は「1」を返します。 タイマの期限が切れる前に RLO が「0」に切り替わると、動作中の IEC タイマがリセットされます。 「1」のタイマステータスの照会は、シグナル状態「0」を返します。 命令の入力で次の立ち上がりエッジが検出されると、IEC タイマが再起動します。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、IEC タイマの開始と照会の期限切れのレベルが異なる可能性があります。

S7-1200 CPU の場合

「オンディレータイマの起動」命令がそのデータをデータタイプ IEC_TIMER または TON_TIME の構造体に保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TON_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「オンディレータイマの起動」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体にデータを保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TON_TIME、TON_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および指定された IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは、IEC タイマの構造体コンポーネント ET に保存されます。 タイマステータスは、バイナリ論理演算を使って照会できます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「オンディレータイマの起動」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「オンディレータイマの起動」命令のパラメータを示します。

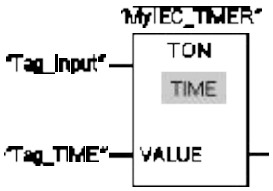
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	IEC タイマが動作する時間。

<IEC timer>	InOut	IEC_TIMER, TON_TIME	IEC_TIMER, IEC_LTIMER, TON_TIME, TON_LTIME	D、L	開始される IEC タイマ。

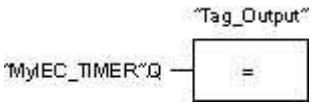
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「オンディレータイマの起動」命令は、オペランド「Tag_Input」のシグナル状態が「0」から「1」に切り替わると実行されます。 MyIEC_TIMER タイマは、オペランド「Tag_TIME」内に格納された時間で開始されます。



タイマ「MyIEC_TIMER」の期限が切れ、オペランド「Tag_Input」のシグナル状態が「1」の場合、タイマステータス("MyIEC_TIMER".Q を照会すると、シグナル状態「1」が返され、オペランド「Tag_Output」がセットされます。 オペランド「Tag_Input」がシグナル状態「0」に切り替わった場合、タイマステータスを照会するとシグナル状態「0」が返され、オペランド「Tag_Output」がリセットされます。

TOF: オフディレータイマの起動

説明

「オフディレータイマの起動」命令を使用して、オフディレーとして指定された持続時間分、IEC タイマを起動します。 命令の入力での論理演算の結果(RLO)のシグナル状態が「1」の場合、「1」のタイマステータスの照会は、シグナル状態「0」を返します。 RLO が「1」から「0」(立ち下がり工ッジ)に切り替わると、指定された持続時間の IEC タイマが起動します。 IEC タイマが実行している限り、タイマステータスのシグナル状態は「1」のままです。 タイマの期限が切れ、命令の入力の RLO のシグナル状態が「0」の場合、タイマステータスはシグナル状態「0」にセットされます。 タイマの期限が切れる前に RLO が「1」に切り替わった場合、動作中の IEC タイマがリセットされ、タイマステータスがシグナル状態「1」のママになります。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、IEC タイマの開始と照会の期限切れのレベルが異なる可能性があります。

S7-1200 CPU の場合

「オンディレータイマの起動」命令がそのデータをデータタイプ IEC_TIMER または TOF_TIME の構造体に保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TOF_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「オフディレータイマの起動」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体にデータを保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TOF_TIME、TOF_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および指定された IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは、IEC タイマの構造体コンポーネント ET に保存されます。 タイマステータスは、バイナリ論理演算を使って照会できます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「オフディレータイマの起動」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワークの終端にのみ配置できます。

パラメータ

次の表に、「オフディレータイマの起動」命令のパラメータを示します。

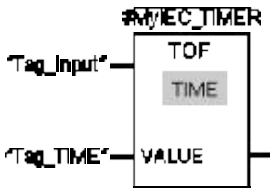
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	IEC タイマが動作する時間。

<IEC timer>	InOut	IEC_TIMER, TOF_TIME	IEC_TIMER, IEC_LTI-MER, TOF_TIME, TOF_LTIME	D、L	開始される IEC タイマ。

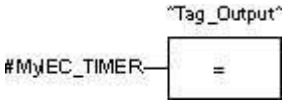
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「オフディレータイマの起動」命令は、オペランド「Tag_Input」のシグナル状態が「1」から「0」に切り替わると実行されます。 #MyIEC_TIMER タイマは、オペランド「Tag_TIME」内に格納された時間で開始されます。



タイマ#MyIEC_TIMER が作動している限り、時間のステータス(#MyIEC_TIMER.Q)を照会するとシグナル状態「1」が返され、オペランド「Tag_Output」が設定されます。 タイマの期限が切れ、オペランド「Tag_Input」がシグナル状態「0」になった場合、タイマステータスを照会するとシグナル状態「0」が返されます。 オペランド「Tag_Input」のシグナル状態が「1」に切り替わってからタイマ #MyIEC_TIMER の期限が切れる場合、タイマはリセットされます。 オペランド「Tag_Input」のシグナル状態が「1」の場合、タイマステータスを照会するとシグナル状態「1」が返されます。

TONR: タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用し、命令「1」の入力時の信号の長さを記録することが可能です。 この命令は、入力での論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がり工ッジ)に開始されます。 時間は、RLO が「1」である限り記録されます。 RLO が「0」に切り替わると、この命令は中断します。 RLO が「1」に戻ると、時間の記録が継続されます。 記録された時間が指定された持続時間値を超え、コイルの入力での RLO が「1」の場合、「1」のタイマステータスを照会するとシグナル状態「1」が返されます。

「タイマのリセット」命令を使って、タイマステータスおよび現在期限切れのタイマを「0」にリセットできます。

注記
出力 Q または ET を照会するたびに IEC_TIMER 構造体が更新されるため、IEC タイマの開始と照会の期限切れのレベルが異なる可能性があります。

S7-1200 CPU の場合

「タイムアキュムレータ」命令がそのデータをデータタイプ IEC_TIMER または TONR_TIME の構造体に保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TONR_TIME」または「IEC_TIMER」のローカルタグとしての宣言(#MyIEC_TIMER など)

S7-1500 CPU の場合

「タイムアキュムレータ」命令は、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体にデータを保存します。 構造体は、次のように宣言できます。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TONR_TIME、TONR_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

命令のデータは命令が呼び出される時、および指定された IEC タイマがアクセスされるたびに更新されます。

現在のタイマステータスは、IEC タイマの構造体コンポーネント ET に保存されます。 タイマステータスは、バイナリ論理演算を使って照会できます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

「タイムアキュムレータ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワークの終端にのみ配置できます。

パラメータ

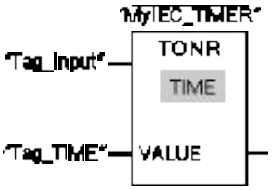
次の表に、「タイムアキュムレータ」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
VALUE	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	IEC タイマが動作する時間。
<IEC timer>	InOut	IEC_TIMER, TONR_TIME	IEC_TIMER, IEC_LTIMER, TONR_TIME, TONR_LTIME	D、L	開始される IEC タイマ。

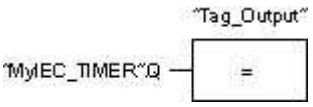
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



RLO に立ち上がりエッジがある場合、「タイムアキュムレータ」命令が実行されます。 オペランド「Tag_Input」のシグナル状態が「1」である限り、時間が記録されます。



記録された時間がオペランド「Tag_TIME」の値を超えている場合、タイマステータス("MyIEC_TIMER".Q)を照会するとシグナル状態「1」が返され、オペランド「Tag_Output」がセットされます。

RT: タイマのリセット

説明

「タイマのリセット」命令を使用し、IEC タイマを「0」にリセットすることが可能です。 リセットする IEC タイマは、IEC タイマの構造体を含むデータブロックの名前を命令の上にあるプレースホルダに入力して指定します。

この命令は、ボックス入力の論理演算の結果(RLO)が「1」の場合のみ実行されます。 この命令が実行されると、指定したデータブロックで IEC タイマの構造体コンポーネントが「0」にリセットされます。ボックス入力の RLO が「0」の場合、この命令は実行されません。

この命令は、RLO に影響を与えません。 ボックス入力の RLO は、ボックス出力に直接転送されます。

プログラムで宣言された IEC タイマを「タイマのリセット」命令に割り当てる必要があります。

命令のデータは命令が呼び出されるときのみ更新され、割り当てられた IEC タイマがアクセスされるたびには更新されません。 データの照会は、命令の呼び出しから命令の次の呼び出しまでのみ同一になります。

パラメータ

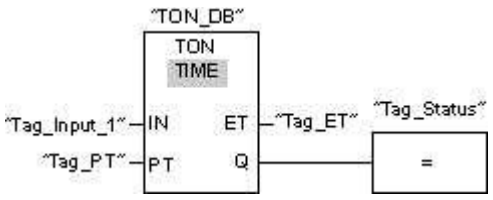
次の表に、「タイマのリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<IEC timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	リセットされる IEC タイマ。

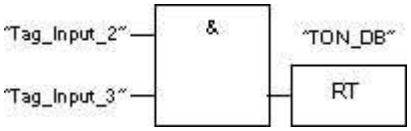
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「オンディレー生成」命令は、オペランド「Tag_Input_1」のシグナル状態が「0」から「1」に切り替わると実行されます。 インスタンスデータブロック「TON_DB」に保存された IEC タイマは、オペランド「Tag_PT」で指定された持続時間で起動します。



オペランド「Tag_Input_2」および「Tag_Input_3」のシグナル状態が「1」の場合、「タイマのリセット」命令が実行され、データブロック「TON_DB」に格納された IEC タイマがリセットされます。

PT: 持続時間のロード

説明

「時間の長さのロード」命令を使って、IEC タイマの持続時間を設定します。 命令の入力において、論理演算の結果(RLO)のシグナル状態が「1」の場合、サイクルごとにこの命令を実行します。 この命令は、指定された IEC タイマの構造体に、指定された持続時間を書き込みます。

注記
実行中に、指定した IEC タイマが作動している場合、この命令が指定した IEC タイマの現在の持続時間を上書きします。 その結果、IEC タイマのタイマステータスが変わる場合があります。

プログラムで宣言された IEC タイマを「時間の長さのロード」命令に割り当てる必要があります。

命令のデータは命令が呼び出されるとき、および割り当てられた IEC タイマがアクセスされるたびに更新されます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

パラメータ

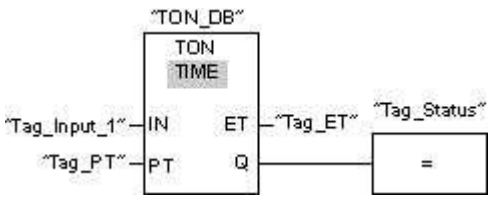
次の表に、「時間の長さのロード」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、または定数	持続時間
<IEC timer>	InOut	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	持続時間を設定する IEC タイマ

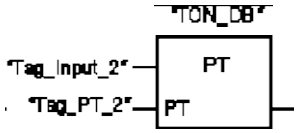
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「オンディレー生成」命令は、オペランド「Tag_Input_1」のシグナル状態が「0」から「1」に切り替わると実行されます。 インスタンスデータブロック「TON_DB」に保存された IEC タイマは、オペランド「Tag_PT」で指定された持続時間で起動します。



「時間の長さのロード」命令は、オペランド「Tag_Input_2」のシグナル状態が「1」の場合に実行されます。 この命令は、インスタンスデータブロック「TON_DB」に持続時間「Tag_PT_2」を書き込み、同時にデータブロック内のオペランド「Tag_PT」の値を上書きします。 結果的にタイマステータスのシグナル状態は、次の照会時、または "MyTimer".Q または "MyTimer".ET へのアクセス時に変化する可能性があります。

注記

Tag_Input_2 は、その持続時間が 1 プログラムサイクルのみで読み込まれるように、パルスフラグとして実行されます。

レガシー

この章には下記に関する情報が記載されています：

S_PULSE: パルスタイマパラメータの割り当てと起動

説明

「パルスタイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。入力 S のシグナル状態が「1」の場合は、タイマでプログラミングされた時間(TV)の期限が切れます。 プログラミングされた時間の期限が切れる前に、入力 S がシグナル状態「0」に切り替わった場合、タイマが停止します。 この場合、出力 Q のシグナル状態は「0」です。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。 現在の時間値は、出力 BI でバイナリコード化され、出力 BCD で BCD コード化されます。

タイマが動作していて、かつ入力 R のシグナル状態が「1」に切り替わった場合、現在の時間値およびタイムベースも 0 にセットされます。 タイマが動作していない場合、R 入力のシグナル状態が「1」であっても効果はありません。

「パルスタイマパラメータの割り当てと開始」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

命令データはアクセスのたびに更新されます。 このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「パルスタイマパラメータの割り当てと開始」命令のパラメータを示します。

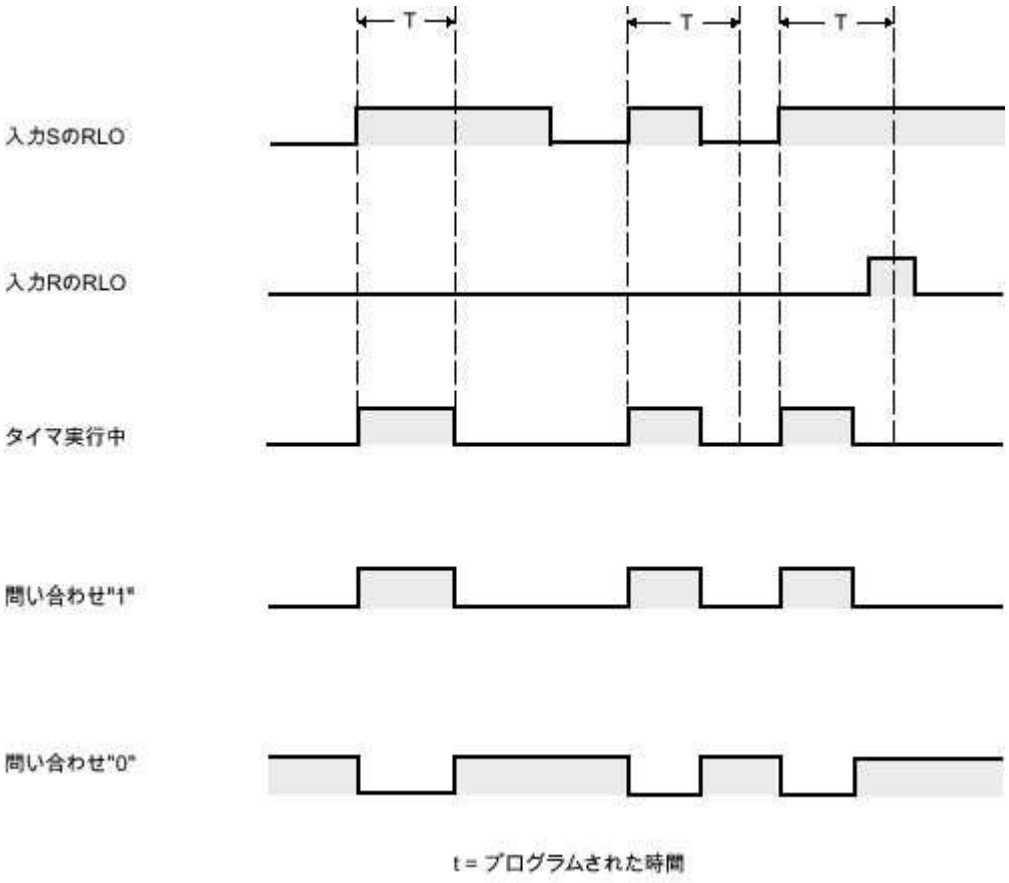
パラメータ	宣言	データタイプ	メモリ領域	説明
<timer>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(バイナリコード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値(BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

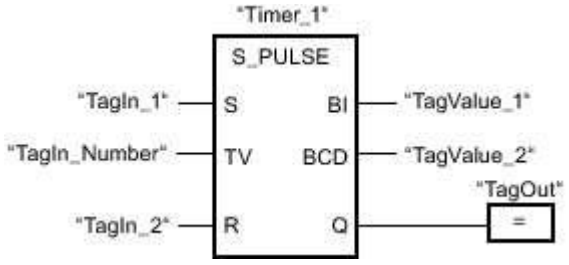
パルスタイミング図

次の図に、「パルスタイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。「TagIn_1」オペランドのシグナル状態「1」である限り、「TagIn_Number」オペランドの時間値の間、タイマが実行されます。 オペランド「TagIn_1」のシグナル状態が、タイマの期限が切れる前

に「1」から「0」に切り替わった場合、タイマ「Timer_1」が停止します。 オペランド「TagOut」は、シグナル状態「0」にリセットされます。

タイマが実行中でオペランド「TagIn_1」のシグナル状態が「1」である間は、オペランド「TagOut」のシグナル状態は「1」です。 時間の期限が切れるかリセットされた場合、オペランド「TagOut」が「0」にリセットされます。

S_PEXT: 拡張パルスタイマパラメータを割り当てと起動

説明

「拡張パルスタイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S がシグナル状態「0」に切り替わっても、タイマでプログラムされた持続時間(TV)の期限が切れます。 タイマが実行中である限り、出力 Q のシグナル状態は「1」です。 タイマの時間が経過すると、出力 Q が「0」にリセットされます。 タイマの実行中に入力 S のシグナル状態が「0」から「1」に切り替わると、タイマは入力 TV でプログラミングされた時間によって再開されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。 現在の時間値は、出力 BI でバイナリコード化され、出力 BCD で BCD コード化されます。

タイマが動作していて、かつ入力 R のシグナル状態が「1」に切り替わった場合、現在の時間値およびタイムベースも 0 にセットされます。 タイマが動作していない場合、R 入力のシグナル状態が「1」であっても効果はありません。

「拡張パルスタイマパラメータの割り当てと開始」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

命令データはアクセスのたびに更新されます。 このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「拡張パルスタイマパラメータの割り当てと開始」命令のパラメータを示します。

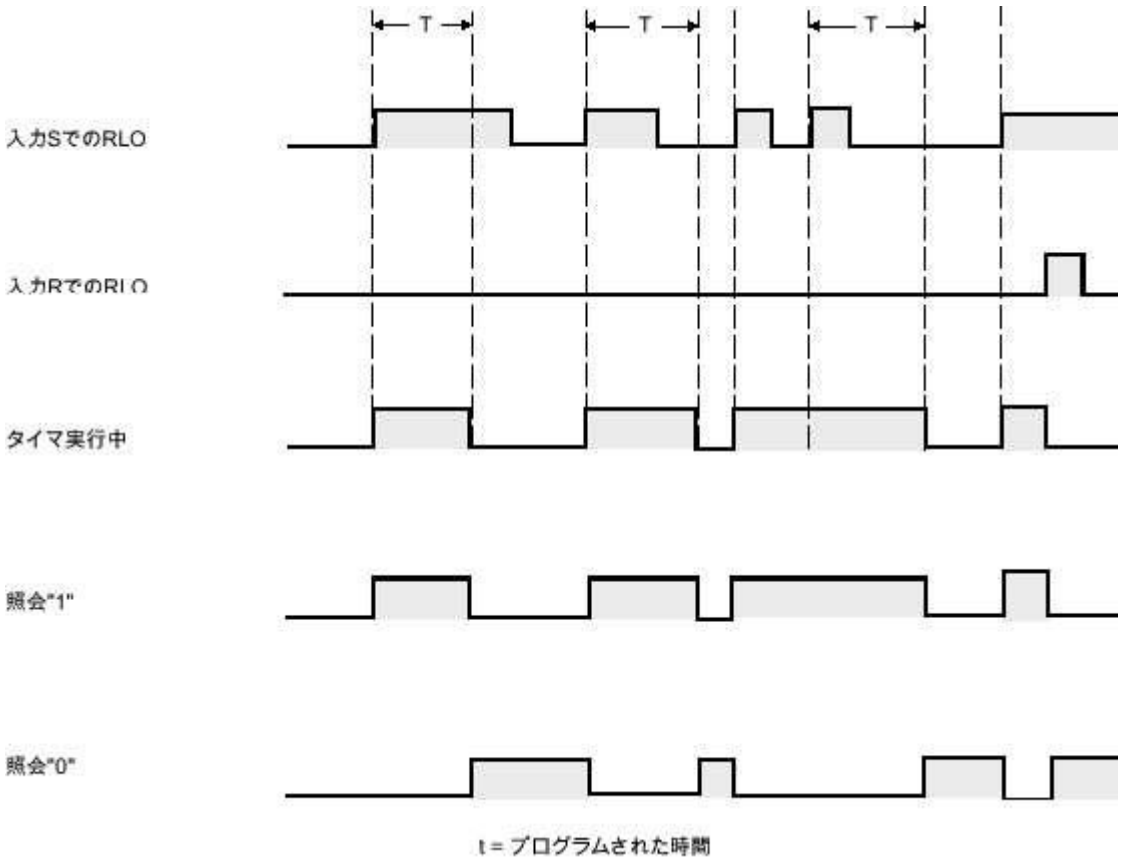
パラメータ	宣言	データタイプ	メモリ領域	説明
<timer>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(バイナリコード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値(BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

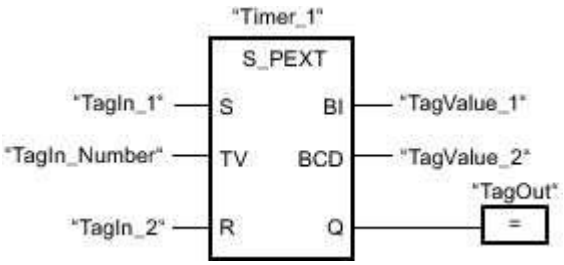
パルスタイミング図

次の図に、「拡張パルスタイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 「TagIn_Number」オペランドの時間値の間タイマが実行されます。S 入力の立ち下がりエッジには影響されません。 タイマの期限が切れる前にオペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると、タイマが再起動します。

タイマの実行中は、オペランド「TagOut」のシグナル状態は「1」です。 時間の期限が切れるかリセットされた場合、オペランド「TagOut」が「0」にリセットされます。

S_ODT: オンディレータイマパラメータの割り当てと開始 

説明

「オンディレータイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S のシグナル状態が「1」の場合は、タイマでプログラミングされた時間(TV)の期限が切れます。 タイマの時間が正常に経過し、さらに入力 S のシグナル状態が「1」のままである場合、出力 Q がシグナル状態「1」を返します。 タイマの実行中に入力 S のシグナル状態が「1」から「0」に切り替わると、タイマは停止します。 この場合、出力 Q がシグナル状態「0」にリセットされます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。 現在の時間値は、出力 BI でバイナリコード化され、出力 BCD で BCD コード化されます。

タイマが動作していて、かつ入力 R のシグナル状態が「0」から「1」に切り替わった場合、現在の時間値およびタイムベースも 0 にセットされます。 この場合、出力 Q のシグナル状態は「0」です。 タイマが実行中でなく、入力 S の RLO が「1」の場合でも、R 入力のシグナル状態が「1」の場合、タイマがリセットされます。

「オンディレータイマパラメータの割り当てと開始」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

命令データはアクセスのたびに更新されます。 このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「オンディレータイマパラメータの割り当てと開始」命令のパラメータを示します。

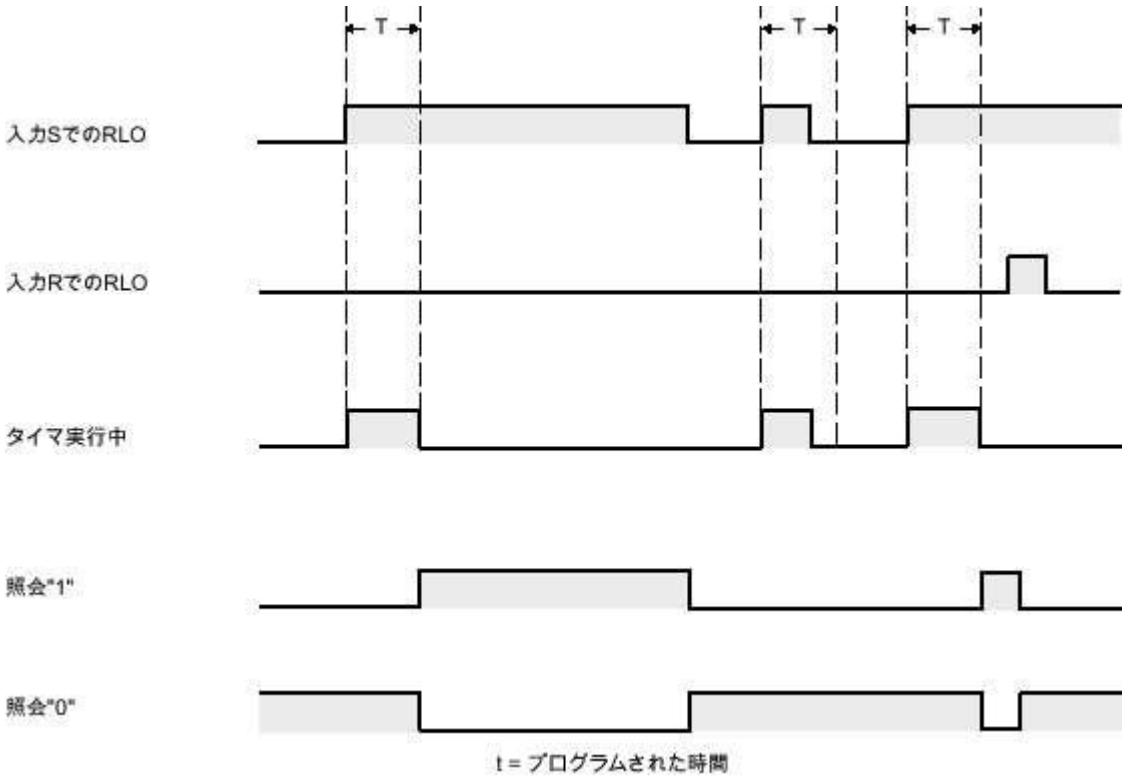
パラメータ	宣言	データタイプ	メモリ領域	説明
<timer>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(バイナリコード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値(BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

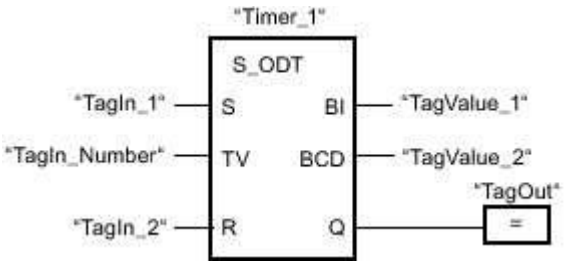
パルスタイミング図

次の図に、「オンディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 オペランド「TagIn_Number」の値でタイマの期限が切れます。 タイマの期限が切れてオペランドのシグナル状態が「1」の場合、「TagOut」オペランドが「1」にセットされます。 タイマの期限が切れる前にオペランド「TagIn_1」のシグナル状態が「1」から「0」に切り替わると、タイマは停止します。 オペランド「TagOut」のシグナル状態は「0」です。

S_ODTS: 保持型オンディレータイマパラメータの割り当てと開始

説明

「保持型オンディレータイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 入力 S がシグナル状態「0」に切り替わっても、タイマでプログラムされた持続時間(TV)の期限が切れます。 タイマの期限が切れた場合、「Q」出力は、入力「S」でのシグナル状態に関わらずシグナル状態「1」を返します。 タイマの実行中に入力 S のシグナル状態が「0」から「1」に切り替わると、タイマは入力(TV)でプログラミングされた時間によって再開されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。 現在の時間値は、出力 BI でバイナリコード化され、出力 BCD で BCD コード化されます。

入力 R のシグナル状態「1」は、開始入力 S でのシグナル状態に関わらず、現在の時間値およびタイムベースを「0」にリセットします。 この場合、出力 Q のシグナル状態は「0」です。

「保持型オンディレータイマパラメータの割り当てと開始」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

命令データはアクセスのたびに更新されます。 このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「保持型オンディレータイマパラメータの割り当てと開始」命令のパラメータを示します。

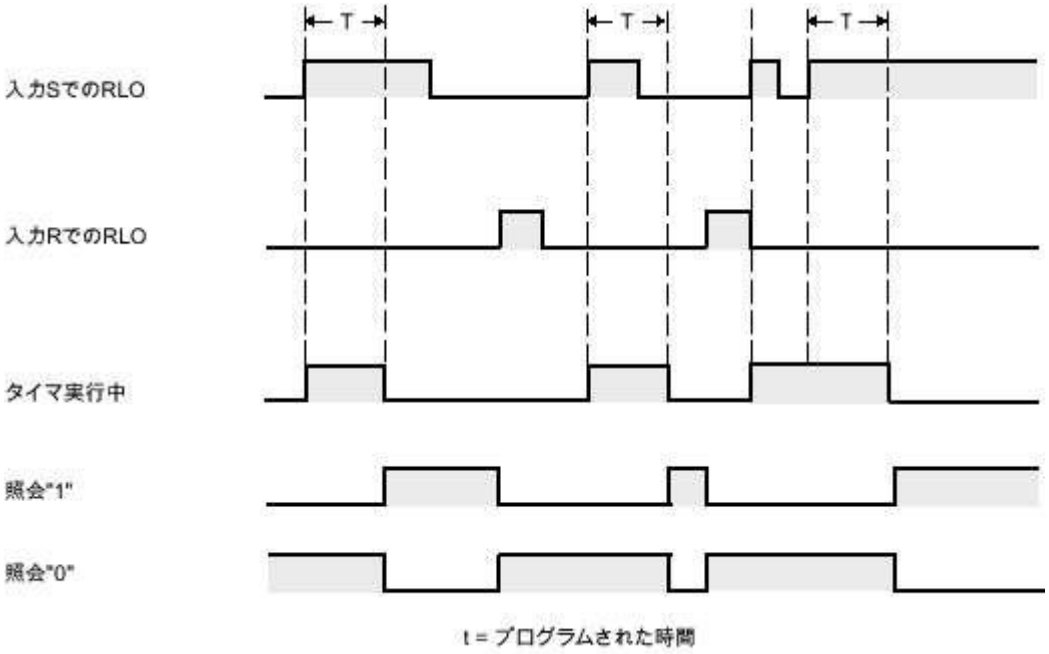
パラメータ	宣言	データタイプ	メモリ領域	説明
<timer>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(バイナリコード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値(BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

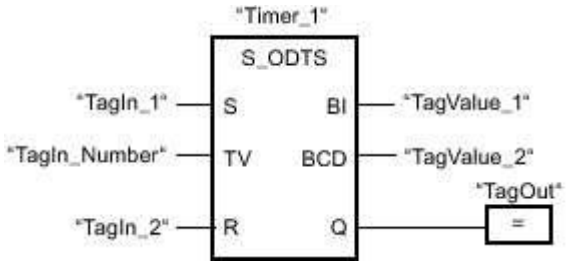
パルスタイミング図

次の図に、「保持型オンディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 タイマは、「TagIn_1」オペランドのシグナル状態が「0」になっても、「TagIn_Number」オペランドの時間値で期限切れになります。 時間の期限が切れると、オペランド「TagOut」が「1」にリセットされます。 タイマの実行中にオペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると、タイマが再起動します。

S_OFFDT: オフディレータイマパラメータの割り当てと開始

説明

「オフディレータイマパラメータの割り当てと開始」命令は、入力 S の論理演算(RLO)の結果で「1」から「0」への切り替え(信号立ち下がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、プログラムされた時間(TV)によって期限が切れます。 タイマが動作している場合、または入力 S がシグナル状態「1」を返す場合は、入力 Q のシグナル状態は「1」となります。 タイマの時間が経過し、かつシグナル状態が「0」の場合は、出力 Q がシグナル状態「0」にリセットされます。 タイマの実行中に入力 S のシグナル状態が「0」から「1」に切り替わると、タイマは停止します。 タイマは、入力 S で信号の立ち下がりエッジが検出された後にのみ再開されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。 現在の時間値は、出力 BI でバイナリコード化され、出力 BCD で BCD コード化されます。

入力 R でのシグナル状態「1」は、現在の時間値およびタイムベースを「0」にリセットします。 この場合、出力 Q のシグナル状態は「0」です。

「オフディレータイマパラメータの割り当てと開始」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

命令データはアクセスのたびに更新されます。 このため、サイクルの最初のデータのクエリは、サイクルの最後のデータと異なる値を返す可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

次の表に、「オフディレータイマパラメータの割り当てと開始」命令のパラメータを示します。

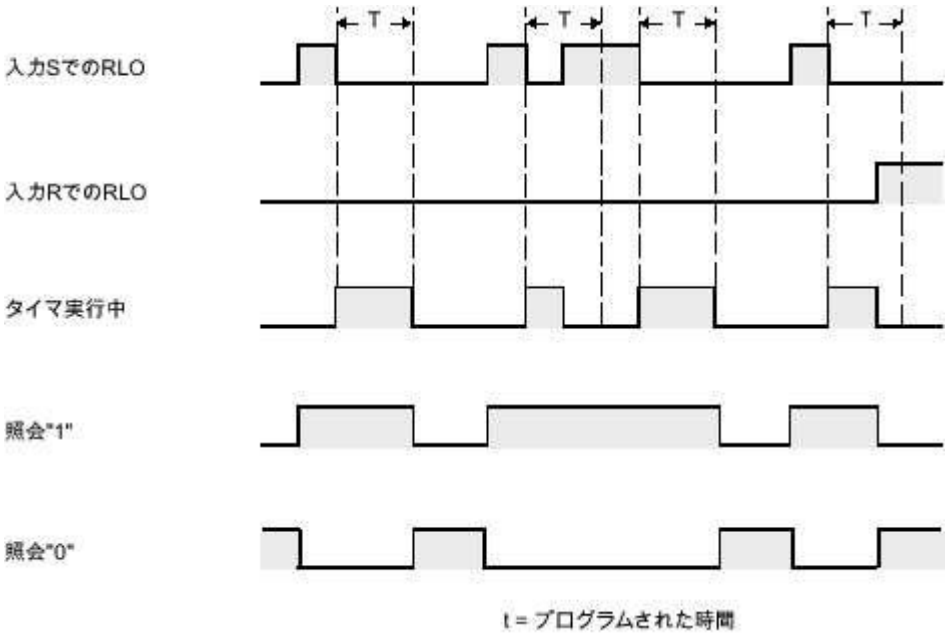
パラメータ	宣言	データタイプ	メモリ領域	説明
<timer>	InOut/Input	TIMER	T	命令の時間 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
R	Input	BOOL	I、Q、M、T、C、D、L、P、または定数	リセット入力
BI	Output	WORD	I、Q、M、D、L、P	現在の時間値(バイナリコード)

BCD	Output	WORD	I、Q、M、D、L、P	現在の時間値(BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

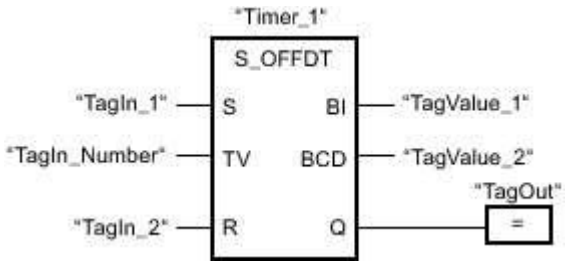
パルスタイミング図

次の図に、「オフディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「1」から「0」に切り替わると「Timer_1」が開始します。 オペランド「TagIn_Number」の値でタイマの期限が切れます。 タイマが実行中であるかオペランド「TagIn_1」のシグナル状態が「0」の場合、オペランド「TagOut」が「1」にセットされます。 タイマの実行中にオペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると、タイマがリセットされます。

SP: パルスタイマの起動

説明

「パルスタイマの起動」命令は、初期入力における論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、RLO のシグナル状態が「1」である限り、指定された時間の間実行されます。 タイマが実行されている限り、タイマステータス「1」を照会するとシグナル状態「1」が返されます。 時間値の期限が切れる前に、RLO で「1」から「0」への切り替えがあった場合、タイマが停止します。 この場合、タイマステータス「1」を照会するとシグナル状態「0」が返されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。

「パルスタイマの起動」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

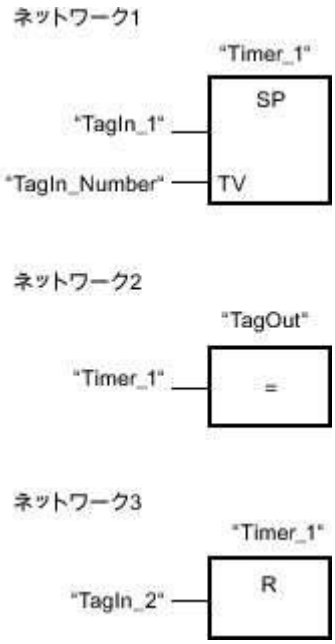
次の表に、「パルスタイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
<timer>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 タイマは、「TagIn_1」オペランドのシグナル状態が「1」である限り、「TagIn_Number」オペランドの時間値で期限切れになります。 タイマの期限が切れる前にオペランド「TagIn_1」のシグナル状態が「1」から「0」に切り替わると、タイマは停止します。 タイマの実行中は、オペランド「TagOut」がシグナル状態「1」を返します。 「TagIn_1」オペランドのシグナル状態が「0」から「1」に切り替わると、タイマがリセットされます。つまり、タイマが停止し、現在の時間値が「0」にセットされます。

SE: 拡張パルスタイマの起動

説明

「拡張パルスタイマの起動」命令は、初期入力における論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がり工ッジ)が検出されると、プログラミングされたタイマを起動します。 RLO がシグナル状態「0」に変わっても、タイマが指定された時間実行されます。 タイマが実行されている限り、タイマステータス「1」を照会するとシグナル状態「1」が返されます。 タイマの実行中に RLO が「0」から「1」に切り替わると、プログラムされた時間でタイマが再開されます。 タイマの期限が切れた場合、タイマステータス「1」をクエリするとシグナル状態「0」が返されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。

「拡張パルスタイマの起動」命令には、前に工ッジ評価のための論理演算が必要で、ネットワークの右側工ッジのみに配置することができます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

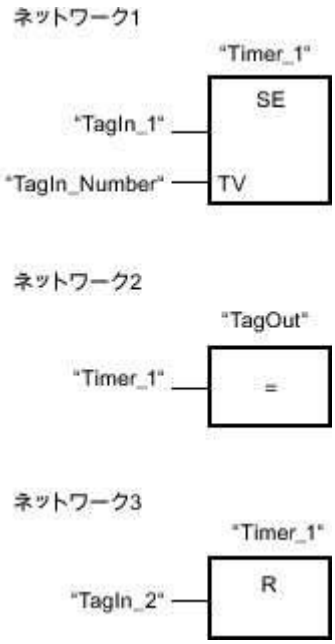
次の表に、「拡張パルスタイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
<timer>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 タイマは、「TagIn_Number」オペランドの時間値で期限が切れます。RLO の立ち下がりエッジに影響されません。 タイマの実行中は、オペランド「TagOut」がシグナル状態「1」を返します。 タイマの期限が切れる前にオペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると、タイマが再起動します。

SD: オンディレータイマの起動

説明

「オンディレータイマの起動」命令は、初期入力における論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 タイマは、RLO が「1」である限り指定された時間実行されます。 タイマの期限が切れ、RLO のシグナル状態が「1」になった場合、タイマステータス「1」を照会するとシグナル状態「1」が返されます。 タイマの実行中に RLO が「1」から「0」に切り替わった場合、タイマが停止します。 この場合、タイマステータス「1」を照会するとシグナル状態「0」が返されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。

「オンディレータイマの起動」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

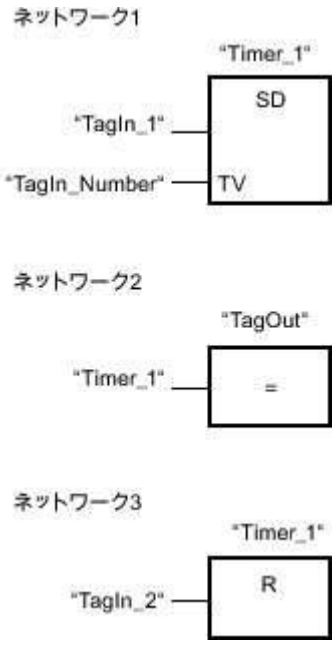
次の表に、「オンディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
<timer>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 オペランド「TagIn_Number」の値でタイマの期限が切れます。 タイマの期限が切れて RLO のシグナル状態が「1」の場合、「TagOut」オペランドが「1」にセットされます。 タイマの期限が切れる前にオペランド「TagIn_1」のシグナル状態が「1」から「0」に切り替わると、タイマは停止します。「TagIn_2」オペランドのシグナル状態が「1」に切り替わると、「Timer_1」がリセットされます。つまり、タイマが停止し、現在の時間値が「0」にセットされます。

SS: 拡張オンディレータイマの起動

説明

「保持型オンディレータイマの起動」命令は、初期入力における論理演算(RLO)の結果で「0」から「1」への切り替え(信号立ち上がりエッジ)が検出されると、プログラミングされたタイマを起動します。 RLO がシグナル状態「0」に変わっても、タイマが指定された時間実行されます。 タイマの期限が切れた場合、タイマステータス「1」をクエリするとシグナル状態「1」が返されます。 タイマの期限が切れた場合、タイマは明示的にリセットした場合のみ再起動できます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。 この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。

「保持型オンディレータイマの起動」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

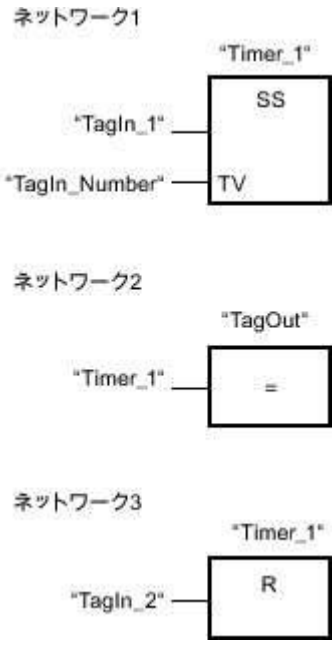
次の表に、「保持型オンディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
<timer>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると「Timer_1」が開始します。 オペランド「TagIn_Number」の値でタイマの期限が切れます。 時間の期限が切れると、オペランド「TagOut」が「1」にリセットされます。 タイマの実行中にオペランド「TagIn_1」のシグナル状態が「0」から「1」に切り替わると、タイマが再起動します。「TagIn_2」オペランドのシグナル状態が「1」に切り替わると、「Timer_1」がリセットされます。つまり、タイマが停止し、現在の時間値が「0」にセットされます。

SF:オフディレータイマの起動

説明

「オフディレータイマの起動」命令は、初期入力おける論理演算(RLO)の結果で「1」から「0」への切り替え(信号立ち下がりエッジ)が検出されると、プログラミングされたタイマを起動します。 IEC タイマが、指定された持続時間の間、作動します。 タイマが実行されている限り、タイマステータス「1」を照会するとシグナル状態「1」が返されます。 タイマの実行中に RLO が「0」から「1」に切り替わった場合、タイマが停止します。 RLO が「1」から「0」に切り替わると、タイマは必ず再起動されます。

この時間は、内部で時間値とタイムベースから構成され、パラメータ TV でプログラミングされます。この命令が開始されると、プログラムされた時間値がゼロの方向にカウントダウンされます。 タイムベースは、時間値の時間を決定します。

「オフディレータイマの起動」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

パラメータ

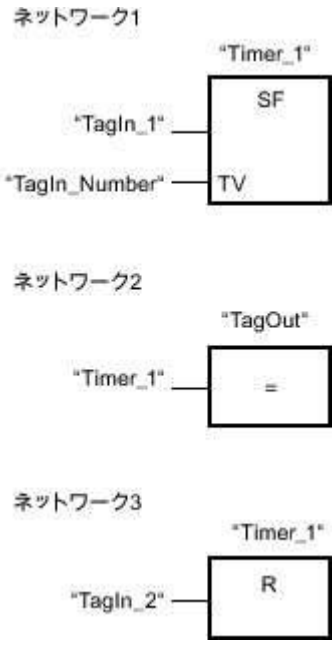
次の表に、「オフディレータイマの起動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、T、C、D、L、P	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L、または定数	持続時間
<timer>	InOut/Input	TIMER	T	開始されるタイマ タイマの数は CPU によって異なります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



"オペランド「TagIn_1」のシグナル状態が「1」から「0」に切り替わると「Timer_1」が開始します。 オペランド「TagIn_Number」の値でタイマの期限が切れます。 タイマの実行中は、オペランド「TagOut」が「1」にセットされます。 タイマの実行中にオペランド「TagIn_1」のシグナル状態が「1」から「0」に切り替わると、タイマが再起動します。「TagIn_2」オペランドのシグナル状態が「1」に切り替わると、「Timer_1」がリセットされます。つまり、タイマが停止し、現在の時間値が「0」にセットされます。

カウンタ演算

この章には下記に関する情報が記載されています：

- CTU: カウントアップ (S7-1200, S7-1500)
- CTD: カウントダウン (S7-1200, S7-1500)
- CTUD: カウントアップ/カウントダウン (S7-1200, S7-1500)
- レガシー (S7-1500)

CTU: カウントアップ

説明

出力 CV の値のインクリメントには、「カウントアップ」命令を使用できます。 CU 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、この命令が実行され、CV 出力のカウンタ現在値が 1 ずつインクリメントされます。 立ち上がりエッジを検出するたびに、CV 出力において指定されたデータタイプの上限値に達するまで、カウンタ値がインクリメントされます。 上限値に達すると、CU 入力のシグナル状態はこの命令に影響を与えなくなります。

Q 出力でカウンタステータスを照会することができます。 Q 出力のシグナル状態は、PV パラメータによって決まります。 カウンタ現在値が PV パラメータ値以上の場合、Q 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、Q 出力のシグナル状態は「0」になります。 PV パラメータには定数も指定できます。

入力 R のシグナル状態が「1」に切り替わると、CV 出力の値が「0」にリセットされ、エッジメモリビットに保存されます。R 入力のシグナル状態が「1」である限り、CU 入力のシグナル状態は命令に影響しません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER

	• IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTU_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyIEC_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。 インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントアップ」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

次の表に、「カウントアップ」命令のパラメータを示します。

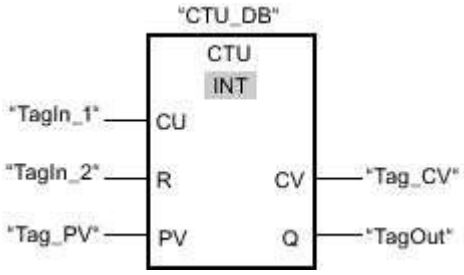
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
CU	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウント入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	リセット入力
PV	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	出力 Q がセットされる値。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	I、Q、M、D、L、P	カウンタ現在値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「0」から「1」に切り替わると、「カウントアップ」命令が実行され、「Tag_CV」カウンタ現在値が 1 インクリメントされます。 信号立ち上がり エッジが検出されるたびにカウント値がインクリメントされ、指定されたデータタイプの上限値(INT = 32767)に達するまでインクリメントされます。

PV パラメータの値が「TagOut」出力を確定する限度として採用されます。 カウンタ現在値がオペランド「Tag_PV」の値以上の場合、「TagOut」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut」出力はシグナル状態「0」を返します。

CTD: カウントダウン

説明

「カウントダウン」命令を使用して、出力 CV の値をデクリメントします。 CD 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、この命令が実行され、CV 出力のカウンタ現在値が 1 ずつデクリメントされます。 指定されたデータタイプの下限値に達するまで、信号の立ち上がりエッジが検出されるたびにカウンタがデクリメントされます。 下限値に達すると、CD 入力のシグナル状態はこの命令に影響を与えなくなります。

Q 出力でカウンタステータスを照会することができます。 カウンタ現在値が「0」以下である場合、Q 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、Q 出力のシグナル状態は「0」になります。 PV パラメータには定数も指定できます。

LD 入力のシグナル状態が「0」から「1」になると、CV 出力の値が PV パラメータの値に設定され、エッジメモリビットに保存されます。LD 入力のシグナル状態が「1」である限り、CD 入力のシグナル状態は命令に影響しません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

	• IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyIEC_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。 インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントダウン」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

次の表に、「カウントダウン」命令のパラメータを示します。

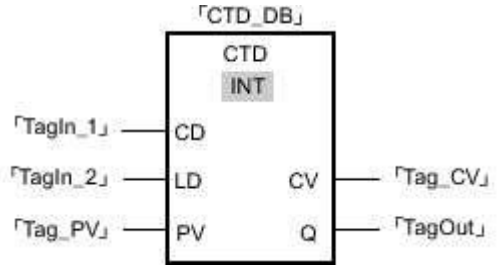
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
CD	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウント入力
LD	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	CV 出力が LD = 1 にセットされる値。
Q	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	I、Q、M、D、L、P	カウンタ現在値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CV」出力の値が 1 デクリメントされます。 それ以降、信号立ち上がり工ッジが検出されるたびに、指定したデータタイプの下限値(INT = -32768)に達するまで、カウンタ値がデクリメントされます。

PV パラメータの値が「TagOut」出力を確定する限度として採用されます。 カウンタ現在値が「0」以下である限り、「TagOut」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut」出力はシグナル状態「0」を返します。

CTUD: カウントアップ/カウントダウン

説明

「カウントアップ/カウントダウン」命令を使用して、CV 出力のカウンタ値のインクリメント、およびデクリメントが可能です。 CU 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、カウンタ現在値が 1 インクリメントされ、CV 出力に格納されます。 CD 入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、CV 出力のカウンタ現在値が 1 デクリメントされます。 1 つのプログラムサイクル内で CU および CD 入力に立ち上がりエッジが発生した場合、CV 出力のカウンタ現在値は変わりません。

カウンタ値は、CV 出力で指定したデータタイプの上限値に達するまでインクリメントすることができます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。カウンタ値は、指定されたデータタイプの下限値に達するとデクリメントされなくなります。

LD 入力のシグナル状態が「1」に切り替わると、CV 出力のカウンタ値が PV パラメータの値に設定され、エッジメモリビットに格納されます。LD 入力のシグナル状態が「1」である限り、CU および CD 入力のシグナル状態は命令に影響しません。

入力 R のシグナル状態が「1」に切り替わると、カウンタ値が「0」にセットされ、エッジメモリビットに格納されます。R 入力のシグナル状態が「1」である限り、CU、CD、および LD 入力のシグナル状態が切り替わっても「カウントアップ/カウントダウン」命令には影響を与えません。

QU 出力でアップカウンタのステータスを照会することができます。 カウンタ現在値が PV パラメータ値以上の場合、QU 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、QU 出力のシグナル状態は「0」になります。 PV パラメータには定数も指定できます。

QD 出力でダウンカウンタのステータスを照会することができます。 カウンタ現在値がゼロ以下の場合、QD 出力がシグナル状態「1」にセットされます。 それ以外のすべての場合、QD 出力のシグナル状態は「0」になります。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ/カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT • IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTUD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyIEC_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントアップ/カウントダウン」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

パラメータ

次の表に、「カウントアップ/カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
CU	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	カウントダウン入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	リセット入力

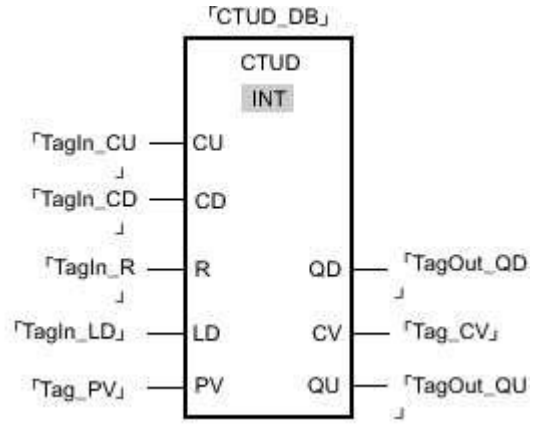
LD	Input	BOOL	I、Q、M、D、L、P、または定数	I、Q、M、T、C、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	出力 QU がセットされる値。 /CV 出力が LD = 1 にセットされる値。
QU	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウントアップのステータス
QD	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	カウントダウンのステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	I、Q、M、D、L、P	カウンタ現在値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_CU」または「TagIn_CD」入力のシグナル状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、「カウントアップ/カウントダウン」命令が実行されます。 「TagIn_CU」入力で立ち上がりエッジが発生する場合、カウンタ現在値は 1 ずつインクリメントされ、「Tag_CV」出力に保存されます。 「TagIn_CD」入力で立ち上がりエッジが発生する場合、カウンタ値は 1 ずつデクリメントされ、「Tag_CV」出力に保存されます。 CU 入力で立ち上がりエッジが発生した場合、上限値(INT = 32767)に達するまでカウンタ値がインクリメントされます。 入力 CD で信号立ち上がりエッジが発生した場合、下限値(INT = -32768)に達するまでカウンタ値がデクリメントされます。

カウンタ現在値が「Tag_PV」入力の値以上である限り、「TagOut_GU」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut_QU」出力はシグナル状態「0」を返します。

カウンタ現在値が「0」以下である限り、「TagOut_QD」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「TagOut_QD」出力のシグナル状態は「0」になります。

レガシー

この章には下記に関する情報が記載されています：

- S_CU:パラメータおよびカウントアップの割り当て (S7-1500)
- S_CD:パラメータおよびカウントダウンの割り当て (S7-1500)
- S_CUD:パラメータおよびカウントアップ/カウントダウンの割り当て (S7-1500)
- SC: カウンタ値の設定 (S7-1500)
- CU: カウントアップ (S7-1500)
- CD: カウントダウン (S7-1500)

S_CU:パラメータおよびカウントアップの割り当て

説明

「パラメータおよびカウントアップの割り当て」命令を使用して、カウンタ値のインクリメントが可能です。 CU 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 インクリメントされます。 現在のカウンタ値は、出力 CV に 16 進数値で出力され、出力 CV_BCD で BCD コード化されます。 カウンタ値は、上限値の「999」に達するまでインクリメントされます。 制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。

入力 S の信号状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタが設定され、入力 CU の RLO が「1」の場合、信号エッジで切り替えが検出されていない場合でも、カウンタはその次のスキャンサイクルで適宜カウントします。

R 入力の信号状態が「1」に切り替わると、カウンタ値がゼロに設定されます。 R 入力の信号状態が「1」である限り、CU および S 入力の信号状態の処理はこのカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、出力 Q の信号状態は「1」となります。 カウンタ値がゼロに等しい場合、出力 Q の信号状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「パラメータおよびカウントアップの割り当て」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

パラメータ

次の表に、「パラメータおよびカウントアップの割り当て」命令のパラメータを示します。

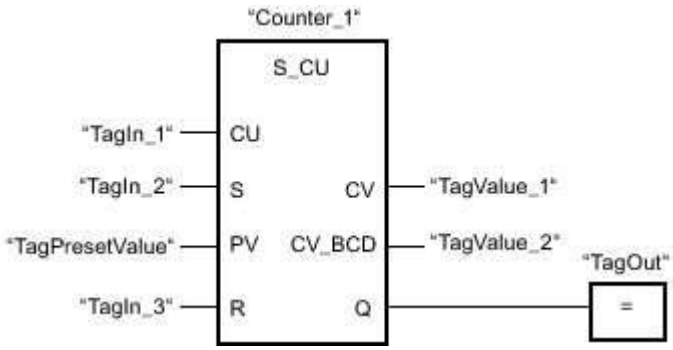
パラメータ	宣言	データタイプ	メモリ領域	説明
<Counter>	InOut/Input	COUNTER	C	命令のカウンタ カウンタの数は CPU によって異なります。
CU	Input	BOOL	I、Q、M、D、L、T、C	カウントアップ入力
S	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウンタのプリセット用の入力
PV	Input	WORD	I、Q、M、D、L、または定数	カウンタ値のプリセット(C#0 ~ C#999)
R	Input	BOOL	I、Q、M、D、L、T、C、または定数	リセット入力
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値 (16 進数)

CV_BCD	Output	WORD, S5TIME, DATE	I、Q、M、D、L	現在のカウンタ値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn_1」入力の信号状態が「0」から「1」に切り替わり(立ち上がりエッジ)、現在のカウンタ値が「999」未満の場合、カウンタ値が 1 インクリメントされます。 「TagIn_2」入力の信号状態が「0」から「1」に切り替わると、カウンタ値がオペランド「TagPresetValue」の値にセットされます。 オペランド「TagIn_3」の信号状態が「1」の場合、カウンタ値が「0」にリセットされます。

現在のカウンタ値は、オペランド「TagValue_1」では 16 進数、オペランド「TagValue_2」では BCD 符号化になります。

現在のカウンタ値が「0」と等しくない場合、「TagOut」出力の信号状態は「1」です。

S_CD:パラメータおよびカウントダウンの割り当て

説明

「パラメータおよびカウントダウンの割り当て」命令を使用して、カウンタ値のデクリメントが可能です。 CD 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 デクリメントされます。 現在のカウンタ値は、出力 CV に 16 進数値で出力され、出力 CV_BCD で BCD コード化されます。 カウンタ値は、下限値の「0」に達するまでデクリメントされます。 下限値に達すると、立ち上がりエッジでカウンタ値はそれ以上デクリメントされません。

入力 S の信号状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタが設定され、入力 CD の RLO が「1」の場合、信号エッジで切り替えが検出されていない場合でも、カウンタはその次のスキャンサイクルで適宜カウントします。

R 入力の信号状態が「1」に切り替わると、カウンタ値がゼロに設定されます。 R 入力の信号状態が「1」である限り、CD および S 入力の信号状態の処理はこのカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、出力 Q の信号状態は「1」となります。 カウンタ値がゼロに等しい場合、出力 Q の信号状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「パラメータおよびカウントダウンの割り当て」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

パラメータ

次の表に、「パラメータおよびカウントダウンの割り当て」命令のパラメータを示します。

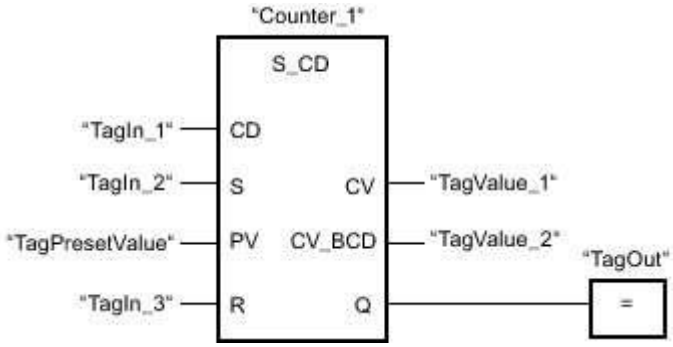
パラメータ	宣言	データタイプ	メモリ領域	説明
<Counter>	InOut/Input	COUNTER	C	命令のカウンタ カウンタの数は CPU によって異なります。
CD	Input	BOOL	I、Q、M、D、L、または定数	カウントダウン入力
S	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウンタのプリセット用の入力
PV	Input	WORD	I、Q、M、D、L、または定数	カウンタ値のプリセット(C#0 ~ C#999)
R	Input	BOOL	I、Q、M、D、L、T、C、または定数	リセット入力
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値 (16 進数)
CV_BCD	Output	WORD, S5TIME, DATE	I、Q、M、D、L	現在のカウンタ値 (BCD フォーマット)

Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn_1」入力の信号状態が「0」から「1」に切り替わり(信号立ち上がりエッジ)、現在のカウンタ値が「0」よりも大きい場合、カウンタ値が 1 デクリメントされます。「TagIn_2」入力の信号状態が「0」から「1」に切り替わると、カウンタ値がオペランド「TagPresetValue」の値にセットされます。 オペランド「TagIn_3」の信号状態が「1」の場合、カウンタ値が「0」にリセットされます。

現在のカウンタ値は、オペランド「TagValue_1」では 16 進数、オペランド「TagValue_2」では BCD 符号化になります。

現在のカウンタ値が「0」と等しくない場合、「TagOut」出力の信号状態は「1」です。

S_CUD:パラメータおよびカウントアップ/カウントダウンの割り当て

説明

「パラメータおよびカウントアップ/カウントダウンの割り当て」命令を使用して、カウンタ値のインクリメントおよびデクリメントが可能です。 CU 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 インクリメントされます。 CD 入力の信号状態が「0」から「1」に切り替わった場合(立ち上がりエッジ)、現在のカウンタ値が 1 デクリメントされます。 現在のカウンタ値は、出力 CV に 16 進数値で出力され、出力 CV_BCD で BCD コード化されます。 1 つのプログラムサイクル内で CU および CD 入力に立ち上がりエッジが発生した場合、カウンタ値は変わりません。

カウンタ値は、上限値の「999」に達するまでインクリメントされます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。 下限の「0」に到達すると、カウンタ値はそれ以上デクリメントされません。

入力 S の信号状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタが設定され、入力 CU および CD の RLO が「1」の場合、信号エッジで切り替えが検出されない場合でも、カウンタはその次のスキャンサイクルで適宜カウントします。

R 入力の信号状態が「1」に切り替わると、カウンタ値がゼロに設定されます。 R 入力の信号状態が「1」である限り、CU、CD および S 入力の信号状態の処理はこのカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、出力 Q の信号状態は「1」となります。 カウンタ値がゼロに等しい場合、出力 Q の信号状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「パラメータおよびカウントアップ/カウントダウンの割り当て」命令には、前にエッジ評価のための論理演算が必要で、ネットワーク内またはネットワークの最後に配置することができます。

パラメータ

次の表に、「パラメータおよびカウントアップ/カウントダウンの割り当て」命令のパラメータを示します。

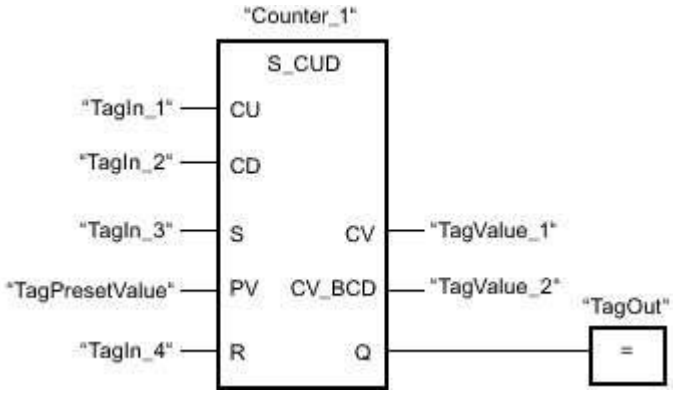
パラメータ	宣言	データタイプ	メモリ領域	説明
<Counter>	InOut/Input	COUNTER	C	命令のカウンタ カウンタの数は CPU によって異なります。
CU	Input	BOOL	I、Q、M、D、L、T、C	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウントダウン入力
S	Input	BOOL	I、Q、M、D、L、T、C、または定数	カウンタのプリセット用の入力
PV	Input	WORD	I、Q、M、D、L、または定数	カウンタ値のプリセット(C#0 ~ C#999)

R	Input	BOOL	I、Q、M、D、L、T、C、または定数	リセット入力
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値 (16 進数)
CV_BCD	Output	WORD, S5TIME, DATE	I、Q、M、D、L	現在のカウンタ値 (BCD フォーマット)
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



"TagIn_1" または "TagIn_2" 入力の信号状態が「0」から「1」に切り替わった(信号立ち上がりエッジ)場合、「パラメータおよびカウントアップ/カウントダウンの割り当て」命令が実行されます。「TagIn_1」入力で立ち上がりエッジが発生し、現在のカウンタ値が「999」未満の場合、カウンタ値が 1 インクリメントされます。 「TagIn_2」入力で立ち上がりエッジが発生し、現在のカウンタ値が「0」よりも大きい場合、カウンタ値が 1 デクリメントされます。

「TagIn_3」入力の信号状態が「0」から「1」に切り替わると、カウンタ値がオペランド「TagPreset-Value」の値にセットされます。 オペランド「TagIn_4」の信号状態が「1」の場合、カウンタ値が「0」にリセットされます。

現在のカウンタ値は、オペランド「TagValue_1」では 16 進数、オペランド「TagValue_2」では BCD 符号化になります。

現在のカウンタ値が「0」と等しくない場合、「TagOut」出力の信号状態は「1」です。

SC: カウンタ値の設定

説明

「カウンタ値のセット」命令を使用して、カウンタの値を設定できます。 この命令は、命令の開始入力での論理演算の結果(RLO)が「0」から「1」へ変わった場合(立ち上がりエッジ)に実行されます。命令が実行されると、カウンタは指定されたカウンタ値に設定されます。

「カウンタ値のセット」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

パラメータ

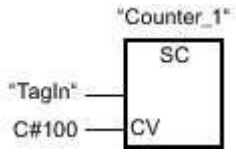
次の表に、「カウンタ値のセット」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、T、C、D、L	開始入力
CV	Input	WORD	I、Q、M、D、L、または定数	カウンタがプリセットされる BCD フォーマットの値。 (C#0 ~ C#999)
<Counter>	InOut/Input	COUNTER	C	プリセットされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn」オペランドの信号状態が「0」から「1」に切り替わると、カウンタ「Counter_1」が値「100」で開始されます。

CU: カウントアップ

説明

「カウントアップ」命令を使用し、指定したカウンタの値を開始入力の立ち上がりエッジで 1 カウント分インクリメントします。カウンタ値は、上限値の「999」に達するまでインクリメントされます。制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。

「カウントアップ」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

パラメータ

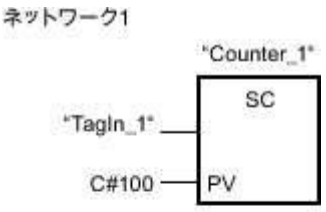
次の表に、「カウントアップ」命令のパラメータを示します。

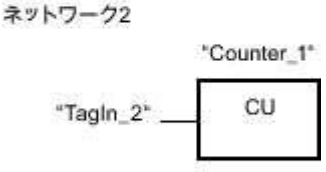
パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Input	BOOL	I、Q、M、T、C、D、L	開始入力
<カウンタ>	InOut/Input	COUNTER	C	値がインクリメントされるカウンタ。

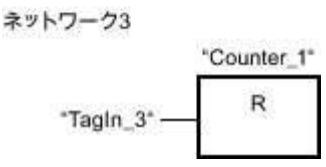
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。







オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると(立ち上がりエッジ)、カウンタ「Counter_1」が値「100」でプリセットされます。

オペランド「TagIn_2」の信号状態が「0」から「1」に切り替わると、「Counter_1」の値が 1 カウントインクリメントされます。

オペランド「TagIn_3」が信号状態「1」を返すと、「Counter_1」の値が「0」にリセットされます。

CD: カウントダウン

説明

「カウントダウン」命令を使用し、指定したカウンタの値を開始入力の立ち上がりエッジで 1 カウント分デクリメントします。 カウンタ値は、下限値の「0」に達するまでデクリメントされます。 制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値は変更されなくなります。

「カウントダウン」命令には、前にエッジ評価のための論理演算が必要で、ネットワークの右側エッジのみに配置することができます。

パラメータ

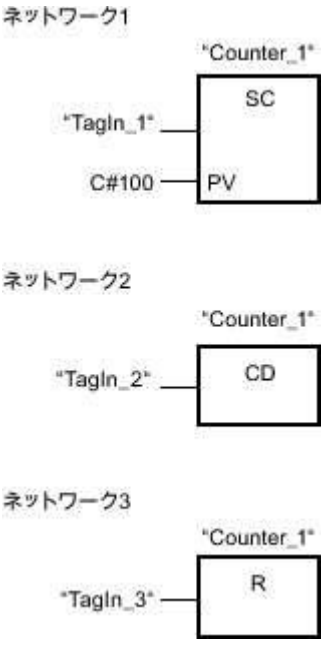
次の表に、「カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Input	BOOL	I、Q、M、T、C、D、L	開始入力
<カウンタ>	InOut/Input	COUNTER	C	値がデクリメントされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」の信号状態が「0」から「1」に切り替わると(立ち上がりエッジ)、カウンタ「Counter_1」が値「100」でプリセットされます。

オペランド「TagIn_2」の信号状態が「0」から「1」に切り替わると、「Counter_1」の値が 1 カウントインクリメントされます。

オペランド「TagIn_3」が信号状態「1」を返すと、「Counter_1」の値が「0」にリセットされます。

比較演算

この章には下記に関する情報が記載されています：

CMP ==: 等価

説明

「等しい」命令を使用して、入力 IN1 の値が入力 IN2 の値と等しいか照会します。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。

次の表に、文字列比較の例をリストします。

IN1	IN2	命令の RLO
'AA'	'AA'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0

「等しい」命令は、文字列の個々の文字も比較します。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

IEC チェックが有効になっている場合、比較するオペランドが同じデータタイプであることが必要です。 IEC チェックが有効になっていない場合、オペランドの幅(長さ)が同じであることが必要です。浮動小数点数を比較する場合、IEC チェックの設定に関わらず比較するオペランドは同じデータタイプであることが必要です。

注記

浮動小数点数の比較

データタイプ REAL または LREAL を比較する場合は、命令「CMP ==: 等しい」の代わりに、命令「IN_RANGE: 範囲内の値」を使用します。

注記

データタイプ PORT の比較

「等しい」命令を使用して「PORT」データタイプのオペランドを比較するには、命令ボックスのドロップダウンリストで WORD データタイプを選択する必要があります。

パラメータ

次の表に、「等しい」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	ビット列、整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、	I、Q、M、D、L、P、または定数	最初の比較値

			LTOD、DTL、DT、LDT		
IN2	Input	ビット列、整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- 「Tag_Value1」 = 「Tag_Value2」の場合に、比較命令の条件が満たされていること。

CMP <>: 不等価

説明

「等しくない」命令を使用して、入力 IN1 の値が入力 IN2 の値と等しくないか照会します。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。

次の表に、文字列比較の例をリストします。

IN1	IN2	命令の RLO
'AA'	'aa'	1
'Hello World'	'HelloWorld'	1
'AA'	'AA'	0

「等しくない」命令は、文字列の個々の文字も比較します。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

IEC チェックが有効になっている場合、比較するオペランドが同じデータタイプであることが必要です。 IEC チェックが有効になっていない場合、オペランドの幅(長さ)が同じであることが必要です。浮動小数点数を比較する場合、IEC チェックの設定に関わらず比較するオペランドは同じデータタイプであることが必要です。

注記

データタイプ PORT の比較

「次の値に等しくない」命令を使用して「PORT」データタイプのオペランドを比較するには、命令ボックスのドロップダウンリストで WORD データタイプを選択する必要があります。

パラメータ

次の表に、「次の値に等しくない」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	ビット列、整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
IN2	Input	ビット列、整数、浮動小数点数、文字列、	ビット列、整数、浮動小数点数、文字列、TIME、LTIME、	I、Q、M、D、L、P、または定数	第 2 の比較値

		TIME、DATE、TOD、DTL	DATE、TOD、LTOD、DTL、DT、LDT		

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- 「Tag_Value1」 <> 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP >=: 以上

説明

「以上」命令を使用して、入力 IN1 の値が入力 IN2 の値以上か照会します。 比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。長い方の文字列の左の部分が、短い方の文字列と同一の場合、長い方の文字列が大きいと見なされます。

次の表に、文字列比較の例をリストします。

IN1	IN2	命令の RLO
'BB'	'AA'	1
'AAA'	'AA'	1
'Hello World'	'Hello World'	1
'Hello World'	'HelloWorld'	0
'AA'	'aa'	0
'AAA'	'a'	0

「以上」命令は、文字列の個々の文字も比較します。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

タイマ値の比較では、入力 IN1 のタイマが入力 IN2 のタイマよりも大きい(より最近)、または等しい場合、命令の RLO は「1」です。

パラメータ

次の表に、「次の値以上」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
IN2	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- 「Tag_Value1」 >= 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP <=: 以下

説明

「以下」命令を使用して、入力 IN1 の値が入力 IN2 の値以下か照会します。比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。 最初に異なる文字が、比較の結果を決定します。長い方の文字列の左の部分が、短い方の文字列と同一の場合、短い方の文字列が小さいと見なされます。

次の表に、文字列比較の例をリストします。

IN1	IN2	命令の RLO
'AA'	'aa'	1
'AAA'	'a'	1
'Hello World'	'Hello World'	1
'HelloWorld'	'Hello World'	0
'BB'	'AA'	0
'AAA'	'AA'	0

「以下」命令は、文字列の個々の文字も比較します。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

タイマ値の比較では、入力 IN1 のタイマが入力 IN2 のタイマよりも小さい(より古い)、または等しい場合、命令の RLO は「1」です。

パラメータ

次の表に、「次の値以下」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
IN2	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- 「Tag_Value1」 <= 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP >: 超過

説明

「超過」命令を使用して、入力 IN1 の値が入力 IN2 の値よりも大きいか照会します。比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。比較は左から右に実行されます。最初に異なる文字が、比較の結果を決定します。長い方の文字列の左の部分が、短い方の文字列と同一の場合、長い方の文字列が大きいと見なされます。

次の表に、文字列比較の例をリストします。

IN1	IN2	命令の RLO
'BB'	'AA'	1
'AAA'	'AA'	1
'AA'	'aa'	0
'AAA'	'a'	0

「超過」命令は、文字列の個々の文字も比較します。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

タイマ値の比較では、入力 IN1 のタイマが入力 IN2 のタイマよりも大きい(より最近)場合、命令の RLO は「1」です。

パラメータ

次の表に、「次の値より大きい」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
IN2	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- 「Tag_Value1」 > 「Tag_Value2」の場合、比較命令の条件が満たされます。

CMP <: 未満

説明

「未満」命令を使用して、入力 IN1 の値が入力 IN2 の値未満であるかを照会します。 比較対象となる 2 つの値は、同じデータタイプである必要があります。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

文字列を比較する際には、個々の文字はそれらのコードによって比較されます(たとえば、「a」は「A」よりも大きい)。 比較は左から右に実行されます。 最初に異なる文字が、比較の結果を決定します。 長い方の文字列の左の部分が、短い方の文字列と同一の場合、短い方の文字列が小さいと見なされます。

次の表に、文字列比較の例をリストします。

<Operand1>	<Operand2>	命令の RLO
'AA'	'aa'	1
'AAA'	'a'	1
'BB'	'AA'	0
'AAA'	'AA'	0

「未満」命令は、文字列の個々の文字も比較します。 比較する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば、「MyString[2]」は「MyString」文字列の 2 番目の文字を比較します。

タイマ値の比較では、入力 IN1 のタイマが入力 IN2 のタイマよりも小さい(より古い)場合、命令の RLO は「1」です。

パラメータ

次の表に、「次の値よりも小さい」命令のパラメータをリストします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	最初の比較値
IN2	Input	整数、浮動小数点数、文字列、TIME、DATE、TOD、DTL	整数、浮動小数点数、文字列、TIME、LTIME、DATE、TOD、LTOD、DTL、DT、LDT	I、Q、M、D、L、P、または定数	第 2 の比較値

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn_1」の信号状態が「1」であること。
- 「Tag_Value1」 < 「Tag_Value2」の場合、比較命令の条件が満たされます。

IN_RANGE: 範囲内の値

説明

「範囲内の値」命令を使って、入力 VAL の値が特定の値の範囲内にあるか照会できます。

MIN および MAX 入力で値の範囲の限度を指定します。「範囲内の値」命令は、入力 VAL の値と、入力 MIN および MAX の値を比較し、結果をボックス出力に送信します。 入力 VAL の値が、比較 MIN <= VAL または VAL <= MAX を満たしている場合、ボックス出力の信号状態は「1」です。 比較の条件が満たされない場合、出力ボックスの信号状態は「0」です。

この比較ファンクションは、比較対象となる値が同じデータタイプである場合のみ実行できます。

パラメータ

次の表に、「範囲内の値」命令のパラメータを示します。

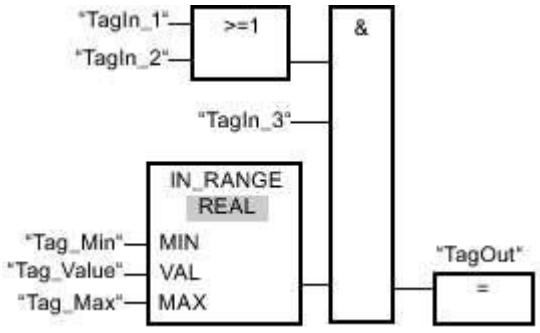
パラメータ	宣言	データタイプ	メモリ領域	説明
MIN	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の下限値
VAL	Input	整数、浮動小数点数	I、Q、M、D、L または定数	比較値
MAX	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の上限値
Box output	Output	BOOL	I、Q、M、D、L	比較の結果

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、出力「TagOut」が設定されます。

- オペランド「TagIn_1」または「TagIn_2」の信号状態が「1」であること。
- オペランド「TagIn_3」の信号状態が「1」であること。

- オペランド「Tag_Value」の値が、オペランド「Tag_Min」および「Tag_Max」(MIN <= VAL または VAL <= MAX)の現在値で指定された値の範囲内になっています。

OUT_RANGE: 範囲外の値

説明

「範囲外の値」命令を使って、入力 VAL の値が特定の値の範囲外にあるか照会できます。

MIN および MAX 入力で値の範囲の限度を指定します。「範囲外の値」命令は、入力 VAL の値と、入力 MIN および MAX の値を比較し、結果をボックス出力に送信します。 入力 VAL の値が、比較 MIN > VAL または VAL > MAX を満たしている場合、ボックス出力の信号状態は「1」です。 データタイプ REAL の指定したオペランドの値が不正な場合、ボックス出力の信号状態は「1」です。

入力 VAL の値が MIN > VAL または VAL > MAX の条件を満たさない場合、ボックス出力が信号状態「0」を返します。

この比較ファンクションは、比較対象となる値が同じデータタイプである場合のみ実行できます。

パラメータ

次の表に、「範囲外の値」命令のパラメータを示します。

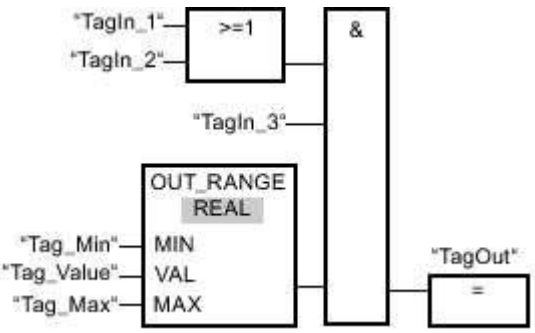
パラメータ	宣言	データタイプ	メモリ領域	説明
MIN	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の下限値
VAL	Input	整数、浮動小数点数	I、Q、M、D、L または定数	比較値
MAX	Input	整数、浮動小数点数	I、Q、M、D、L または定数	値の範囲の上限値
Box output	Output	BOOL	I、Q、M、D、L	比較の結果

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の条件が満たされている場合、出力「TagOut」が設定されます。

- オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」であること。

- オペランド「TagIn_3」の信号状態が「1」であること。
- オペランド「Tag_Value」の値が、オペランド「Tag_Min」および「Tag_Max」(MIN > VAL または VAL > MAX)の現在値で指定された値の範囲外にあります。

OK: 有効性のチェック

説明

「有効性チェック」命令を使って、オペランド(<operand>)の値が有効な浮動小数点数であるかチェックできます。 このチェックは、各プログラムサイクルで実行されます。 照会した時点でオペランドの値が有効な浮動小数点数である場合、出力ボックスが信号状態「1」を返します。 それ以外のすべての場合、「有効性チェック」命令の出力の信号状態は「0」です。

「有効性チェック」命令は、EN 機能と一緒に使用することができます。命令ボックスを EN イネーブル入力に接続すると、このイネーブル入力は有効性クエリの結果が正である場合にのみセットされます。 このファンクションを使用して、指定したオペランドの値が有効な浮動小数点数である場合のみ、この命令を有効にすることができます。

パラメータ

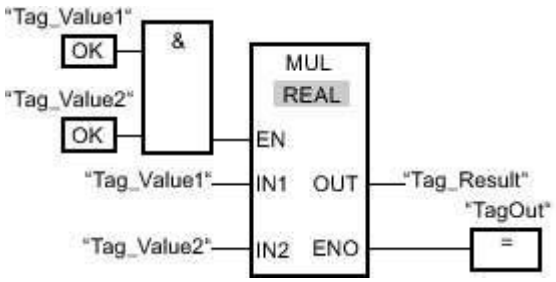
次の例で、「有効性チェック」命令がどのように機能するかを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Input	浮動小数点数	I、Q、M、D、L	チェック対象の値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド Tag_Value1 および Tag_Value2 の値が有効な浮動小数点数を表す場合、「乗算」 (MUL)命令が有効になり、ENO 出力がセットされます。 「乗算」(MUL)命令の実行中、オペランド「Tag_Value1」の値がオペランド「Tag_Value2」の値で乗算されます。 次に、乗算の結果がオペランド「Tag_Result」に格納されます。 この命令の実行中に工ラーが発生しなかった場合、出力 ENO および「TagOut」の信号状態が「1」にセットされます。

NOT_OK: 無効性のチェック

説明

「無効性チェック」命令を使って、オペランド(<operand>)の値が無効な浮動小数点数であるかチェックできます。 このチェックは、各プログラムサイクルで実行されます。 照会した時点でオペランドの値が有効な浮動小数点数である場合、出力ボックスが信号状態「1」を返します。 それ以外のすべての場合、出力ボックスの信号状態は「0」です。

パラメータ

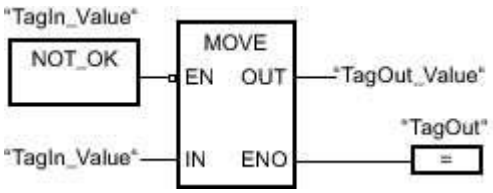
次の表に、「無効性チェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<オペランド>	Input	浮動小数点数	I、Q、M、D、L	チェック対象の値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_Value」の値が無効な浮動小数点数の場合、「値の移動」(MOVE)命令は実行されません。 「TagOut」出力は、信号状態「0」にリセットされます。

VARIANT

この章には下記に関する情報が記載されています：

EQ_Type: EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較

説明

「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグのデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプと、IN2 パラメータのタグのデータタイプを「等しい」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプにすることができます。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

パラメータ

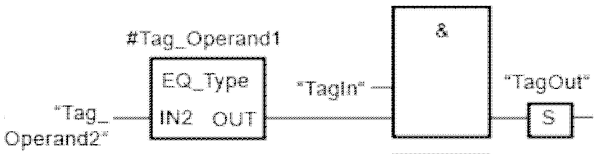
次の表に、「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	I、Q、M、L	最初のオペランド
IN2	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn」のシグナル状態が「1」であること。
- 比較命令の条件が満たされます。すなわち、#Tag_Operand1 オペランドが「Tag_Operand2」に等しくなります。

NE_Type: UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較

説明

「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグが所有しないデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプと、IN2 パラメータのタグのデータタイプを「等しくない」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプにすることができます。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

パラメータ

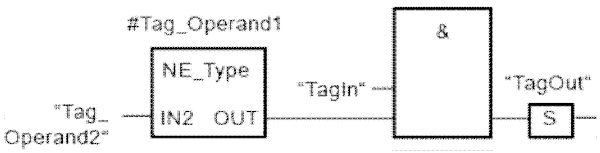
次の表に、「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	I、Q、M、L	最初のオペランド
IN2	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn」のシグナル状態が「1」であること。
- 比較命令の条件が満たされます。すなわち、#Tag_Operand1 オペランドが「Tag_Operand2」に等しくなくなります。

EQ_ElemType: EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較

説明

「EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグのデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプと、IN2 パラメータのタグのデータタイプを「等しい」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプにすることができます。

VARIANT タグのデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

パラメータ

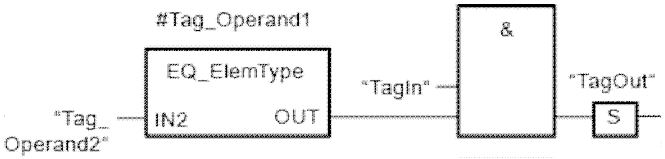
次の表に、「EQUAL (等しい)かどうか配列のデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	I、Q、M、L	最初のオペランド
IN2	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn」のシグナル状態が「1」であること。
- 比較命令の条件が満たされます。すなわち、#Tag_Operand1 オペランドが「Tag_Operand2」に等しくなります。

NE_ElemType: UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較

説明

「UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグが所有しないデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプと、IN2 パラメータのタグのデータタイプを「等しくない」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプにすることができます。

VARIANT タグのデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

比較条件が満たされている場合、この命令は論理操作(RLO)の結果「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

パラメータ

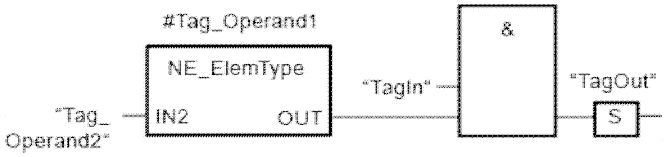
次の表に、「UNEQUAL (等しくない)かどうか配列のデータタイプをタグのデータタイプと比較」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	I、Q、M、L	最初のオペランド
IN2	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L	2 番目のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「TagIn」のシグナル状態が「1」であること。
- 比較命令の条件が満たされます。すなわち、#Tag_Operand1 オペランドが「Tag_Operand2」に等しくなくなります。

IS_NULL: EQUALS NULL ポインタのチェック

説明

「EQUALS NULL ポインタのチェック」命令を使用して、VARIANT が NULL ポインタをポイントし、そのためオブジェクトをポイントしないかどうかを照会することができます。

<Operand>のデータタイプは、VARIANT であることが必要です。

注記

VARIANT タグは、ANY ポインタを指します。

VARIANT タグが ANY ポインタを指す場合、この命令は常に(ANY ポインタが NULL である場合でも)、結果 RLO = 「0」を返します。

パラメータ

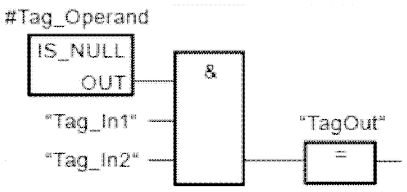
次の表に、「EQUALS NULL ポインタのチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	EQUALS NULL (NULL に等しい)かどうかで比較されるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「Tag_In1」および「Tag_In2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、#Tag_Operand オペランドが、オブジェクトを指していないこと。

NOT_NULL: UNEQUALS NULL ポインタのチェック

説明

「UNEQUALS NULL ポインタのチェック」命令を使用して、が NULL ポインタをポイントせず、VARIANT そのためオブジェクトをポイントするかどうかを照会することができます。

<Operand>のデータタイプは、VARIANT であることが必要です。

注記

VARIANT タグは、ANY ポインタを指します。

VARIANT タグが ANY ポインタを指す場合、この命令は常に(ANY ポインタが NULL である場合でも)、結果 RLO = 「1」を返します。

パラメータ

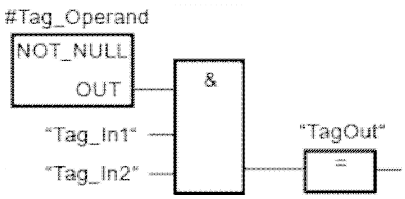
次の表に、「UNEQUALS NULL ポインタのチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	UNEQUALS NULL (NULL に等しくない)かどうかで比較されるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「Tag_In1」および「Tag_In2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされていること。すなわち、#Tag_Operand オペランドが、オブジェクトを指していること。

IS_ARRAY: 配列のチェック

説明

「配列のチェック」命令を使用して、VARIANT が ARRAY データタイプのタグを指すかどうかを照会できます。

<Operand>のデータタイプは、VARIANT であることが必要です。

パラメータ

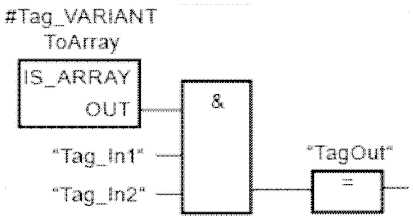
次の表に、「配列のチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	ARRAY に関する照会が行われるオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の条件が満たされている場合、「TagOut」出力がセットされます。

- オペランド「Tag_In1」および「Tag_In2」のシグナル状態が「1」であること。
- 比較命令の条件が満たされます。すなわち「#Tag_VARIANTToArray」オペランドが ARRAY データタイプになります。

四則演算

この章には下記に関する情報が記載されています：

CALCULATE: 計算

説明

「計算」命令を使用して、選択したデータタイプに応じた数学演算および複雑な論理演算の計算のための式を定義し、実行します。

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。 選択したデータタイプに基づいて、特定の命令の機能を組み合わせて複雑な計算を実行できます。 計算する式は、命令ボックスの上部にある「計算機」アイコンで開くことができるダイアログで指定します。 式には、入力パラメータの名前と命令の構文が含めることができます。 オペランド名またはオペランドアドレスの指定は許可されていません。

初期状態では、この命令ボックスには少なくとも 2 つの入力(IN1 および IN2)が含まれています。 入力数は拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。

入力の値は、指定された式を実行するのに使われます。 定義済み入力のすべてを式に使用する必要はありません。 命令の結果は、ボックス出力 OUT に転送されます。

注記
式の算術演算の 1 つでエラーが発生すると、結果は OUT 出力に転送されず、ENO 許可出力のシグナル状態が「1」になります。

式にボックスで使用できない入力を使用すると、これらの入力は自動的に挿入されます。 ただし、式で新たに定義される入力の番号に途切れがないことが条件です。 たとえば、入力 IN3 が定義されていない場合、式で入力 IN4 を使います。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 「計算」命令の結果または暫定結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。
- 式で指定されたいずれかの命令の実行中にエラーが発生していること。

次の表に、選択したデータタイプに基づいて「計算」命令の式で組み合わせて実行できる命令を示します。

データタイプ	命令	構文	例
ビット列	AND: AND 演算	AND	IN1 AND IN2 OR IN3
	OR: OR 論理和演算	OR	
	XOR: EXCLUSIVE OR 排他的論理和演算	XOR	
	INV: 1 の補数の作成	NOT	
	SWAP: スワップ [1)]	SWAP	
整数	ADD: 加算	+	(IN1 + IN2) * IN3; (ABS(IN2)) * (ABS(IN1))
	SUB: 減算	-	
	MUL: 乗算	*	
	DIV: 除算	/	

	MOD: 除算の剰余を返す	MOD	
	INV: 1 の補数の作成	NOT	
	NEG: 2 の補数の作成	-(in1)	
	ABS: 絶対値の形成	ABS( )	
浮動小数点数	ADD: 加算	+	((SIN(IN2) * SIN(IN2) + (SIN(IN3) * SIN(IN3)) / IN3)); (SQR(SIN(IN2)) + (SQR(COS(IN3)) / IN2))
	SUB: 減算	-	
	MUL: 乗算	*	
	DIV: 除算	/	
	EXPT: 累乗	**	
	ABS: 絶対値の形成	ABS( )	
	SQR: 2 乗値の形成	SQR( )	
	SQRT: 平方根の形成	SQRT( )	
	LN: 自然対数の形成	LN( )	
	EXP: 指数値の形成	EXP( )	
	FRAC: 小数部を返す	FRAC( )	
	SIN: 正弦値の形成	SIN( )	
	COS: 余弦値の形成	COS( )	
	TAN: 正接値の形成	TAN( )	
	ASIN: 逆正弦値の形成	ASIN( )	
	ACOS: 逆余弦値の形成	ACOS( )	
	ATAN: 逆正接値の形成	ATAN( )	
	NEG: 2 の補数の作成	-(in1)	
	TRUNC: 数値の切り捨て	TRUNC( )	
	ROUND: 数値の四捨五入	ROUND( )	
	CEIL: 浮動小数点数から次に大きい整数を生成	CEIL( )	
	FLOOR: 浮動小数点数から次に小さい整数を生成	FLOOR( )	

[1)]データタイプ BYTE では使用不可。

パラメータ

次の表に、「計算」命令のパラメータを示します。

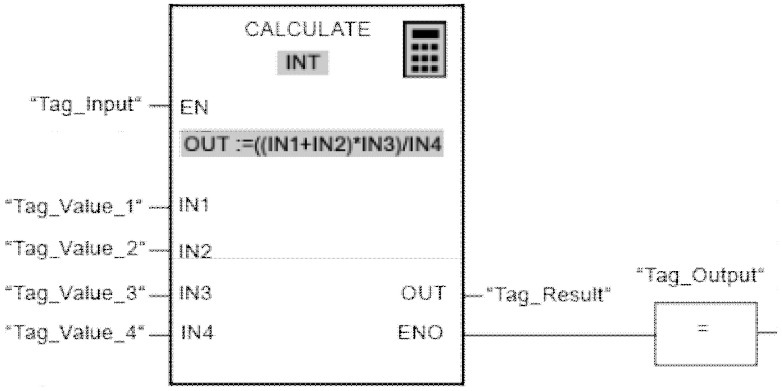
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	ビット列、整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	最初に利用可能な入力

IN2	Input	ビット列、整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	2 番目に使用可能な入力
INn	Input	ビット列、整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	追加で挿入された入力
OUT	Output	ビット列、整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	最終結果の転送先の出力。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value_1	4
IN2	Tag_Value_2	4
IN3	Tag_Value_3	3
IN4	Tag_Value_4	2
OUT	Tag_Result	12

「Tag_Input」のシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value_1」の値がオペランド「Tag_Value_2」の値に加算されます。 合計がオペランド「Tag_Value_3」の値で乗算されます。 積がオペランド「Tag_Value_4」の値で除算されます。 商は、命令の OUT 出力のオペランド「Tag_Result」に終了結果として転送されます。 個々の命令の実行中にエラーが発生しない場合、出力 ENO およびオペランド「Tag_Output」が「1」にセットされます。

ADD: 加算

説明

「加算」命令を使って、入力 IN1 の値を入力 IN2 の値に加算し、出力 OUT (OUT := IN1+IN2)で合計を照会できます。

初期状態では、この命令ボックスには 2 つ以上の入力(IN1 および IN2)が含まれています。 入力数は拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。 この命令の実行中に、すべての使用可能な入力パラメータの値が加算されます。 合計は出力 OUT に格納されます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「加算」命令のパラメータを示します。

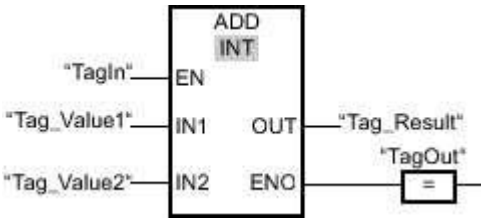
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	加算する最初の数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	加算する 2 番目の数
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	加算されるオプションの入力値。
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	合計

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値がオペランド「Tag_Value2」の値に加算されます。 加算の結果はオペランド「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

SUB: 減算

説明

「減算」命令を使って、入力 IN2 の値を入力 IN1 から減算し、出力 OUT (OUT := IN1-IN2)での差分を照会します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「減算」命令のパラメータを示します。

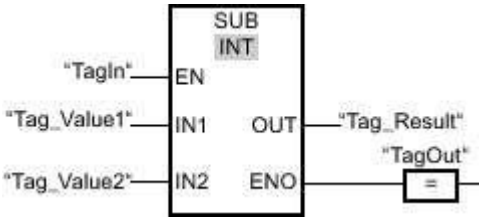
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	被減数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	減数
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	差

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value2」の値が、オペランド「Tag_Value1」の値から減算されます。 減算の結果はオペランド「Tag_Re-

sult」に格納されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

MUL: 乗算

説明

「乗算」命令を使って、入力 IN1 の値を入力 IN2 の値で乗算し、出力 OUT (OUT := IN1 * IN2)での積を照会します。

初期状態では、この命令ボックスには 2 つ以上の入力(IN1 および IN2)が含まれています。 入力数は拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。 命令が実行されると、すべての使用可能な入力パラメータの値が乗算されます。 積は OUT 出力に格納されます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「乗算」命令のパラメータを示します。

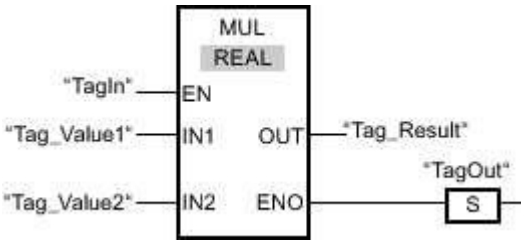
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	乗算の最初の値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	乗算の 2 番目の値
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	乗算されるオプションの入力値。
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	製品

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値をオペランド「Tag_Value2」の値で乗算します。 乗算の結果はオペランド「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

DIV: 除算

説明

「除算」命令を使って、入力 IN1 の値を入力 IN2 の値で除算し、出力 OUT (OUT := IN1/IN2)での商を照会します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「除算」命令のパラメータを示します。

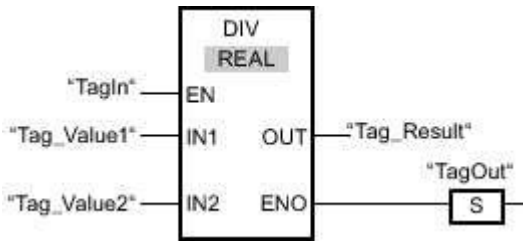
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	被除数
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	除数
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	商の値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値が、オペランド「Tag_Value2」の値で除算されます。 除算の結果はオペランド「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

MOD: 除算の剰余を返す

説明

「除算の剰余を返す」命令を使用して、入力 IN1 の値を入力 IN2 の値で除算し、出力 OUT で除算の余りを照会します。

パラメータ

次の表に、「除算の剰余を返す」命令のパラメータを示します。

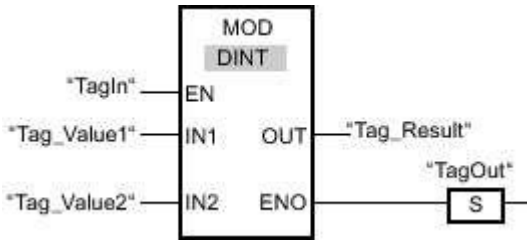
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	被除数
IN2	Input	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	除数
OUT	Output	整数	I、Q、M、D、L、P	I、Q、M、D、L、P	除算の余り

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値が、オペランド「Tag_Value2」の値で除算されます。 除算の余りはオペランド「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

NEG: 2 の補数の作成

説明

「2 の補数の作成」命令を使用して、入力 IN の値の符号を変更し、出力 OUT で結果を照会します。 たとえば、入力 IN に正の値がある場合、この値をマイナスにした値が出力 OUT に出力されます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の結果が、出力 OUT で指定されたデータタイプに許可されている範囲外であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「2 つの補数の作成」命令のパラメータを示します。

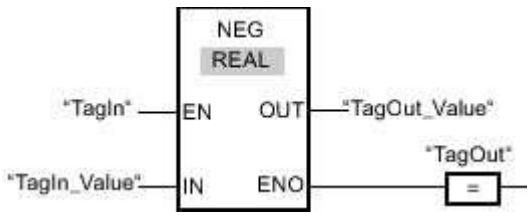
パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力値の 2 の補数

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」での値の符号は変更され、結果は出力「TagOut_Value」に格納されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

INC: インクリメント

説明

「インクリメント」命令を使って、パラメータ IN/OUT のオペランドの値を次の大きな値に変更し、結果を照会します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「インクリメント」命令のパラメータを示します。

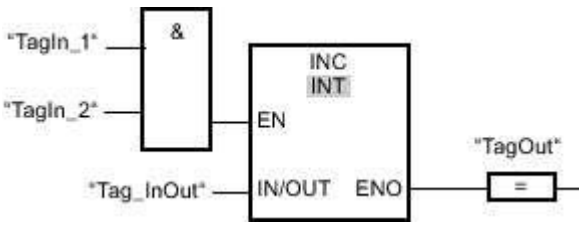
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN/OUT	InOut	整数	I、Q、M、D、L	I、Q、M、D、L	インクリメントされる値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



オペランド TagIn_1 および TagIn_2 のシグナル状態が「1」の場合、オペランド「Tag_InOut」の値が 1 インクリメントされ、出力「TagOut」が設定されます。

DEC: デクリメント

説明

「デクリメント」命令を使って、パラメータ IN/OUT のオペランドの値を次の小さな値に変更し、結果を照会します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「デクリメント」命令のパラメータを示します。

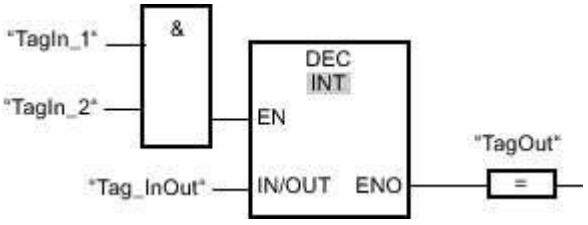
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN/OUT	InOut	整数	I、Q、M、D、L	I、Q、M、D、L	デクリメントされる値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、オペランド「Tag_InOut」の値が 1 デクリメントされ、出力「TagOut」が設定されます。

ABS: 絶対値の形成

説明

「絶対値の形成」命令を使って、入力 IN で指定した値の絶対値を計算します。 命令の結果は、OUT 出力で出力され、そこで照会できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「絶対値の形成」命令のパラメータを示します。

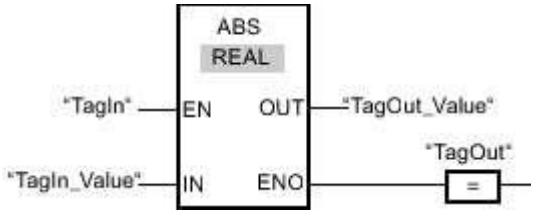
パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力値の絶対値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	-6.234
OUT	TagOut_Value	6.234

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「TagIn_Value」の値の絶対値を計算し、結果を出力「TagOut_Value」に送信します。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

MIN: 最小値の取得

説明

「最小値の取得」命令は、使用可能な入力の値を比較し、最小値を出力 OUT に書き込みます。 入力の数は、追加入力によって、命令ボックスで拡張できます。 入力は、ボックスで昇順に番号付けされます。

初期状態では、この命令ボックスには 100 つ以上の入力(IN1 および IN2)が含まれています。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「最小値の取得」命令のパラメータを示します。

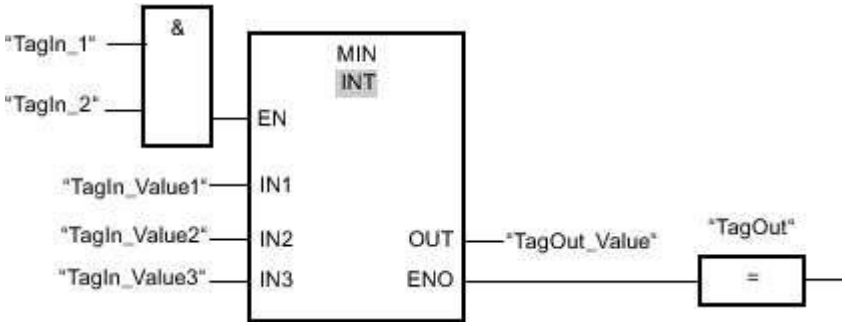
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	2 番目の入力値
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	値が比較される追加挿入された入力
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	結果
IEC チェックが有効でない場合も、同じ長さのビット列をデータタイプとして選択することによって、データタイプ TIME、LTIME、TOD、LTOD、DATE および LDT のタグを使用することができます (TIME => DINT の代わりに、UDINT または DWORD = 32 ビットなど)。					

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	12222

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、指定されたオペランドの値を比較し、最小値(「TagIn_Value1」)を出力「TagOut_Value」にコピーします。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

MAX: 最大値の取得

説明

「最大値の取得」命令は、使用可能な入力の値を比較し、最大値を出力 OUT に書き込みます。 入力の数は、追加入力によって、命令ボックスで拡張できます。 入力は、ボックスで昇順に番号付けされます。

初期状態では、この命令ボックスには 100 つ以上の入力(IN1 および IN2)が含まれています。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

パラメータ

次の表に、「最大値の取得」命令のパラメータを示します。

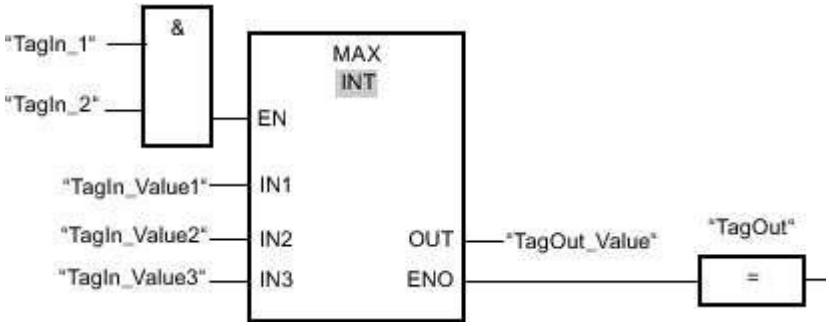
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	2 番目の入力値
INn	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	値を比較する追加で挿入された入力。
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	結果
IEC チェックが有効でない場合も、同じ長さのビット列をデータタイプとして選択することによって、データタイプ TIME、LTIME、TOD、LTOD、DATE および LDT のタグを使用することができます (TIME => DINT の代わりに、UDINT または DWORD = 32 ビットなど)。					

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	TagOut_Value	14444

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、指定されたオペランドの値を比較し、最大値(「TagIn_Value2」)を出力「TagOut_Value」にコピーします。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

LIMIT: 制限値の設定

説明

「制限値の設定」命令を使用して入力 IN の値を入力 MN および MX の値に制限することができます。IN 入力の値が MN 条件<= IN <= MX を満たす場合、この値は OUT 出力にコピーされます。 条件が満たされず入力値 IN が下限値 MN を下回ると、出力 OUT は入力 MN の値にセットされます。 上限値 MX を超えると、OUT 出力が MX 入力の値にセットされます。

入力 MN の値が入力 MX の値よりも大きい場合、結果は定義されず、許可出力 ENO は「0」となります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 指定された複数のタグが同じデータタイプでないこと。
- オペランドの値が無効である。
- MN 入力の値が MX 入力の値よりも大きいこと。

パラメータ

次の表に、「制限値の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
MN	Input	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	下限値
IN	Input	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
MX	Input	整数、浮動小数点数、TIME、TOD、DATE	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	上限値
OUT	Output	整数、浮動小数点数、	整数、浮動小数点数、	I、Q、M、D、L、P	I、Q、M、D、L、P	結果

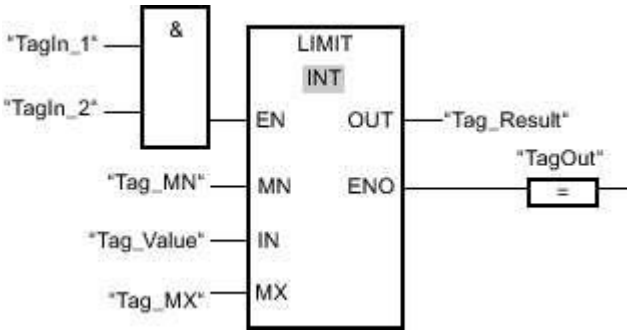
		TIME、TOD、DATE	TIME、LTIME、TOD、LTOD、DATE、LDT			
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。						

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MN	Tag_MN	12000
IN	Tag_Value	8000
MX	Tag_MX	16000
OUT	Tag_Result	12000

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 オペランド「Tag_Value」の値が、オペランド「Tag_MN」および「Tag_MX」の値と比較されます。「Tag_Value」 オペランドの値が下限値未満であるため、「Tag_MN」オペランドの値が「Tag_Result」出力にコピーされます。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

SQR: 2 乗値の形成

説明

「2 乗値の形成」命令を使用して、入力 IN の浮動小数点数の値を二乗し、結果を出力 OUT に書き込みます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「2 乗値の形成」命令のパラメータを示します。

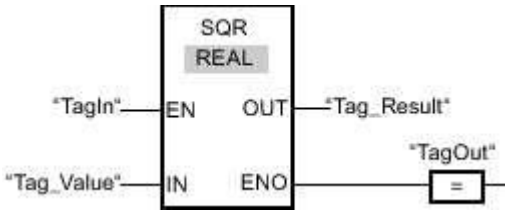
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力値の 2 乗値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	5.0
OUT	Tag_Result	25.0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、オペランド「Tag_Value」の値を 2 乗し、結果を出力「Tag_Result」に送信します。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

SQRT: 平方根の形成

説明

「平方根の形成」命令を使用して、入力 IN の浮動小数点数の値の平方根を形成し、結果を出力 OUT に書き込みます。 この命令は、入力値がゼロよりも大きい場合に正の結果になります。 入力値がゼロ未満の場合、出力 OUT が無効な浮動小数点数を返します。 入力 IN の値が「0」の場合、結果も「0」になります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。
- 入力 IN の値が負。

パラメータ

次の表に、「平方根の形成」命令のパラメータを示します。

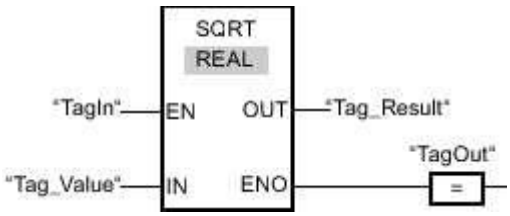
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L	I、Q、M、D、L	入力値の平方根

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	25.0
OUT	Tag_Result	5.0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、オペランド「Tag_Value」の平方根を計算し、結果を出力「Tag_Result」に送信します。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

LN: 自然対数の形成

説明

「自然対数の形成」命令を使用して、入力 IN の値の底 e (e=2.718282)に対する自然対数を計算できます。 結果は出力 OUT に送信され、そこで照会できます。 この命令は、入力値がゼロよりも大きい場合に正の結果になります。 入力値がゼロ未満の場合、出力 OUT が無効な浮動小数点数を返します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。
- 入力 IN の値が負。

パラメータ

次の表に、「自然対数の形成」命令のパラメータを示します。

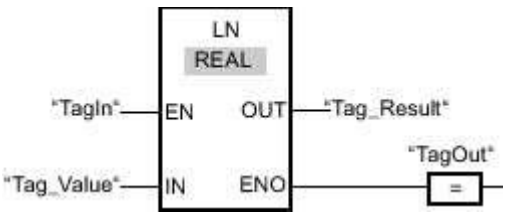
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力値の自然対数

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「Tag_Value」の値の自然対数を形成し、結果を出力「Tag_Result」に送信します。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

EXP: 指数値の形成

説明

「指数値の形成」命令を使用して、底 e (e=2.718282)および入力 IN で指定された値から指数を計算できます。 出力 OUT で提供される結果を照会できます(OUT = e^{IN})。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「指数値の形成」命令のパラメータを示します。

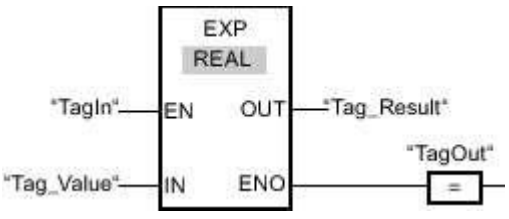
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力値 IN の指数値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、底 e とオペランド「Tag_Value」の値から指数を形成し、結果を出力「Tag_Result」に送信します。この命令が エラーなしで実行された場合、「TagOut」出力がセットされます。

SIN: 正弦値の形成

説明

「正弦値の形成」命令を使って、角度の正弦を計算できます。 角度の大きさは、入力 IN で弧度法を使って指定します。 命令の結果は、OUT 出力で出力され、そこで照会できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「正弦値の形成」命令のパラメータを示します。

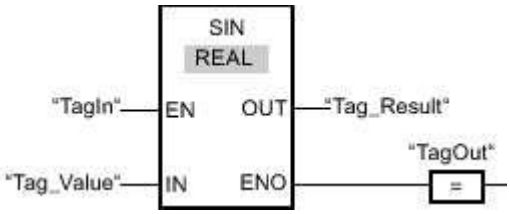
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	ラジアン単位の角度のサイズ
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	指定された角度の正弦

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	1.0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「Tag_Value」で指定され角度の正弦を計算し、結果を出力「Tag_Result」に送信します。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

COS: 余弦値の形成

説明

「余弦値の形成」命令を使って、角度の余弦を計算できます。 角度の大きさは、入力 IN で弧度法を使って指定します。 命令の結果は、OUT 出力で出力され、そこで照会できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「余弦値の形成」命令のパラメータを示します。

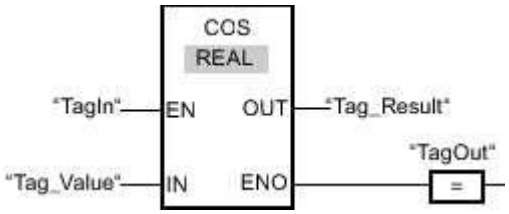
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	ラジアン単位の角度のサイズ
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	指定された角度の余弦

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	+1.570796 (π/2)
OUT	Tag_Result	0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「Tag_Value」で指定され角度の余弦を計算し、結果を出力「Tag_Result」に送信します。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

TAN: 正接値の形成

説明

「正接値の形成」命令を使って、角度の正接を計算できます。角度の大きさは、入力 IN で弧度法を使って指定します。命令の結果は、出力 OUT で提供され、そこで照会できます。

次のいずれかの条件が満たされる場合、イネーブル出力 ENO の信号状態は「0」になります。

- イネーブル入力 EN の信号状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「正接値の形成」命令のパラメータを示します。

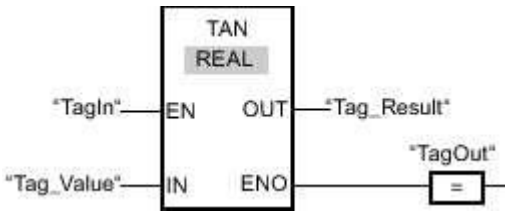
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	イネーブル出力
IN	Input	浮動小数点数	I、Q、M、D、L、P または定数	I、Q、M、D、L、P または定数	ラジアン単位の角度のサイズ
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	指定された角度の正接

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	+3.141593 (π)

OUT	Tag_Result	0

オペランド「TagIn」の信号状態が「1」の場合、「正接値の形成」命令が実行されます。 この命令は、入力「Tag_Value」で指定され角度の正接を計算し、結果を出力「Tag_Result」に送信します。 この命令の実行中にエラーが発生しなかった場合、出力「TagOut」が設定されます。

ASIN: 逆正弦値の形成

説明

「逆正弦値の形成」命令を使用すると、この値と対応する入力 IN で指定された正弦値から角度の大きさを計算できます。 入力 IN で、-1 ~ +1 の範囲の有効な浮動小数点数のみを指定できます。計算された角度の大きさは、弧度法を使用して OUT 出力に出力され、-π/2 ~ +π/2 の範囲の値を取ることができます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。
- 入力 IN の値が許容範囲外(-1 ~ +1)。

パラメータ

次の表に、「逆正弦値の形成」命令のパラメータを示します。

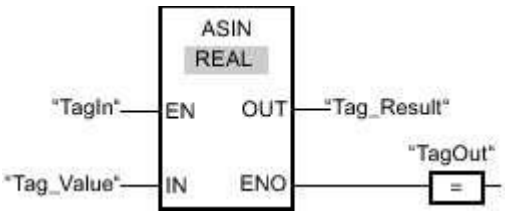
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	正弦値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	1.0
OUT	Tag_Result	+1.570796 (π/2)

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「Tag_Value」に対応する正弦値の角度の大きさを計算します。 命令の結果は、「Tag_Result」出力に格納されます。この命令が工ラーなしで実行された場合、「TagOut」出力がセットされます。

ACOS: 逆余弦値の形成

説明

「逆余弦値の形成」命令を使用すると、この値と対応する入力 IN で指定された余弦値から角度の大きさを計算できます。 入力 IN で、-1 ~ +1 の範囲の有効な浮動小数点数のみを指定できます。 計算された角度の大きさは、弧度法を使用して OUT 出力に出力され、0 ~ +π の範囲の値を取ることができます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。
- 入力 IN の値が許容範囲外(-1 ~ +1)。

パラメータ

次の表に、「逆余弦値の形成」命令のパラメータを示します。

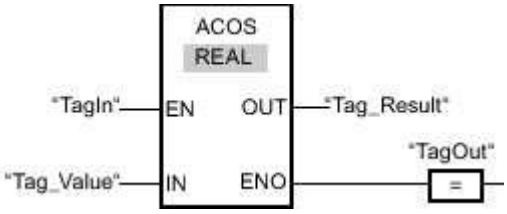
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	余弦値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	0
OUT	Tag_Result	+1.570796 (π/2)

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「Tag_Value」に対応する余弦値の角度の大きさを計算します。 命令の結果は、「Tag_Result」出力に格納されます。この命令が工ラーなしで実行された場合、「TagOut」出力がセットされます。

ATAN: 逆正接値の形成

説明

「逆正接値の形成」命令を使用すると、この値と対応する入力 IN で指定された正接値から角度の大きさを計算できます。 入力 IN には、有効な浮動小数点数(または-NaN/+NaN)のみを指定することができます。 計算された角度の大きさは、弧度法を使用して OUT 出力に出力され、-π/2～+π/2 の範囲の値を取ることができます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN の値が有効な浮動小数点数でない。

パラメータ

次の表に、「逆正接値の形成」命令のパラメータを示します。

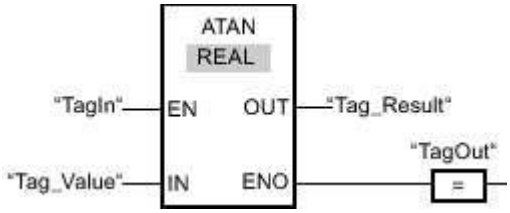
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	正接値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	1.0

OUT	Tag_Result	+0.785398 (π/4)

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、入力「Tag_Value」に対応する正接値の角度の大きさを計算します。 命令の結果は、「Tag_Result」出力に格納されます。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

FRAC: 小数部を返す

説明

「小数部を返す」命令を使用して、IN 入力の値の小数位を求めることができます。 照会の結果は出力 OUT に格納され、そこで照会できます。 たとえば、入力 IN の値が 123.4567 の場合、出力 OUT の値が 0.4567 となります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- この命令の実行中にエラーが発生した場合、たとえば入力に有効な浮動小数点数がない場合。

パラメータ

次の表に、「小数部を返す」命令のパラメータを示します。

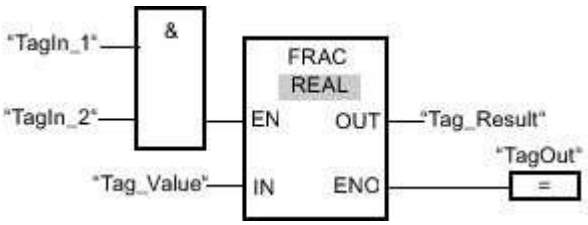
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	小数点以下の桁数を確定する入力値。
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力 IN の入力値の小数位

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	2.555

OUT	Tag_Result	0.555

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が開始されます。 オペランド「Tag_Value」の値からの小数位が、オペランド「Tag_Result」にコピーされます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

EXPT: 累乗

説明

「累乗」命令を使って、入力 IN1 の値を入力 IN2 の値で指定された指数で累乗することができます。命令の結果は、出力 OUT に格納され、そこで照会できます(OUT = $IN1^{IN2}$)。

入力 IN1 の値は、有効な浮動小数点数であることが必要です。 入力 IN2 の設定では、整数も許可されます。

次のいずれかの条件が満たされる場合、イネーブル出力 ENO の信号状態は「0」になります。

- イネーブル入力 EN の信号状態が「0」であること。
- この命令の処理中にエラーが発生した場合、たとえばオーバーフローが発生した場合。

パラメータ

次の表に、「累乗」命令のパラメータを示します。

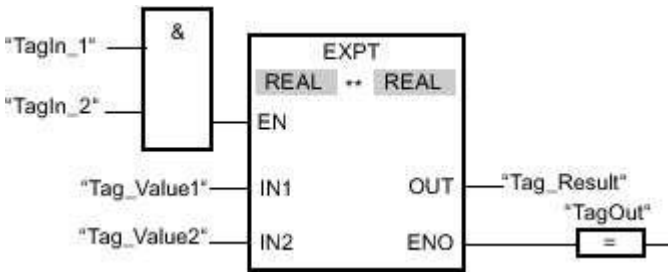
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	イネーブル出力
IN1	Input	浮動小数点数	I、Q、M、D、L、P または定数	I、Q、M、D、L、P または定数	底の値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P または定数	I、Q、M、D、L、P または定数	底の値が累乗される値
OUT	Output	浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	結果

命令ボックスの「<???>」ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



オペランド「TagIn_1」および「TagIn_2」の信号状態が「1」の場合、「累乗」命令が実行されます。オペランド「Tag_Value1」の値が、オペランド「Tag_Value2」の値で累乗されます。 結果は出力「Tag_Result」に格納されます。この命令がエラーなしで実行された場合、ENO イネーブル出力の信号状態が「1」となり、「TagOut」出力がセットされます。

ムーブ操作

この章には下記に関する情報が記載されています：

MOVE: 値のムーブ

説明

「値の移動」命令を使用し、IN 入力のオペランドの内容を OUT1 出力のオペランドに転送します。 この転送は、常に上位アドレスの方向に行われます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- IN パラメータのデータタイプが、OUT1 パラメータで指定されたデータタイプと一致しません。

次の表に、S7-1200 CPU シリーズで使用可能な転送方法をリストします。

転送元(IN)	宛先(OUT1)	
	IEC チェックあり	IEC チェックなし
BYTE	BYTE, WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE ,TOD, CHAR
WORD	WORD, DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD, CHAR
DWORD	DWORD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, TIME, DATE, TOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
USINT	USINT, UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
INT	INT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UINT	UINT, UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
DINT	DINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
UDINT	UDINT	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME, DATE, TOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LREAL
TIME	TIME	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TIME
DATE	DATE	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, DATE
TOD	TOD	BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, TOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD、CHAR、文字列の文字 [1)]

WCHAR	WCHAR	BYTE、WORD、DWORD、CHAR、WCHAR、文字列の文字 1)
文字列の文字 1)	文字列の文字	CHAR、WCHAR、文字列の文字
ARRAY2)	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
PLC データタイプ(UDT)	PLC データタイプ(UDT)	PLC データタイプ(UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUN-TER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER

次の表に、S7-1500 CPU シリーズで使用可能な転送方法をリストします。

転送元(IN)	**宛先(OUT1)**	
	IEC チェックあり	**IEC チェックなし**
BYTE	BYTE, WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE ,TOD, LTOD, CHAR
WORD	WORD, DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, S5TIME, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
DWORD	DWORD, LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
LWORD	LWORD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LREAL, TIME, LTIME, LDT, DATE, TOD, LTOD, CHAR
SINT	SINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
USINT	USINT, UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
INT	INT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD

UINT	UINT, UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD. LTOD
DINT	DINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
UDINT	UDINT, ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
LINT	LINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
ULINT	ULINT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME, LTIME, LDT, DATE, TOD, LTOD
REAL	REAL	DWORD, REAL
LREAL	LREAL	LWORD, LREAL
S5TIME	S5TIME	WORD, S5TIME
TIME	TIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TIME
LTIME	LTIME	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTIME
DATE	DATE	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, DATE
DT	DT	DT
LDT	LDT	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LDT
TOD	TOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, TOD
LTOD	LTOD	BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, LTOD
DTL	DTL	DTL
CHAR	CHAR	BYTE, WORD, DWORD, LWORD、CHAR、文字列の文字 [1)]
WCHAR	WCHAR	BYTE、WORD、DWORD、LWORD、CHAR、WCHAR、文字列の文字 [1)]
文字列の文字 [1)]	文字列の文字	CHAR、WCHAR、文字列の文字
ARRAY[2)]	ARRAY	ARRAY
STRUCT	STRUCT	STRUCT
COUNTER	COUNTER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
TIMER	TIMER, WORD, INT	WORD, DWORD, INT, UINT, DINT, UDINT
PLC データタイプ(UDT)	PLC データタイプ(UDT)	PLC データタイプ(UDT)
IEC_TIMER	IEC_TIMER	IEC_TIMER
IEC_LTIMER	IEC_LTIMER	IEC_LTIMER

IEC_SCOUNT-ER	IEC_SCOUNTER	IEC_SCOUNTER
IEC_US-COUNTER	IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUN-TER	IEC_COUNTER	IEC_COUNTER
IEC_UCOUNT-ER	IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNT-ER	IEC_DCOUNTER	IEC_DCOUNTER
IEC_UD-COUNTER	IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNT-ER	IEC_LCOUNTER	IEC_LCOUNTER
IEC_UL-COUNTER	IEC_ULCOUNTER	IEC_ULCOUNTER

1) 「値のムーブ」命令を使用して、文字列の個々の文字を CHAR または WCHAR データタイプのオペランドに転送することもできます。 転送する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば「MyString[2]」は、「MyString」文字列の 2 番目の文字を転送します。 データタイプ CHAR または WCHAR のオペランドを文字列の個々の文字に転送することも可能です。 文字列の特定の文字を別の文字列の文字と置換することもできます。

2) 配列(ARRAY)全体の転送は、入力 IN および出力 OUT1 のオペランドの配列コンポーネントが同じデータタイプの場合のみ可能です。

IN 入力のデータタイプのビット長が OUT1 出力のデータタイプのビット長を超えると、転送元の値の上位ビットが失われます。 入力 IN のデータタイプのビット長が出力 OUT1 のデータタイプのビット長を下回る場合、転送先の値の上位ビットはゼロで上書きされます。

初期状態では、命令ボックスには 1 出力が含まれています(OUT1)。 出力数は拡張できます。 追加された出力は、ボックスで昇順に番号付けされます。 この命令の実行中、入力 IN のオペランドの内容が、使用できるすべての出力に転送されます。 構造体データタイプ(DTL、STRUCT、ARRAY)または文字列の文字が転送される場合、命令ボックスは拡張できません。

「ブロックの移動」(MOVE_BLK)および「割り込み不可能なブロックの移動」(UMOVE_BLK)命令を使用して、ARRAY データタイプのオペランドを移動することもできます。 データタイプ STRING または WSTRING のオペランドは、「文字列のムーブ」(S_MOVE)を使用してコピーすることができます。

パラメータ

次の表に、「値の移動」命令のパラメータを示します。

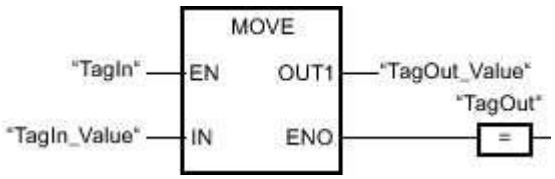
パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数、浮動小数点数、タイマ、日付と時	ビット列、整数、浮動小数点数、タイマ、日付と時	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	移動先アドレスの上書きに使用す

		刻、CHAR、WCHAR、STRUCT、ARRAY、IEC データタイプ、PLC データタイプ (UDT)	刻、CHAR、WCHAR、STRUCT、ARRAY、TIMER、COUNTER、IEC データタイプ、PLC データタイプ(UDT)			るエレメント。
OUT1	Output	ビット列、整数、浮動小数点数、タイマ、日付と時刻、CHAR、WCHAR、STRUCT、ARRAY、IEC データタイプ、PLC データタイプ (UDT)	ビット列、整数、浮動小数点数、タイマ、日付と時刻、CHAR、WCHAR、STRUCT、ARRAY、TIMER、COUNTER、IEC データタイプ、PLC データタイプ(UDT)	I、Q、M、D、L	I、Q、M、D、L	宛先アドレス

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0011 1111 1010 1111
OUT1	TagOut_Value	0011 1111 1010 1111

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、「TagIn_Value」オペランドの内容を「TagOut_Value」オペランドにコピーします。 この命令がエラーなしで実行された場合、ENO および "TagOut" 許可出力がシグナル状態「1」にセットされます。

逆シリアル化: 逆シリアル化

説明

「逆シリアル化」命令を使用して、PLC データタイプ(UDT)のシーケンシャル表示を PLC データタイプに変換して戻し、その内容全体を表示することができます。

PLC データタイプのシーケンシャル表示を配置するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。変換の前に、十分なメモリ領域が存在することを確認してください。

この命令では、変換された PLC データタイプの複数のシーケンシャル表示をそれらの元の状態に、順を追って変換し直すことができます。

PLC データタイプ(UDT)の単一のシーケンシャル表示のみを変換して元に戻す場合は、命令「TRCV: 通信接続経由のデータ受信」を指示することもできます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

パラメータ

次の表に、「逆シリアル化」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC_ARRAY	Input	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるグローバルデータブロック
DEST_VARIABLE	InOut	VARIANT	I、Q、M、L	返された変換済み PLC データタイプ(UDT)が格納されるタグ
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

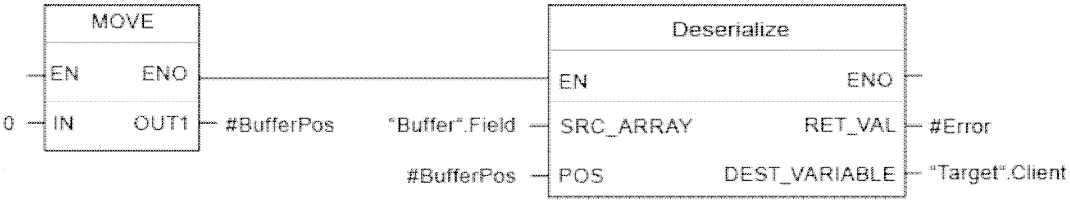
エラーコード*	説明

(W#16#...)	
0000	エラーは発生していません。
80B0	SRC_ARRAY パラメータと DEST_VARIABLE パラメータのためのメモリ領域がオーバーラップしています。
8136	DEST_VARIABLE パラメータのデータブロックが標準アクセスによるブロックではありません。
8150	SRC_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8151	SRC_ARRAY パラメータのコード生成エラー
8153	SRC_ARRAY パラメータのメモリの空き領域が不足しています。
8250	DEST_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8251	DEST_VARIABLE パラメータのコード生成エラー
8254	DEST_VARIABLE パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

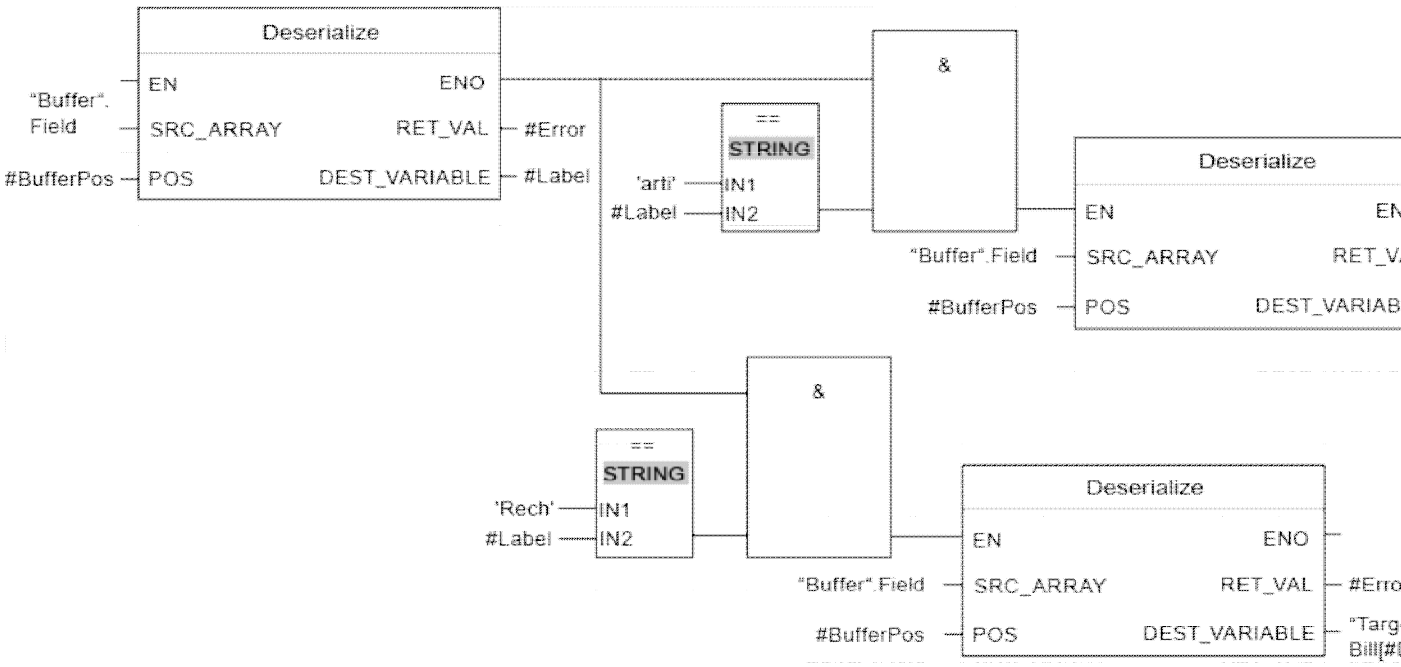
次の例で、命令がどのように動作するかを示します。

ネットワーク 1:



「値のムーブ」命令は、「#BufferPos」オペランドに値「0」を移動します。 「逆シリアル化」命令は、Buffer」データブロックのカスタマデータのシーケンシャル表示を逆シリアル化し、「Target」データブロックに書き込みます。 変換済みカスタマデータで使用されるバイト数は、「#BufferPos」オペランドに格納されます。

ネットワーク 2:



「逆シリアル化」命令は、「Buffer」データブロックの、カスタマデータの後にシーケンシャル表示で保存されたセパレータシートのシーケンシャル表示を逆シリアル化し、これらの文字を「#Label」オペランドに書き込みます。 これらの文字は、比較命令「arti」および「Bill」を使用して比較されます。 「arti」 = TRUE の場合の比較では、データは、逆シリアル化され、「Target」データブロックに書き込まれたアーティクルデータです。 「Bill」 = TRUE の場合の比較では、データは、逆シリアル化され、「Target」データブロックに書き込まれたビリングデータです。

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言をリストします。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Target	Client	「クライアント」(PLC データタイプ)
	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
	Bill	INT の配列[0..10]
Buffer	Field	バイトの配列[0..294]

シリアル化: シリアル化:

説明

「シリアル化」命令を使用して、データタイプの構造体の部分を失うことなく、複数の PLC データタイプ(UDT)を 1 つのシーケンシャル表示に変換することができます。

この命令を使用して、ユーザプログラムの複数の構造体データをキャッシュバッファ(グローバルデータブロックを推奨)に格納し、別の CPU に送信することができます。 変換済み PLC データタイプを格納するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。 変換の前に、十分なメモリ領域が存在することを確認してください。

POS パラメータのオペランドには、変換済み PLC データタイプによって使用されるバイト数に関する情報が収納されています。

単一の PLC データタイプ(UDT)を送信する場合、直接に命令「TSEND: 通信接続経由のデータ送信」を呼び出すことができます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

パラメータ

次の表に、「シリアル化」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC_VARIABLE	Input	VARIANT	I、Q、M、L	シーケンシャル表示に変換される PLC データタイプ (UDT)。
DEST_ARRAY	InOut	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるデータブロック。
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

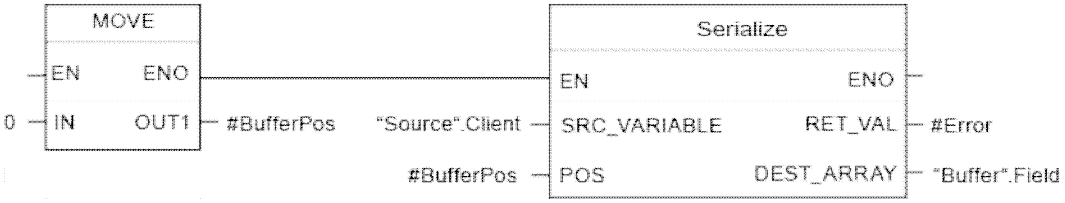
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_VARIABLE パラメータと DEST_ARRAY パラメータのためのメモリ領域がオーバーラップしています。
8150	SRC_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8152	SRC_VARIABLE パラメータのコード生成エラー
8236	DEST_ARRAY パラメータのデータブロックが標準アクセスによるブロックではありません。
8250	DEST_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8252	DEST_ARRAY パラメータのコード生成エラー
8253	DEST_ARRAY パラメータのメモリの空き領域が不足しています。
8254	DEST_ARRAY パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

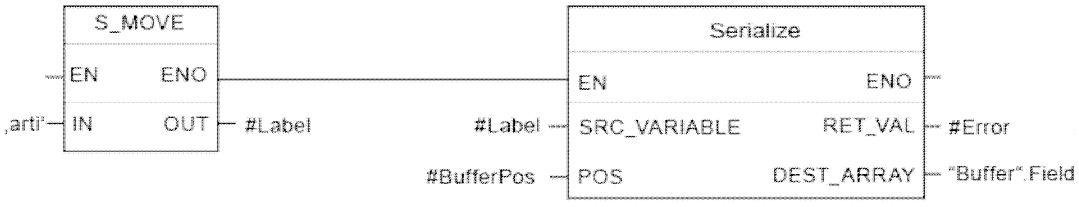
次の例で、命令がどのように動作するかを示します。

ネットワーク 1:



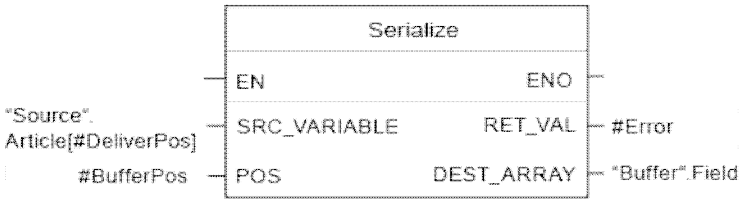
「値のムーブ」命令は、「#BufferPos」オペランドに値「0」を移動します。 「シリアル化」命令は、「Source」データブロックのカスタマデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックに書き込みます。 シーケンシャル表示によって使用されたバイト数は、「#BufferPos」オペランドに格納されます。

ネットワーク 2:



後でのシーケンシャル表示の逆シリアル化を容易にするために、その時点では、一種のセパレータシートが挿入されます。 「文字列のムーブ」命令は、「arti」文字を「#Label」オペランドに移動します。 「シリアル化」命令は、これらの文字を「Buffer」データブロックのカスタマデータの後に書き込みます。 文字が必要とするバイト数が、「#BufferPos」オペランドに格納済みの数に加算されます。

ネットワーク 3:



「シリアル化」命令は、「Source」データブロックの、ランタイム時に計算される特定のアーティクルのデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックの「arti」文字の後に書き込みます。

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言をリストします。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Source	Client	「クライアント」(PLC データタイプ)
	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
Buffer	Field	バイトの配列[0..294]

MOVE_BLK: ブロックムーブ

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 移動先の領域に移動するエレメントの数は、COUNT パラメータで指定されます。 移動するエレメントの幅は、入力 IN のエレメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

パラメータ

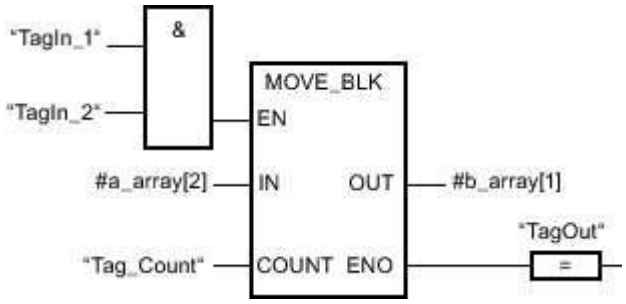
次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT、UINT、UDINT	USINT、UINT、UDINT、ULINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	ソース領域から宛先領域にコピーするエレメントの数。
OUT [1)]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント
[1)] 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。						

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。 これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つのエレメントで構成されます。

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

MOVE_BLK_VARIANT: ブロックムーブ

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 配列全体または配列の1エレメントを同じデータタイプの別の配列にコピーすることができます。 ソース配列と宛先配列のサイズ(エレメントの数)は異なる場合があります。配列内の複数エレメントまたは単一エレメントをコピーすることができます。

ブロックの作成時にこの命令を使用する場合、VARIANT ごとにソースおよび宛先が転送されるため、配列はまだ既知である必要はありません。

SRC_INDEX および DEST_INDEX パラメータでのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- コピーされるデータが使用可能なデータよりも多いこと。

注記

BOOL データタイプと接続している VARIANT

データタイプ VARIANT のパラメータ(ソース領域または宛先領域)とデータタイプ BOOL のタグまたは ARRAY of BOOL を相互接続する場合、以下のオプションがあります。

1. シンボリックにアドレス指定することができます。

   例: SRC パラメータ: "Data_block".myArray
2. 任意のポインタによってアドレス指定することができます。 ただし、領域で指定された長さは 8 で割り切れるようにする必要があります。割り切れない場合、この命令は実行されません。

   例: SRC パラメータ: P#DB123.DBX456.0 BOOL 1000

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC	Input	VARIANT (配列または個々の配列エレメントをポイントする)、<Data_type>の配列	I、Q、M、L	コピー元のソースブロック
COUNT	Input	UDINT	I、Q、M、D、L、または定数	コピーされるエレメント数 ARRAY が SRC パラメータまたは DEST パラメータで指定されていない場合は、

				COUNT パラメータに値「1」を割り当てます。
SRC_INDEX	Input	DINT	I、Q、M、D、L、または定数	• SRC_INDEX パラメータは、ゼロベースで計算されます。 ARRAY がパラメータ SRC で指定されると、パラメータ SRC_INDEX の整数はコピー元のソース領域内の最初の エレメントを指定します。 宣言された配列の限界値には関係ありません。 • ARRAY がパラメータ SRC で指定されないか、または配列の 1 つの単一エレメントのみが指定された場合、パラメータ SRC_INDEX に値「0」を割り当てます。
DEST_INDEX	Input	DINT	I、Q、M、D、L、または定数	• DEST_INDEX パラメータは、ゼロベースで計算されます。 ARRAY がパラメータ DEST で指定されると、パラメータ DEST_INDEX の整数はコピー先の宛先領域内の最初の エレメントを指定します。 宣言された配列の限界値には関係ありません。 • ARRAY が DEST パラメータで指定されていない場合は、DEST_INDEX パラメータに値「0」を割り当てます。
DEST	Output [1)]	VARIANT	I、Q、M、L	ソースブロックの内容がコピーされる宛先領域。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
1) データがタグに流れるため、DEST パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

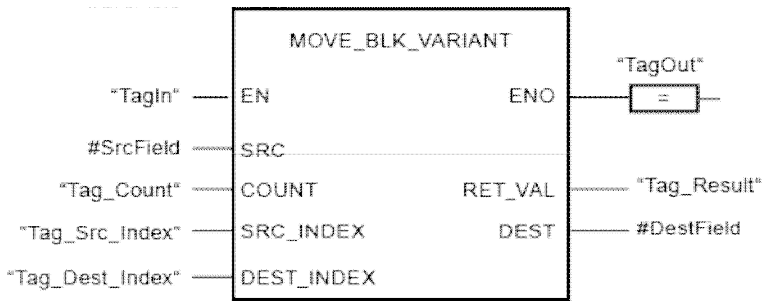
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	データタイプが一致していません
8151	SRC パラメータにアクセスできません。
8152	SRC パラメータのオペランドが入力されていません。
8153	SRC パラメータのコード生成エラー
8154	SRC パラメータのオペランドのデータタイプが BOOL です。
8281	COUNT パラメータの値が無効です
8382	SRC_INDEX パラメータの値が VARIANT の限界値外です。
8383	SRC_INDEX パラメータの値が配列の上限値外です
8482	DEST_INDEX パラメータの値が VARIANT の限界値外です。
8483	DEST_INDEX パラメータの値が配列の上限値外です
8534	DEST パラメータが書き込み保護されています
8551	DEST パラメータにアクセスできません。
8552	DEST パラメータのオペランドが入力されていません。
8553	DEST パラメータのコード生成エラー
8554	DEST パラメータのオペランドのデータタイプが BOOL です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
SRC	Input	#SrcField	ローカルオペランド #SrcField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。

			("MOVE_UDT"の ARRAY[0..10]
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	ローカルオペランド #DestField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。 ("MOVE_UDT"の ARRAY[10..20]

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 UDT の配列の 4 番目のエレメントから始めて、2 つのエレメントが、ソース領域から宛先領域にコピーされます。 コピーが、UDT の配列の 4 番目のエレメントから始めて、挿入されます。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

UMOVE_BLK: 割り込み不可能なブロックムーブ

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 移動先の領域に移動する工レメントの数は、COUNT パラメータで指定されます。 移動する工レメントの幅は、入力 IN の工レメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	ソース領域から宛先領域にコピーするエレメントの数。
OUT [1]	Output	2 進数、整数、浮動小数	2 進数、整数、浮動小数	D、L	D、L	ソース領域の内容がコ

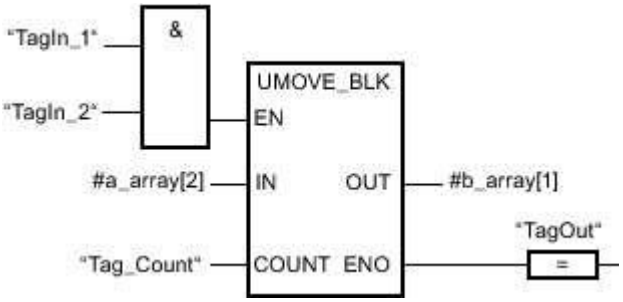
		点数、タイマ、DATE、CHAR、WCHAR、TOD	点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD			ピーされる宛先領域の最初のエレメント

[1] 指定されたデータタイプは、Array 構造体のエレメントとしてしか使用できません。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。 他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

FILL_BLK: フィル命令

説明

「フィル」命令を使用して、IN 入力の値でメモリ領域(宛先領域)を埋めることができます。 宛先領域は、OUT 出力で指定されたアドレスから開始して埋められます。 ムーブ操作の繰り返し回数は、COUNT パラメータの値で指定されます。 この命令が実行された場合、入力 IN の値が選択され、COUNT パラメータの値で指定された回数だけ移動先の領域に移動されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用するエレメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	ムーブ操作の繰り返し回数
OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	D、L	フィル命令が開始する宛先領域のアドレス

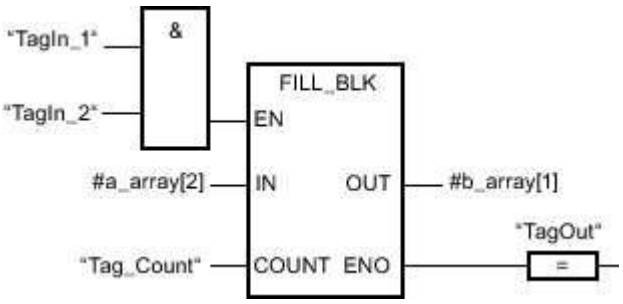
[1] 指定されたデータタイプは、Array 構造体の工レメントとして使用することもできます。

[2] 指定されたデータタイプは、Array 構造体の工レメントとしてしか使用できません。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。 この命令の実行中にエラーが発生しなかった場合、出力 ENO および「TagOut」のシグナル状態が「1」にセットされます。

UFILL_BLK: 割り込み不可能なフィル命令

説明

「割り込み不可能なフィル」命令を使用して、割り込まれることなく IN 入力の値でメモリ領域(宛先領域)を満たすことができます。 宛先領域は、OUT 出力で指定されたアドレスから開始して埋められます。 ムーブ操作の繰り返し回数は、COUNT パラメータの値で指定されます。 この命令が実行された場合、入力 IN の値が選択され、COUNT パラメータの値で指定された回数だけ移動先の領域に移動されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込み不可能なフィル」命令の実行中には、CPU のアラーム応答時間が長くなります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

ARRAY of BOOL がコピーされると、ARRAY 構造体のバイト限界値を超えるまで、オーバーフローの許可出力 ENO が「1」にセットされます。 COUNT 入力の値によって ARRAY 構造体のバイト制限値を超えた場合、ENO 許可出力が「0」にリセットされます。

「割り込み不可能なフィル」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

パラメータ

次の表に、「割り込み不可能なフィル」命令のパラメータを示します。

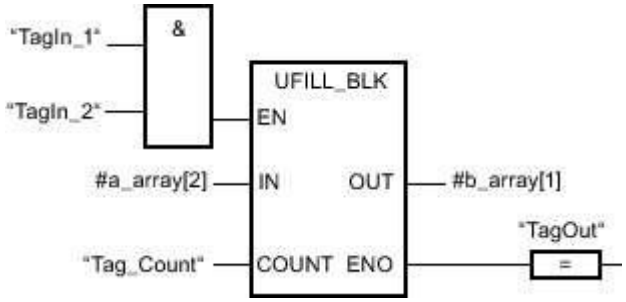
パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用するエレメント
COUNT	Input	USINT、UINT、UDINT	USINT、UINT、UDINT、ULINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	ムーブ操作の繰り返し回数

OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	D、L	フィル命令が開始する宛先領域のアドレス
[1] 指定されたデータタイプは、Array 構造体の工レメントとして使用することもできます。 [2] 指定されたデータタイプは、Array 構造体の工レメントとしてしか使用できません。						

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つの工レメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つの工レメントで構成されます。

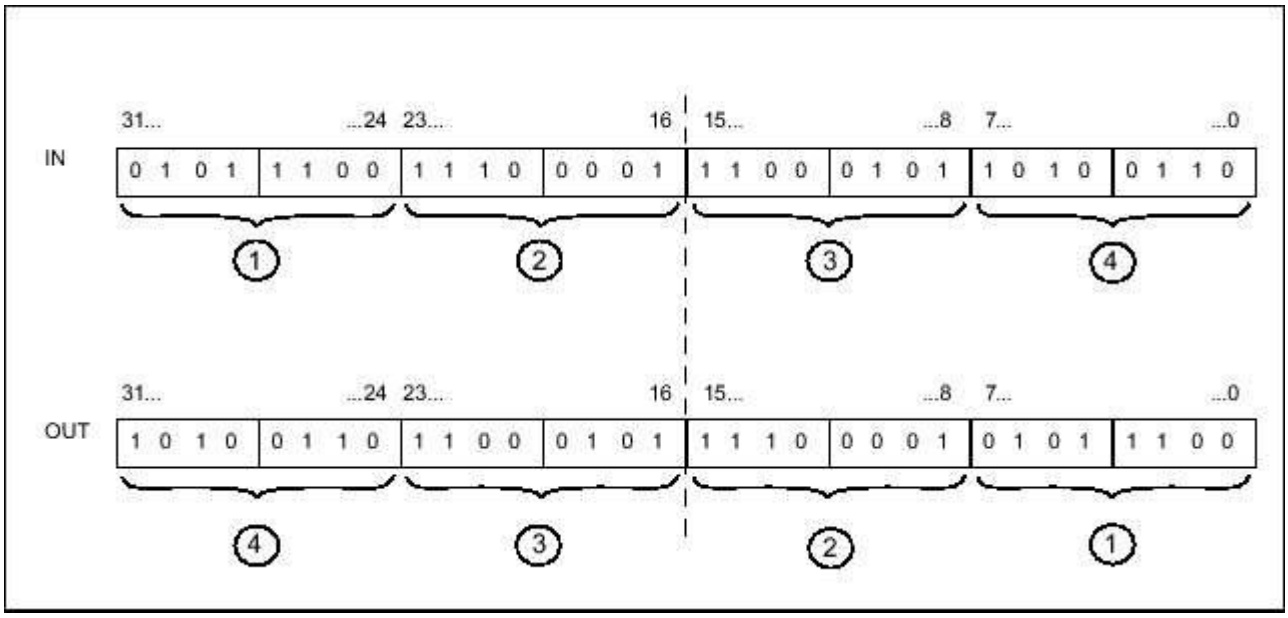
オペランド「TagIn_1」および「TagIn_2」のシグナル状態が「1」の場合、命令が実行されます。 この命令は、#a_array タグの 3 番目の工レメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。 他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 この命令の実行中にエラーが発生しなかった場合、出力 ENO および「TagOut」のシグナル状態が「1」にセットされます。

SWAP: スワップ

説明

「スワップ」命令を使用して、入力 IN のタグ内のバイトの順番を変更し、出力 OUT で結果を照会することができます。

次の図に、「スワップ」命令を使って DWORD データタイプオペランドのバイトを交換する方法を示します。



パラメータ

次の表に、「スワップ」命令のパラメータを示します。

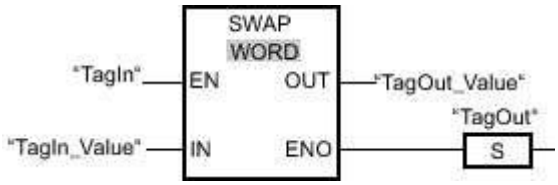
パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	WORD, DWORD	WORD, DWORD, LWORD	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	バイトがスワップされるオペランド
OUT	Output	WORD, DWORD	WORD, DWORD, LWORD	I、Q、M、D、L、P	I、Q、M、D、L、P	結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0000 1111 0101 0101
OUT	TagOut_Value	0101 0101 0000 1111

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 バイトの整列が変更され、オペランド「TagOut_Value」に格納されます。

配列 DB

この章には下記に関する情報が記載されています：

- ReadFromArrayDB: 配列データブロックからの読み出し (S7-1500)
- WriteToArrayDB: 配列データブロックへの書き込み (S7-1500)
- ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し (S7-1500)
- WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み (S7-1500)

ReadFromArrayDB: 配列データブロックからの読み出し

説明

「配列データブロックからの読み出し」命令を使用し、配列データブロックからデータを読み出して、宛先領域に書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列での カウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	Output 1)	VARIANT	I、Q、M、L	読み出されて出力される値
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
1) データがタグに流れるため、VALUE パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

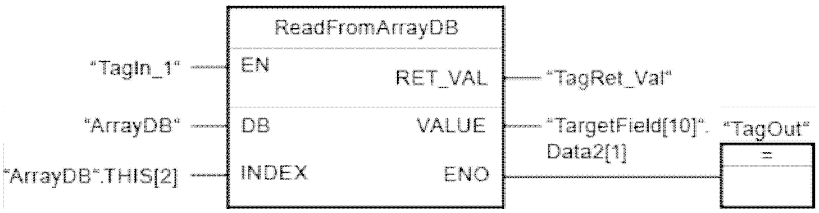
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、書き込み保護されているか、またはロードメモリ内に存在します。
8135	配列データブロックに無効な値が含まれます。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8450	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8452	コード生成エラー
8453	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	TargetField[10].Data2[1]	「TargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[10 to 20]です。

「TagIn1」オペランドのシグナル状態が「1」の場合、「ARRAY データブロックからの読み出し」命令が実行されます。「ArrayDB」から 3 番目のエレメントが読み出され、「TargetField[10].Data2[1]」オペランドに書き込まれます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

WriteToArrayDB: 配列データブロックへの書き込み

説明

「配列データブロックへの書き込み」命令を使用して、配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の工レメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内の工レメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

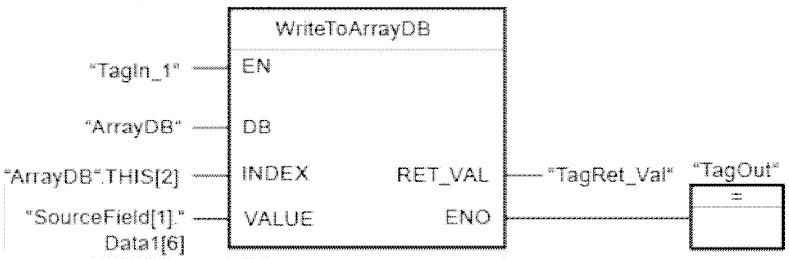
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存された工レメントのデータタイプが、VARIANT で転送される工レメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。

8134	データブロックが書き込み保護されています。
8135	データブロックが配列データブロックではありません。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8350	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8352	コード生成エラー
8353	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	SourceField[1].Data1[6]	「SourceField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。

「TagIn1」オペランドのシグナル状態が「1」の場合、「ARRAY データブロックへの書き込み」命令が実行されます。 「SourceField」オペランドから、2 番目のエレメントの「Data1[6]」エレメントが「ArrayDB」に書き込まれます。 3 番目のエレメントは、「ArrayDB」に書き込まれます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し

説明

「ロードメモリの配列データブロックからの読み出し」命令を使用して、ロードメモリの配列データブロックからデータを読み出します。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 この命令は、BUSY パラメータで信号立ち下がりエッジが検出された場合に終了します。 1 プログラムサイクルの間、DONE パラメータのシグナル状態が「1」になり、このサイクル内に、読み取られた値が VALUE パラメータに出力されます。 他のすべてのプログラムサイクルでは、VALUE パラメータの値は変更されません。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「ロードメモリの配列データブロックからの読み出し」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」: 配列 DB の読み出しから開始
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	InOut	VARIANT	I、Q、M、L	読み出されて出力される値

				TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ読み出し中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

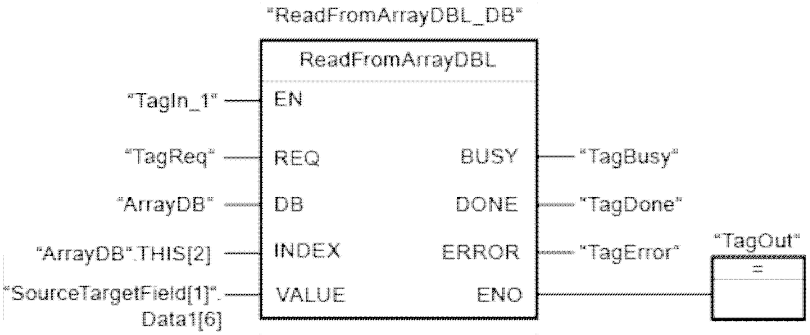
次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL:ロードメモリのデータブロックからの読み出し」および「WRIT_DBL:データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目の工レメント
VALUE	SourceTargetField[1].Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagIn1」オペランドのシグナル状態が「1」で、「TagReq」オペランドで立ち上がりエッジが検出されると、「ロードメモリの配列データブロックからの読み出し」命令が実行されます。「ArrayDB」から 3 番目の工レメントが読み出され、「SourceTargetField[1].Data1[6]」オペランドに書き込まれます。「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値は変更されません。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み

説明

「ロードメモリの配列データブロックへの書き込み」命令を使用して、ロードメモリの配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の工レメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 BUSY パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、VALUE パラメータの値がデータブロックに書き込まれます。 DONE1 パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、パラメータの値がデータブロックに書き込まれます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 命令の実行中にエラーが発生した場合。

パラメータ

次の表に、「ロードメモリの配列データブロックへの書き込み」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」: 配列 DB への書き込みを開始
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内の工レメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値

				TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ書き込み中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

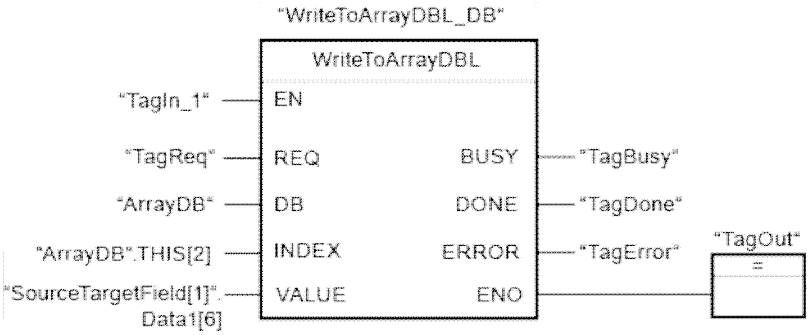
次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存された工レメントのデータタイプが、VARIANT で転送される工レメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8234	データブロックが書き込み保護されています。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックの工レメントの長さと一致しません。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL:ロードメモリのデータブロックからの読み出し」および「WRIT_DBL:データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB.THIS[2]	「ArrayDB」の 3 番目の工レメント
VALUE	SourceTargetField[1].Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagIn1」オペランドのシグナル状態が「1」で、「TagReq」オペランドで立ち上がりエッジが検出された場合、「ロードメモリの配列データブロックへの書き込み」命令が実行されます。 「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値が「ArrayDB」の 3 番目の工レメントに書き込まれます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

VARIANT

この章には下記に関する情報が記載されています：

- VariantGet: VARIANT タグ値の読み出し (S7-1200, S7-1500)
- VariantPut: VARIANT タグ値の書き込み (S7-1200, S7-1500)
- CountOfElements: 配列エレメントの数を取得 (S7-1200, S7-1500)

VariantGet: VARIANT タグ値の読み出し

説明

「VARIANT タグ値の読み出し」命令を使用して、SRC パラメータの VARIANT が指すタグの値を読み出し、その値を DST パラメータのタグに書き込むことができます。

SRC パラメータのデータタイプは、VARIANT です。 DST パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

DST パラメータのタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT: ブロックムーブ」命令を使用できます。 追加情報については、「関連項目」を参照してください。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- データタイプが一致していません。 (値は転送されません。)

パラメータ

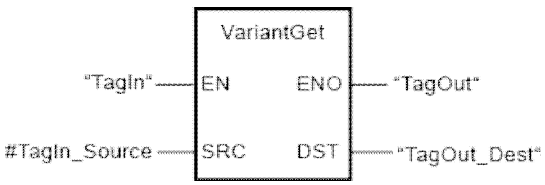
次の表に、「VARIANT タグ値の読み出し」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC	Input	VARIANT	I、Q、M、L	読み出されるタグ
DST	Output	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。「#TagIn_Source」オペランドの VARIANT が指すタグの値が読み出され、「TagOut_Dest」オペランドに書き込まれます。

VariantPut: VARIANT タグ値の書き込み

説明

「VARIANT タグ値の書き込み」命令を使用して、SRC パラメータのタグの値を VARIANT が指す DST パラメータのタグに書き込むことができます。

DST パラメータのデータタイプは、VARIANT です。 SRC パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

SRC パラメータのタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT:ブロックムーブ」命令を使用できます。追加情報については、「関連項目」を参照してください。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- データタイプが一致していません。(値は転送されません。)

パラメータ

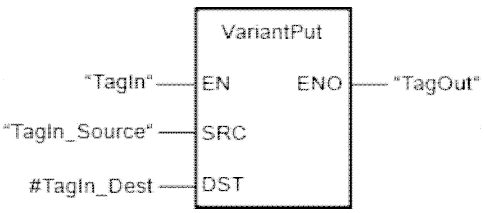
次の表に、「VARIANT タグ値の書き込み」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
SRC	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	読み出されるタグ
DST	Input	VARIANT	I、Q、M、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 「TagIn_Source」オペランドの値が、「#TagIn_Dest」オペランドの VARIANT が指すタグに書き込まれます。

CountOfElements: 配列エレメントの数を取得

説明

「ARRAY エレメント数の取得」命令を使用して、VARIANT を指すタグが所有する ARRAY エレメント数を照会できます。

配列が一次元の ARRAY の場合、上限値と下限値の差 + 1 が結果として出力されます。 配列が多次元の ARRAY の場合、すべての次元の積が結果として出力されます。

IN パラメータのデータタイプは、VARIANT であることが必要です。

次のいずれかの条件が満たされる場合、許可出力 ENO のシグナル状態は「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- VARIANT タグが ARRAY でないこと。 (結果は「0」です。)

VARIANT がブールの配列を指す場合、FILL エレメントもカウントに含まれます。 (たとえば、ブールの配列[0..1]の場合、8 が返されます)

パラメータ

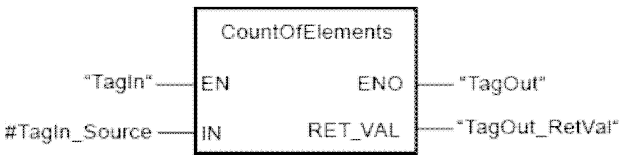
次の表に、「ARRAY エレメント数の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	VARIANT	I、Q、M、L	照会するタグ
RET_VAL	Output	UDINT	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



TagIn」オペランドのシグナル状態が「1」の場合、「ARRAY エレメント数の取得」命令が実行されます。 「#TagIn_Source」オペランドの VARIANT が指すタグの ARRAY エレメント数が読み出され、「TagOut_RetVal」オペランドに出力されます。

レガシー

この章には下記に関する情報が記載されています：

- FieldRead: フィールドの読み出し (S7-1200, S7-1500)
- FieldWrite: フィールドの書き込み (S7-1200, S7-1500)
- BLKMOV: ブロックの移動 (S7-1500)
- UBLKMOV:割り込みなしブロック転送 (S7-1500)
- FILL:ブロックの塗りつぶし (S7-1500)

FieldRead: フィールドの読み出し

説明

「フィールドの読み出し」命令を使用して、MEMBER パラメータで指定されたフィールドから特定のコンポーネントを読み出して、その内容を VALUE パラメータでタグに転送することができます。INDEX パラメータを使用し、読み出すフィールドコンポーネントのインデックスを定義します。MEMBER パラメータで、読み出すフィールドの最初のコンポーネントを指定します。

パラメータ MEMBER のフィールドコンポーネントのデータタイプ、パラメータ VALUE のインデックスおよびタグは、暗黙的変換が不可能なため「フィールドの読み出し」命令のデータタイプに対応する必要があります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- INDEX パラメータで指定されたフィールドコンポーネントが MEMBER パラメータで指定されたフィールドで定義されていない。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「フィールドの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
INDEX	Input	DINT	DINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	内容が読み出されるフィールドコンポーネントのインデックス
MEMBER	Input	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、および WCHAR	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、タイマ、DATE、TOD、LTOD、CHAR、および WCHAR	D、L	D、L	読み出されるフィールドの最初のコンポーネント
VALUE	Output	2 進数、整数、浮動小数点数、タイ	2 進数、整数、浮動小数点数、タイ	I、Q、M、D、L、P	I、Q、M、D、L、P	フィールドコンポーネントの内容

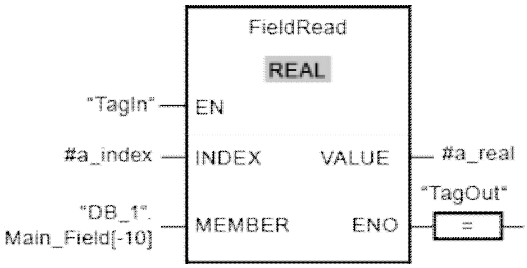
		マ、DATE、TOD、CHAR、WCHAR	マ、DATE、TOD、LTOD、CHAR、WCHAR			の転送先のオペランド。

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	タグ	値
INDEX	a_index	4
MEMBER	"DB_1".Main_Field[-10]	"DB_1"データブロックの"Main_Field[-10..10] of REAL"フィールドの最初のコンポーネント
VALUE	a_real	「Main_Field[-10..10] of REAL」フィールドのインデックス 4 のコンポーネント

インデックス 4 のフィールドコンポーネントが「Main_Field[-10...10] of REAL」フィールドから読み取られ、「a_real」タグに書き込まれます。 読み出されるフィールドコンポーネントは、INDEX パラメータの値で定義されます。

FieldWrite: フィールドの書き込み

説明

「フィールドの書き込み」命令を使用して、VALUE 入力のタグの内容を MEMBER 出力のフィールドの特定のコンポーネントに転送します。 INDEX 入力の値を使用し、記述されるフィールドコンポーネントのインデックスを指定します。 MEMBER 出力で、書き込まれるフィールドの最初のコンポーネントを入力します。

パラメータ MEMBER のフィールドコンポーネントのデータタイプ、パラメータ VALUE のインデックスおよびタグは、暗黙的変換が不可能なため「フィールドの読み出し」命令のデータタイプに対応する必要があります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 INDEX で指定されたフィールドコンポーネントが出力 MEMBER で指定されたフィールドで定義されていないこと。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「フィールドの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
INDEX	Input	DINT	DINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	VALUE の内容で書き込まれるフィールドコンポーネントのインデックス
VALUE	Input	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、TOD、LTOD、CHAR、WCHAR	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	内容のコピー元のオペランド
MEMBER	Output	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、	ARRAY タグのコンポーネントとしての 2 進数、整数、浮動小数点数、	D、L	D、L	VALUE の内容が書き込まれるフィールドの最初のコンポーネント。

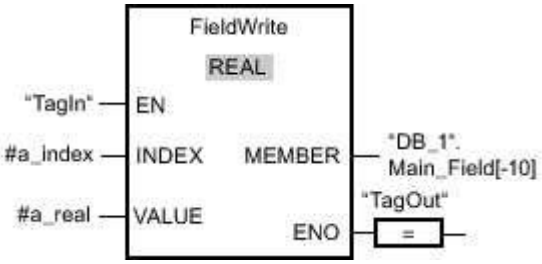
		タイマ、DATE、TOD、CHAR、および WCHAR	タイマ、DATE、TOD、LTOD、CHAR、および WCHAR			

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
INDEX	a_index	4
VALUE	a_real	10.54
MEMBER	"DB_1".Main_Field[-10]	"DB_1"データブロックの"Main_Field[-10..10] of REAL"フィールドの最初のコンポーネント

タグ「a_real」の値「10.54」が、Main_Field[-10 ... 10] of REAL フィールドのインデックス 4 のフィールドコンポーネントに書き込まれます。 タグ「a_real」の内容が転送されるフィールドコンポーネントのインデックスは、入力 INDEX の値によって指定されます。

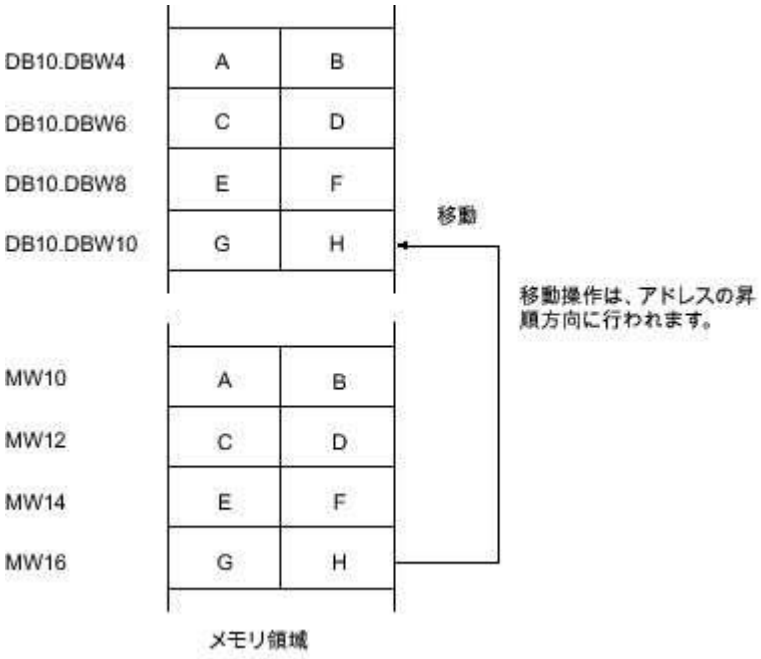
BLKMOV: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

次の図に、ムーブ操作の原理を示します。



ソース領域および宛先領域の整合性

「ブロックムーブ」命令の実行中にソースデータが変更されないことを確認してください。 変更があった場合、宛先データの整合性は保証できません。

割り込み機能

ネストレベルに対する制限はありません。

メモリ領域

「ブロックの移動」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力

- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。ソース領域と宛先領域の長さが異なる場合、小さな領域の長さのみが移動します。

ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

文字列の移動のルール

STRING データタイプの移動元および移動先の領域の移動に「ブロックの移動」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報も、宛先領域に書き込まれます。ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。

文字列の最大長および実際の長さに関する情報を移動するには、SRCBLK および DSTBLK パラメータにバイト単位で領域を指定します。

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
SRCBLK	Input	VARIANT	I、Q、M、L、P	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output [1)]	VARIANT	I、Q、M、L、P	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。					

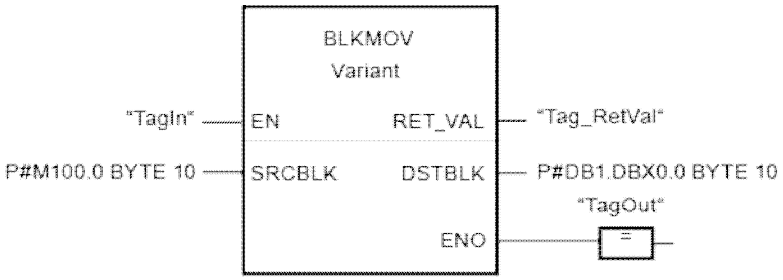
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみで使用できます。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。 ムーブ操作中にエラーが発生した場合、「Tag_RetVal」タグにエラーコードが出力されます。

UBLKMOV:割り込みなしブロック転送

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 その結果、「割り込み不可能なブロックムーブ」命令の実行中に、CPU の割り込み応答時間が長くなる場合があります。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

メモリ領域

「割り込みなしブロック転送」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

「割り込みなしブロック転送」命令の実行中は、ソース領域および宛先領域が重複してはなりません。ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

仮パラメータとして定義されたソース領域または宛先領域が、SRCBLK または DSTBLK パラメータで指定された宛先領域またはソース領域よりも小さい場合、データは転送されません。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

文字列の移動のルール

STRING データタイプのソース領域および宛先領域の移動に「割り込みなしブロック転送」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報は、宛先領域には書き込まれません。ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。 STRING データタイプの領域を移動する場合、領域の長さとして「1」を指定します。

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
SRCBLK	Input	VARIANT	I、Q、M、L、P	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output 1)	VARIANT	I、Q、M、L、P	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。					

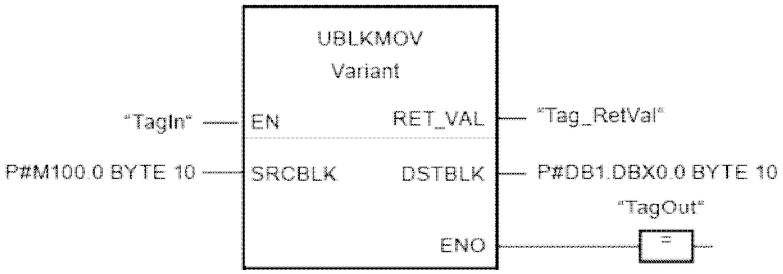
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
8091	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。 ムーブ操作中にエラーが発生した場合、「Tag_RetVal」タグにエラーコードが出力されます。

FILL:ブロックの塗りつぶし

説明

「フィル」命令を使用して、メモリ領域(宛先領域)を別のメモリ領域(ソース領域)の内容によって埋めることができます。 「フィル」命令は、宛先領域がすべて書き込まれるまでソース領域の内容を宛先領域に移動します。 ムーブ操作は、アドレスの昇順方向に行われます。

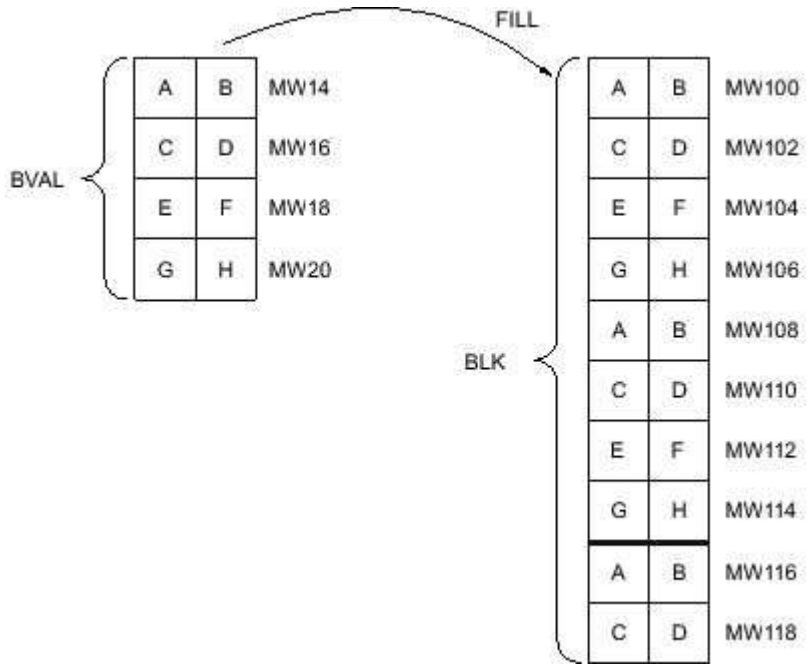
VARIANT を使用して、ソース領域と宛先領域を定義します。

注記

この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

「最適化したブロックアクセス」属性を持つブロックの場合、命令「FILL_BLK: フィル」命令を使用できます。

次の図に、ムーブ操作の原理を示します。



ソース領域および宛先領域の整合性

「フィル」命令の実行中は、宛先データの整合性を保証するため、ソースデータは変更されないことに注意してください。

メモリ領域

「フィル」命令を使用して、以下のメモリ領域を移動することができます。

- データブロックの領域
- ビットメモリ

- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。プリセットする宛先領域が BVAL 入力パラメータの長さの整数倍ではなくても、宛先領域は最後のバイトまで書き込まれます。

プリセットする宛先領域がソース領域よりも小さい場合は、ソース領域に含まれているデータのうち、宛先領域に書き込める量のみがコピーされます。

実際に存在する宛先領域またはソース領域がソース領域または宛先領域に割り当てられたメモリ領域よりも小さい場合(BVAL、BLK パラメータ)、データは転送されません。

ANY ポインタ(ソースまたは宛先)がデータタイプ BOOL の場合、これを絶対アドレス指定する必要があり、かつ指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行されません。

宛先領域が STRING データタイプの場合、この命令は管理情報を含む文字列全体を書き込みます。

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
BVAL	Input	VARIANT	I、Q、M、L、P	その内容に内容が BLK パラメータの宛先領域を埋めるのに使用されるメモリ領域(ソース領域)の指定。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
BLK	Output [1)]	VARIANT	I、Q、M、L、P	ソース領域の内容で埋められるメモリ領域の指定。
1) データがタグに流れるため、BLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

RET_VAL パラメータ

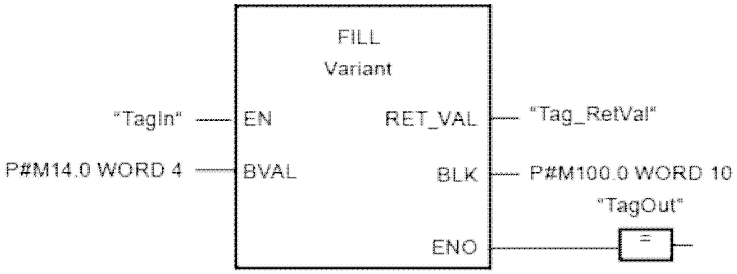
次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは BVAL パラメータでサポートされていません。

8352	WSTRING、WCHAR および BOOL データタイプは BLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

次の例で、命令がどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、ソース領域 MW14 ~ MW20 をコピーし、BVAL パラメータのメモリ領域に含まれる 4 ワードの内容で宛先領域 MW100 ~ MW118 を埋めます。

変換操作

この章には下記に関する情報が記載されています：

CONVERT: 変換データ

説明

「データの変換」命令は、IN パラメータの内容を読み出し、命令ボックスで設定されたデータタイプに従って変換します。 変換された値は、OUT 出力で出力されます。

実行可能な変換に関する情報は、「関連項目」の「明示的な変換」のセクションを参照してください。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

ビット列の変換オプション

ビット列 BYTE および WORD は、命令ボックスで選択できません。 ただし、入力オペランドと出力オペランドの長さが一致する場合は、命令のパラメータでデータタイプ DWORD または LWORD のオペランドを指定できます。 次にオペランドは、入力パラメータまたは出力パラメータのデータタイプに応じてビット列のデータタイプから暗黙的に変換されます。 たとえば、データタイプ DWORD は DINT/UDINT として解釈され、LWORD は LINT/ULINT として解釈されます。 これらの変換オプションは[IEC チェック]が有効な場合でも使用可能です。

注記
S7-1500 シリーズの CPU の場合: データタイプ DWORD および LWORD は、データタイプ REAL または LREAL との間でのみ変換できます。

パラメータ

次の表に、「値の変換」命令のパラメータを示します。

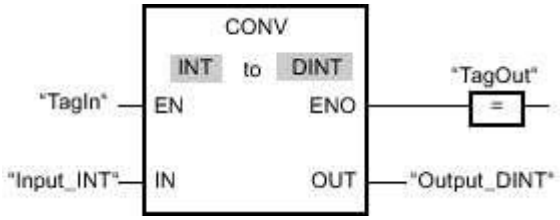
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数、浮動小数点数、CHAR、WCHAR、BCD16、BCD32	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	変換する値
OUT	Output	ビット列、整数、浮動小数点数、CHAR、WCHAR、BCD16、BCD32	I、Q、M、D、L、P	I、Q、M、D、L、P	変換の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

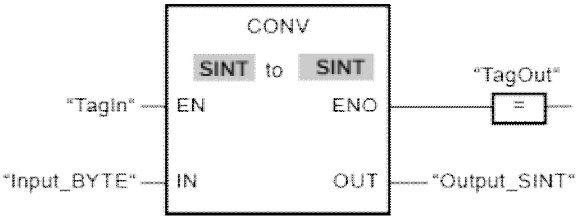
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

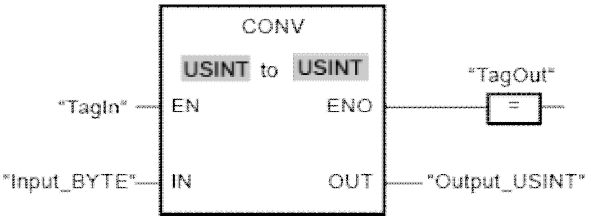
次の例は、整数(16 ビット)から別の整数(32 ビット)への変換を示します。



次の例は、 バイト(8 ビット)から整数 SINT (8 ビット)への変換を示します。



次の例は、バイト(8 ビット)から符号なし整数 USINT (8 ビット)への変換を示します。



これらの変換は、2 つのオペランドの長さが同じであるため可能です。

ROUND:数値の四捨五入

説明

「数値の四捨五入」命令を使用して、入力 IN の値を近似の整数に四捨五入することができます。 この命令は、入力 IN の値を浮動小数点数として解釈し、これを最も近い整数に変換します。 入力値が偶数と奇数のちょうど間にある場合、偶数が変換されます。 命令の結果は、OUT 出力で出力され、そこで照会できます。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「数値の四捨五入」命令のパラメータを示します。

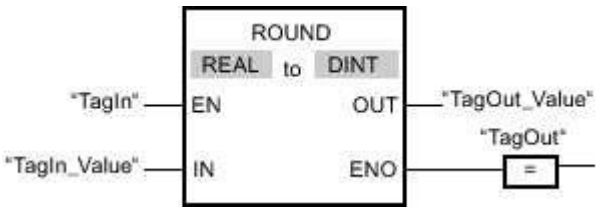
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	四捨五入される入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	四捨五入の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	1.50000000	-1.50000000

OUT	TagOut_Value	2	-2

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」の浮動小数点数は、最も近い偶数の整数に四捨五入され、出力「TagOut_Value」に送信されます。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

CEIL: 浮動小数点数から次に大きい整数を生成

説明

「浮動小数点数から次に大きい整数を生成」命令を使用して、入力 IN の値を次に大きい整数に切り上げることができます。 この命令は、入力 IN の値を浮動小数点数として解釈し、これを次に大きい整数に変換します。 命令の結果は、OUT 出力で出力され、そこで照会できます。 出力値は、入力値以上になることがあります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「浮動小数点数から次に大きい整数を生成」命令のパラメータを示します。

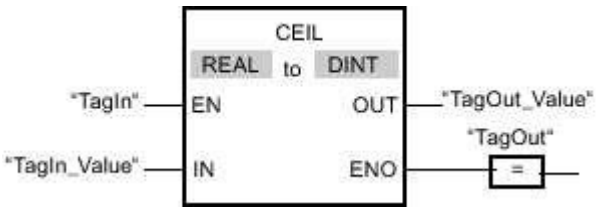
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	浮動小数点数としての入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	次に大きい整数の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	0.50000000	-0.50000000

OUT	TagOut_Value	1	0

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」が次に大きな整数に切り上げられ、出力「TagOut_Value」に送信されます。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

FLOOR: 浮動小数点数から次に小さい整数を生成

説明

「浮動小数点数から次に小さい整数を生成」命令を使用して、入力 IN の値を次に小さい整数に切り下げることができます。 この命令は、入力 IN の値を浮動小数点数として解釈し、浮動小数点数の値を次に小さい整数に変換します。 命令の結果は、OUT 出力で提供され、そこで照会できます。 出力値は、入力値以下になります。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「浮動小数点数から次に小さい整数を生成」命令のパラメータを示します。

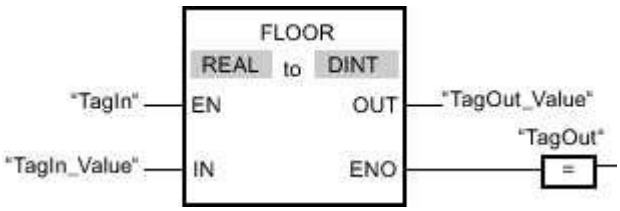
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	浮動小数点数としての入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	I、Q、M、D、L、P	次に小さい整数の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	0.50000000	-0.50000000

OUT	TagOut_Value	0	-1

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」の浮動小数点数は、次に小さい整数に切り下げられ、出力「TagOut_Value」に送信されます。この命令がエラーなしで実行された場合、「TagOut」出力がセットされます。

TRUNC: 値の切り捨て

説明

「数値を切り捨てる」命令を使って、入力 IN の値から整数を形成します。 入力 IN の値は、浮動小数点数として解釈されます。 この命令は、浮動小数点数の整数部分のみを選択し、これを小数点なしで出力 OUT に送信します。

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 実行中にオーバーフローなどのエラーが発生した。

パラメータ

次の表に、「数値を切り捨てる」命令のパラメータを示します。

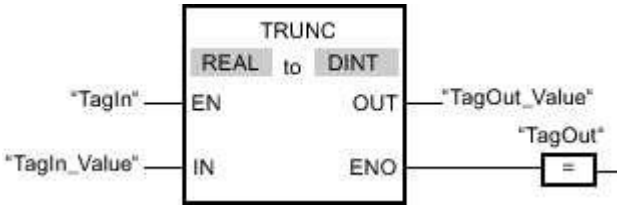
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
IN	Input	浮動小数点数	I、Q、M、D、L、または定数	浮動小数点数としての入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L	浮動小数点数の整数部分の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	1.50000000	-1.50000000
OUT	TagOut_Value	1	-1

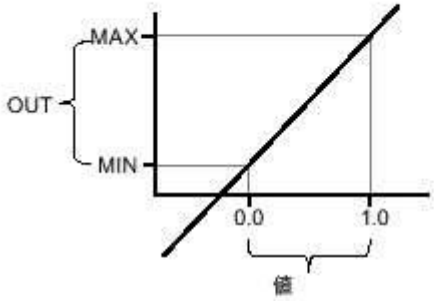
「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「TagIn_Value」の浮動小数点数の整数部分が整数に変換され、出力「TagOut_Value」に送信されます。この命令が エラーなしで実行された場合、「TagOut」出力がセットされます。

SCALE_X: スケール

説明

「スケール」命令を使用して、VALUE 入力の値を指定した値の範囲にマッピングしてスケーリングすることができます。 「スケール」命令が実行されると、入力 VALUE の浮動小数点値が、パラメータ MIN および MAX によって定義された値の範囲にスケーリングされます。 スケーリングの結果は整数で、これは出力 OUT に格納されます。

次の図に、値をどのようにスケーリングできるかを示します。



「スケール」命令は、以下の等式で機能します。

OUT = [VALUE * (MAX – MIN)] + MIN

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 MIN の値が入力 MAX の値以上であること。
- 指定した浮動小数点数の値が、IEEE-754 に従い正規化された数の範囲外であること。
- オーバーフローが発生していること。
- 入力 VALUE の値が NaN (非数値 = 無効な数学演算の結果)あること。

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
MIN	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE	Input	浮動小数点数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	スケーリングされる値 定数を入力する場合は、それを宣言する必要があります。
MAX	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値の範囲の上限値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L	I、Q、M、D、L	スケーリングの結果

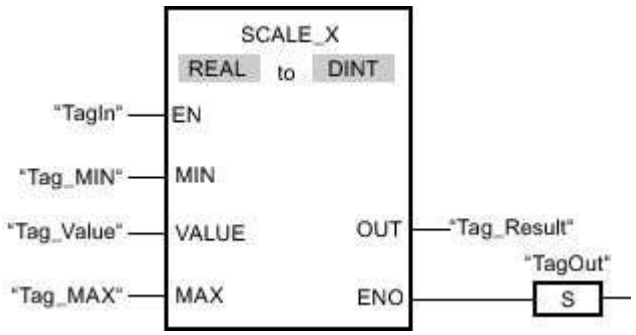
命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_MIN	10
VALUE	Tag_Value	0.5
MAX	Tag_MAX	30
OUT	Tag_Result	20

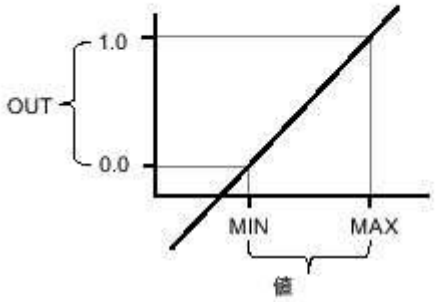
「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「Tag_Value」の値が、入力「Tag_MIN」および「Tag_MAX」の値で定義された値の範囲にスケーリングされます。結果は出力「Tag_Result」に格納されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

NORM_X: 正規化

説明

「正規化」命令を使用して、VALUE 入力のタグの値をリニアスケーリングにマッピングして正規化することができます。 MIN および MAX パラメータを使用し、スケーリングに適用する値の範囲の限界を定義できます。 この値の範囲の正規化された値の位置に応じて、出力 OUT で結果が計算され、浮動小数点数として格納されます。 正規化する値が入力 MIN の値と等しい場合、出力 OUT が値「0.0」を返します。 正規化する値が入力 MAX の値と等しい場合、出力 OUT が値「1.0」を返します。

次の図に、値が正規化される例を示します。



「正規化」命令は、以下の等式で機能します。

OUT = (VALUE – MIN) / (MAX – MIN)

次のいずれかの条件が満たされると、許可出力 ENO のシグナル状態が「0」になります。

- 許可入力 EN のシグナル状態が「0」であること。
- 入力 MIN の値が入力 MAX の値以上であること。
- 指定した浮動小数点数の値が、IEEE-754 に従い正規化された数の範囲外であること。
- 入力 VALUE の値が NaN (無効な数学演算の結果)であること。

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

パラメータ

次の表に、「正規化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
MIN [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	正規化する値
MAX [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値の範囲の上限値
OUT	Output	浮動小数点数	I、Q、M、D、L	I、Q、M、D、L	正規化の結果
[1)] これらの 3 つのパラメータで定数を使用する場合は、そのいずれかを宣言するのみで済みます。					

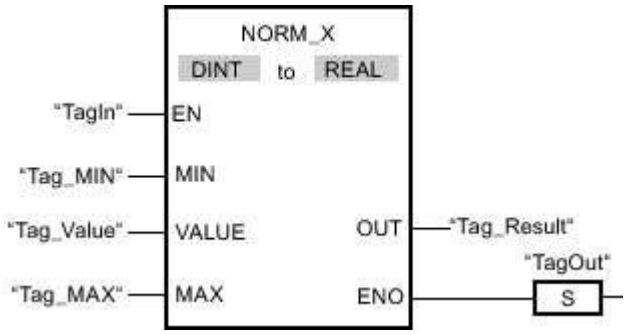
命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_MIN	10
VALUE	Tag_Value	20
MAX	Tag_MAX	30
OUT	Tag_Result	0.5

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 入力「Tag_Value」の値が、入力「Tag_MIN」および「Tag_MAX」の値で定義された値の範囲に割り当てられます。 入力「Tag_Value」のタグ値が定義された値の範囲に対応して正規化されます。 結果は出力「Tag_Re-

sult」で浮動小数点数として格納されます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」となり、「TagOut」出力がセットされます。

レガシー

この章には下記に関する情報が記載されています：

- SCALE:スケール (S7-1500)
- UNSCALE:スケール解除 (S7-1500)

SCALE:スケール

説明

「"スケール」命令を使用して、IN パラメータの整数を下限値と上限値間の物理単位でスケーリングが可能な浮動小数点数に変換します。LO_LIM および HI_LIM パラメータを使用し、入力値をスケーリングする範囲の下限値と上限値を指定します。 命令の結果は、OUT パラメータに出力されます。

「スケール」命令は、以下の等式で機能します。

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は-27648.0 となり、定数「K2」の値は+27648.0 となります。
- シグナル状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は 0.0 となり、定数「K2」の値は+27648.0 となります。

IN パラメータの値が定数値「K2」よりも大きい場合、命令の結果が上限値(HI_LIM)にセットされ、エラーが出力されます。

IN パラメータの値が定数値「K1」未満の場合、命令の結果が下限値(LO_LIM)にセットされ、エラーが出力されます。

示された下限値が上限値よりも大きい場合(LO_LIM > HI_LIM)、結果は入力値に逆の割合でスケーリングされます。

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	INT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	スケーリングする入力値。
HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解

					釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
OUT	Output	REAL	I、Q、M、D、L、P	I、Q、M、D、L、P	命令の結果
RET_VAL	Output	WORD	I、Q、M、D、L、P	I、Q、M、D、L、P	エラー情報

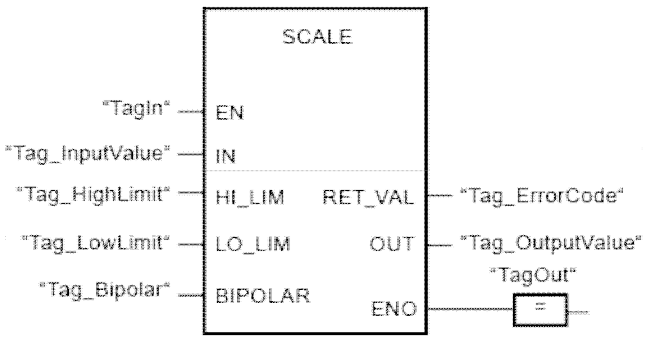
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が「K2」定数値を超えているか、または「K1」定数値未満になっています。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0

LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

UNSCALE:スケール解除

説明

「スケール解除」命令は、IN パラメータの浮動小数点数を下限値と上限の間の物理的単位にスケーリング解除し、それらを整数に変換します。LO_LIM および HI_LIM パラメータを使用し、入力値をスケーリング解除する範囲の下限値と上限値を指定します。 命令の結果は、OUT パラメータに出力されます。

「スケール解除」命令は、以下の等式で機能します。

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は-27648.0 となり、定数「K2」の値は+27648.0 となります。
- シグナル状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は 0.0 となり、定数「K2」の値は+27648.0 となります。

IN パラメータの値が定数値「HI_LIM」よりも大きい場合、命令の結果が定数値(K2)にセットされ、エラーが出力されます。

IN パラメータの値が下限値の定数値「LO_LIM」未満の場合、命令の結果が定数値(K1)にセットされ、エラーが出力されます。

パラメータ

次の表に、「スケール解除」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	REAL	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	整数値にスケーリング解除する入力値。
HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラ

					として解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
OUT	Output	INT	I、Q、M、D、L、P	I、Q、M、D、L、P	命令の結果
RET_VAL	Output	WORD	I、Q、M、D、L、P	I、Q、M、D、L、P	エラー情報

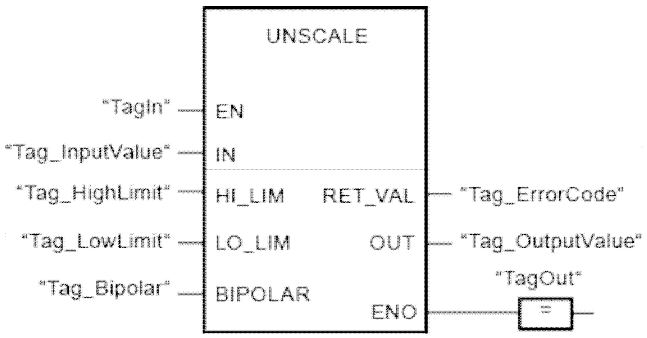
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が上限値(HI_LIM)を超えているか、または下限値(LO_LIM)未満になっています。
一般エラー情報	関連項目:"GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0

BIPOLAR	Tag_Bipolar	1
OUT	Tag_OutputValue	22
RET_VAL	Tag_ErrorCode	W#16#0000

プログラム制御演算

この章には下記に関する情報が記載されています：

JMP: RLO = 1 ならばジャンプ

説明

「RLO = 1 ならばジャンプ」命令を使用して、プログラムのリニア実行を中断し、別のネットワークで再開できます。 宛先ネットワークをジャンプラベル(LABEL)で識別する必要があります。 ジャンプラベルの記述は、命令ボックスの上にあるプレースホルダに入力します。

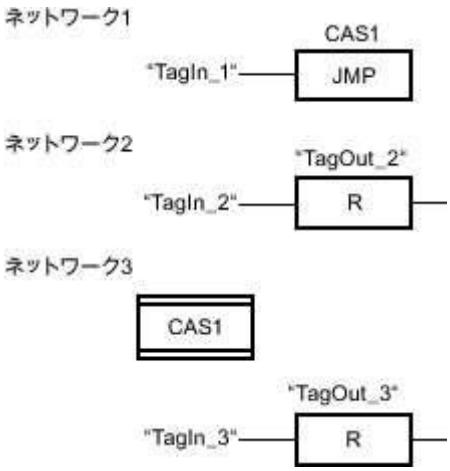
指定したジャンプラベルは、命令が実行されるブロックと同じブロックにあることが必要です。 指定する名前はブロック内で固有であることが必要です。 1 つのネットワークで使用できるジャンピングコイルは 1 つのみです。

命令の入力でのボックス入力の論理演算の結果(RLO)が「1」の場合、ジャンプラベルで識別されたネットワークへのジャンプが実行されます。 ジャンプ方向は、大きいまたは小さいネットワーク番号の方向になります。

命令の入力での条件が満たされない場合(RLO = 0)、プログラムの実行は次のネットワークで続行されます。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 プログラムの線形実行が中断され、ジャンプラベル CAS1 で識別されるネットワーク 3 で続行されます。 入力「TagIn_3」のシグナル状態が「1」の場合、出力「TagOut_3」がリセットされます。

JMPN: RLO = 0 ならばジャンプ

説明

「RLO = 0 ならばジャンプ」命令を使用し、命令の入力での論理演算の結果が「0」の場合、プログラムの線形実行に割り込み、それを別のネットワークで再開することができます。 宛先ネットワークをジャンプラベル(LABEL)で識別する必要があります。 ジャンプラベルの記述は、命令ボックスの上にあるプレースホルダに入力します。

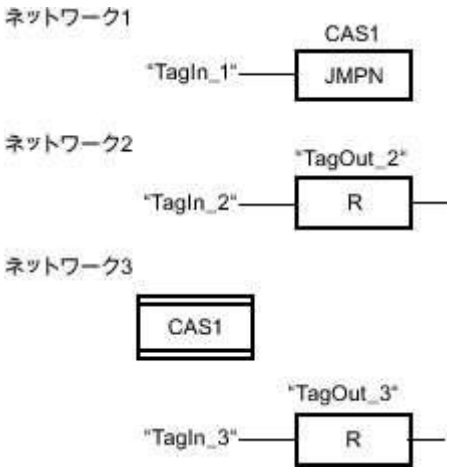
指定したジャンプラベルは、命令が実行されるブロックと同じブロックにあることが必要です。 指定する名前はブロック内で固有であることが必要です。 1 つのネットワークで使用できるジャンピングコイルは 1 つのみです。

命令の入力でのボックス入力の論理演算の結果(RLO)が「0」の場合、ジャンプラベルで識別されたネットワークへのジャンプが実行されます。 ジャンプ方向は、大きいまたは小さいネットワーク番号の方向になります。

命令の入力での論理演算の結果 RLO が「1」の場合、プログラムの実行は、次のネットワークで継続します。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「0」の場合、この命令が実行されます。プログラムの線形実行が中断され、ジャンプラベル CAS1 で識別されるネットワーク 3 で続行されます。入力「TagIn_3」のシグナル状態が「1」の場合、出力「TagOut_3」がリセットされます。

LABEL: ジャンプラベル

説明

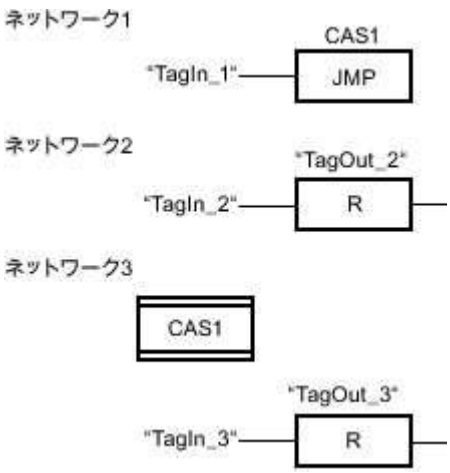
ジャンプラベルは、ジャンプ命令の実行後にプログラムの実行を再開できる宛先ネットワークを識別します。

ジャンプラベルとジャンプラベルを指定する命令は、同じブロックに置く必要があります。 ジャンプラベルの名前は、ブロック内で 1 回のみ割り当てることができます。 CPU S7-1200 を使用している場合は最大で 32 のジャンプラベル、CPU S7-1500 を使用している場合は最大で 256 のジャンプラベルを宣言できます。

ネットワークに置くことのできるジャンプラベルは 1 つのみです。 各ジャンプラベルから複数の位置にジャンプできます。

例

次の例で、命令がどのように動作するかを示します。



「TagIn_1」オペランドのシグナル状態が「1」の場合、この命令が実行されます。プログラムの線形実行が中断され、ジャンプラベル CAS1 で識別されるネットワーク 3 で続行されます。入力「TagIn_3」のシグナル状態が「1」の場合、出力「TagOut_3」がリセットされます。

JMP_LIST: ジャンプリストの定義

説明

「ジャンプリストの定義」を使って、複数の条件ジャンプを定義できます。また、K パラメータの値に基づいて、指定したネットワーク内でプログラム実行を継続できます。

ジャンプはジャンプラベル(LABEL)を使用して定義します。ジャンプラベルは、命令ボックスの出力で指定します。 出力の数は、命令ボックスで拡張できます。 CPU S7-1200 を使用している場合は最大で 32 の出力、CPU S7-1500 を使用している場合は最大で 99 の出力を宣言できます。

出力の番号付けは、値「0」から開始され、それぞれの新しい出力で昇順に継続されます。 命令の出力では、ジャンプラベルのみを指定できます。 命令またはオペランドの指定は許可されません。

K パラメータの値は、出力の番号、およびプログラム実行が再開されるジャンプラベルを指定します。 K パラメータの値が使用可能な出力の数よりも大きい場合、ブロックの次のネットワークでプログラム実行が再開されます。

「ジャンプリストの定義」命令は、EN 許可入力のシグナル状態が「1」の場合にのみ実行されます。

パラメータ

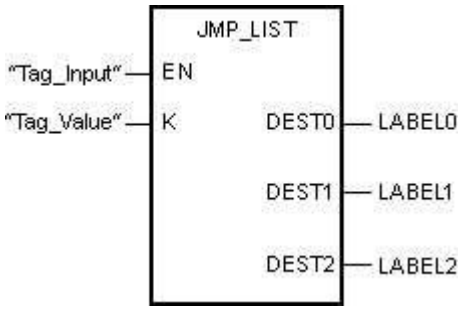
次の表に、「ジャンプリストの定義」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、L、D	許可入力
K	Input	UINT	I、Q、M、L、D、または定数	出力の番号、ひいては実行されるジャンプを指定します。
DEST0	-	-	-	最初のジャンプラベル
DEST1	-	-	-	2 番目のジャンプラベル
DESTn	-	-	-	オプションのジャンプラベル

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド/ジャンプラベル	値
K	Tag_Value	1
DEST0	LABEL0	ジャンプラベル「LABEL0」で識別されるネットワークにジャンプします。
DEST1	LABEL1	ジャンプラベル「LABEL1」で識別されるネットワークにジャンプします。
DEST2	LABEL2	ジャンプラベル「LABEL2」で識別されるネットワークにジャンプします。

「Tag_Input」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 プログラムの実行は、ジャンプラベル「LABEL1」で識別されるネットワークのオペランド「Tag_Value」の値に基づいて継続されます。

SWITCH: ディストリビュータにジャンプ

説明

「ディストリビュータジャンプ」命令を使用して複数のプログラムジャンプを定義し、1 つ以上の比較命令の結果に応じて実行することができます。

パラメータ K で、比較する値を指定します。 この値は、個々の入力が返す値と比較されます。 それぞれの入力で、比較のタイプを選択します。 比較命令の使用可能性は、命令のデータタイプによって異なります。

次の表に、選択されたデータタイプに応じて使用可能な比較命令を示します。

データタイプ		命令	構文
S7-1200	S7-1500		
ビット列	ビット列	等価	==
		不等価	<>
整数、浮動小数点数、TIME、DATE、TOD	整数、浮動小数点数、TIME、LTIME、DATE、TOD、LTOD、LDT	等価	==
		不等価	<>
		以上	>=
		以下	<=
		超過	>
		未満	<

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。 比較命令を選択し、その命令のデータタイプがまだ定義されていない場合、「???」ドロップダウンリストには、選択した比較命令で許可されたデータタイプが表示されます。

命令の実行は、最初の比較から開始され、比較条件が満たされるまで実行されます。 比較条件が満たされている場合、それ以後の比較条件は考慮されません。 指定した条件のいずれも満たされていない場合、出力 ELSE でジャンプが実行されます。 出力 ELSE でジャンプラベルが定義されていない場合、プログラムの線形実行は割り込まれず、次のネットワークで続行します。

初期状態では、この命令ボックスには 2 つ以上の出力(DEST0 および DEST1)が含まれています。 出力数は拡張できます。 出力の番号付けは、値「0」から開始され、それぞれの新しい出力で昇順に継続されます。 命令の出力でジャンプラベル(LABEL)を指定します。 命令の出力での命令またはオペランドの指定は許可されません。

追加された出力に対し、自動的に入力が挿入されます。 出力でプログラミングされたジャンプは、対応する比較条件が満たされると実行されます。

パラメータ

次の表に、「ディストリビュータジャンプ」命令のパラメータを示します。

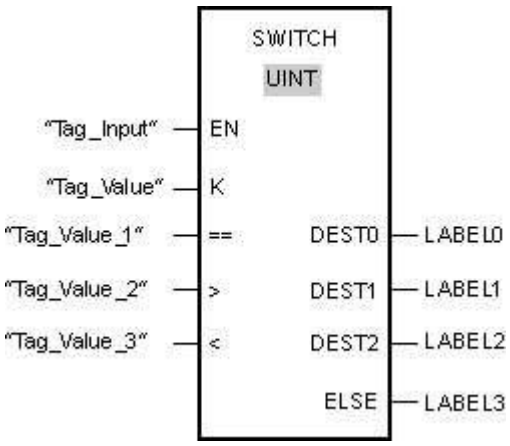
パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
EN	Input	BOOL	BOOL	I、Q、M、D、L	許可入力

K	Input	UINT	UINT	I、Q、M、D、L、または定数	比較する値を指定します。
<Comparison values>	Input	ビット列、整数、浮動小数点数、TIME、DATE、TOD	ビット列、整数、浮動小数点数、TIME、LTIME、DATE、TOD、LTOD、LDT	I、Q、M、D、L、または定数	パラメータ K の値と比較される入力値
DEST0	-	-	-	-	最初のジャンプラベル
DEST1	-	-	-	-	2 番目のジャンプラベル
DEST(n)	-	-	-	-	オプションのジャンプラベル (n = 2 ~ 99)
ELSE	-	-	-	-	いずれの比較条件も満たされない場合に実行されるプログラムジャンプ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド/ジャンプラベル	値
K	Tag_Value	23
==	Tag_Value_1	20
>	Tag_Value_2	21
<	Tag_Value_3	19

DEST0	LABEL0	パラメータ K の値が 20 と等しい場合、ジャンプラベル「LABEL0」にジャンプ。
DEST1	LABEL1	パラメータ K の値が 21 よりも大きい場合、ジャンプラベル「LABEL1」にジャンプ。
DEST2	LABEL2	パラメータ K の値が 19 未満の場合、ジャンプラベル「LABEL2」にジャンプ。
ELSE	LABEL 3	いずれの比較条件も満たされない場合、ジャンプラベル「LABEL3」にジャンプ。

「Tag_Input」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 ジャンプラベル「LABEL1」で識別されるネットワークでプログラムの実行を継続。

RET: 戻り値

説明

「戻り値」命令を使って、ブロックの実行を停止します。 この結果は、ブロック処理を完了することができ 3 つのタイプになります。

- 「戻り値」命令の呼び出しなし

  最後のネットワークの実行後にブロックが終了します。 呼び出しファンクションの ENO のシグナル状態が「1」にセットされます。
- 論理演算による「戻り値」命令の呼び出し(例を参照)

  左コネクタのシグナル状態が「1」の場合、ブロックが終了されます。ファンクション呼び出しの ENO がオペランドに対応します。
- 論理演算なしの「戻り値」命令の呼び出し

  ブロックが終了されます。ファンクション呼び出しの ENO がオペランドに対応します。

> **注記**
> 1 つのネットワークで使用できるジャンピングコイルは 1 つのみです(「戻り値」、「RLO=1 ならばジャンプ」、「RLO=0 ならばジャンプ」)。

「戻り値」命令の入力で論理演算(RLO)の結果が「1」の場合、現在呼び出されているブロックでプログラムの実行が終了され、呼び出し元ブロックの呼び出しファンクションの後で続行されます(たとえば、呼び出し元 OB で)。 呼び出しファンクションのステータス(ENO)は、命令のパラメータで決定されます。 以下の値が想定されます。

- [RLO]
- TRUE/FALSE
- <Operand>

パラメータ値を設定するには、命令をダブルクリックし、ドロップダウンリストで対応する値を選択します。

次の表に、呼び出したブロック内のネットワークで「戻り値」命令がプログラミングされた場合の呼び出しファンクションのステータスを示します。

[RLO]	パラメータ値	呼び出しファンクションの ENO
1	[RLO]	1
	TRUE	1
	FALSE	0
	<Operand>	<Operand>
0	[RLO]	この場合、プログラムの実行は、呼び出されたブロックの次のネットワークで続行します。
	TRUE	
	FALSE	
	<Operand>	

OB が完了した場合、優先度クラスシステムによって別のブロックが選択され、開始されるか、再実行されます。

- OB プログラムサイクルが完了された場合、再開されます。
- 別のブロックに割り込んだ OB(たとえばアラーム OB)が完了すると、割り込まれたブロック(たとえば OB プログラムサイクル)が実行されます。

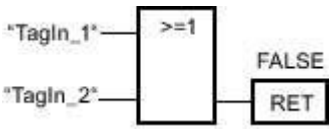
パラメータ

次の表に、「戻り値」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
				RLO = 1 の場合の呼び出し元ファンクションのステータス:
[RLO]	-	-	-	RLO のシグナル状態にセットされます。
TRUE	-	-	-	1
FALSE	-	-	-	0
<Operand>	Input	BOOL	I、Q、M、D、L	指定したオペランドのシグナル状態

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」または「TagIn_2」のいずれかのシグナル状態が「1」の場合、命令が実行されます。 呼び出されたブロックのプログラム実行は終了され、呼び出し元のブロックで続行します。呼び出しファンクションの許可出力 ENO は、シグナル状態「0」にリセットされます。

ランタイム制御

この章には下記に関する情報が記載されています：

- ENDIS_PW: パスワードの適正化の制限および有効化 (S7-1200, S7-1500)
- RE_TRIGR: サイクルタイムモニタのリスタート (S7-1200, S7-1500)
- STP: プログラム終了 (S7-1200, S7-1500)
- GET_ERROR: ローカルでエラーを取得 (S7-1200, S7-1500)
- GET_ERR_ID: ローカルでエラー ID を取得 (S7-1200, S7-1500)
- INIT_RD: すべての保持データの初期化 (S7-1500)
- WAIT: 遅延時間の設定 (S7-1500)
- RUNTIME: プログラムランタイムの測定 (S7-1200, S7-1500)

ENDIS_PW: パスワードの適正化の制限および有効化

説明

「パスワード正当性の制限および有効化」命令を使用して、設定されたパスワードを CPU に対して正当化できるかどうかを指定します。 このため、正しいパスワードがわかっている場合でも、正規の接続を防止することができます。

命令呼び出し時に REQ パラメータのシグナル状態が「0」の場合、現在設定されている状態が、出力パラメータに表示されます。 入力パラメータに変更を加えても、変更は出力パラメータに転送されません。

命令呼び出し時に REQ パラメータのシグナル状態が「1」の場合、シグナル状態は、入力パラメータ(F_PWD、FULL_PWD、R_PWD、HMI_PWD)から取得されます。FALSE は、パスワードごとの正当化が許可されていないことを意味します。TRUE は、パスワードを使用できることを意味します。

パスワードの有効化または無効化を個別に許可または禁止することができます。 たとえば、フェールセーフパスワードを除く、すべてのパスワードを禁止することができます。 したがって、アクセスオプションを小さいユーザーグループに制限することができます。 出力パラメータ(F_PWD_ON、FULL_PWD_ON、R_PWD_ON、HMI_PWD_ON)は、REQ パラメータに関係なく、常にパスワード使用の現在のステータスを示します。

設定済みでないパスワードの場合、入力のシグナル状態が TRUE でなければならず、出力にシグナル状態 TRUE を返します。 フェールセーフパスワードは、F-CPU の場合のみ設定できるため、標準 CPU では、常にシグナル状態 TRUE と相互接続する必要があります。 命令がエラーを返す場合、この呼び出しには効果がなく、直前のロックがまだ有効です。

無効なパスワードは、以下の条件で再度有効にすることができます。

- CPU が工場出荷時設定にリセットされること。
- S7-1500 CPU のフロントパネルが、パスワードを再度有効にすることができる適切なメニューに移動することができるダイアログをサポートしていること。
- 「パスワードの適正化の制限および有効化」命令を呼び出す場合、目的のパスワードの入力パラメータのシグナル状態が「1」であること。
- モードセレクタを STOP に設定すること。 このスイッチの設定を RUN に戻すと直ちに、パスワードの適正化に対する制限が再確立されます。
- S7-1200 CPU への空きメモリカード(転送モジュールまたはプログラムカード)の差し込み。
- S7-1200 CPU では、電源オフから電源オンへの移行によって、保護が無効になります。 この場合、プログラム(たとえば、スタートアップ OB)で、再度、「パスワード正当性の制限および有効化」命令を呼び出す必要があります。

注記
「パスワード正当性の制限および有効化」命令は、HMI パスワードが有効にされていない場合、HMI システムからのアクセスをブロックします。

注記
既存の適正化された接続はアクセス権を保持し、「パスワードの適正化の制限および有効化」命令を使用してそれらを制限することはできません。

S7-1500 CPU での意図しないロックアウトの防止

CPU のフロントパネルで設定を行うことができ、CPU は最新の設定を保存します。

偶発的なロックアウトを防止するには、S7-1500 CPU でモードセレクタを STOP に設定することによって保護を無効にすることができます。 モードセレクタを RUN に設定することによって、「パスワ

ードの適正化の制限および有効化」命令を再度呼び出したり、フロントパネルで追加の操作を実行したりせずに、保護が自動的に再度有効になります。

S7-1200 CPU での意図しないロックアウトの防止

S7-1200 CPU にはモードスイッチが存在しないため、電源オフから電源オンへの移行時の保護は無効です。 このため、ユーザープログラム内の特定のプログラムシーケンスによって偶発的なロックアウトを防止することをお奨めします。

これを行うには、サイクリック割り込み OB、またはメイン OB (OB 1)のタイマのいずれかを使用して、時間制御をプログラミングします。 これによって、電源オフから電源オンへの移行と、関連する保護の無効化の後、比較的迅速に、各 OB (たとえば、OB 1 または OB 35)内で「パスワード正当性の制限および有効化」命令を呼び出すオプションが可能になります。 スタートアップ OB (OB 100)でこの命令を呼び出して、この命令が無効で、パスワード正当化の制限が存在しない時間帯を比較的小さなものに保ちます。 この手順は、未許可のアクセスに対する可能な最大限の保護を提供します。

偶発的なロックアウトが発生した場合は、スタートアップ OB での呼び出しをスキップし(たとえば、入力パラメータを照会することによって)、ロックが再び有効になる前に、CPU へ接続確立する時間(たとえば、10 秒～1 分)を設定することができます。

ユーザープログラムコードにタイマが存在しない場合にロックアウトが発生した場合は、空の転送カードまたはプログラムカードを CPU に挿入します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 この後、ユーザープログラムを再び STEP 7 から CPU にダウンロードする必要があります。

S7-1200 CPU でパスワードが失われた場合の手順

パスワード保護された S7-1200 CPU のパスワードを紛失した場合は、空の転送カードまたはプログラムカードを使用して、パスワード保護されたプログラムを削除します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 その後、新しいユーザープログラムを STEP 7 Basic から CPU にロードすることができます。

警告

空の転送カードの挿入

ランタイム中に CPU に転送カードを挿入すると、CPU は STOP モードに切り替わります。 動作状態が不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期しない動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

転送カードを取り出すと、転送カードの内容が内部ロードメモリで使用可能になります。 この時点に、カードにプログラムが含まれていないことを確認してください。

警告

空のプログラムカードの挿入

ランタイム中に CPU にプログラムカードを挿入すると、CPU は STOP モードに移行します。 動作状態が不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期しない動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

プログラムカードが空であることを確認してください。 内部ロードメモリが空のプログラムカードにコピーされます。 直前に空であったプログラムカードを取り出すと、内部ロードメモリは空になります。

CPU を RUN モードに切り替える前に、転送カードまたはプログラムカードを取り出す必要があります。

動作モードへのパスワードの使用の影響

次の表に、「パスワードの適正化の制限および有効化」命令によるパスワードの使用が動作モードと対応するユーザーの操作に与える影響を示します。

操作	命令によるパスワードの保護
その後の基本状態 • STOP への動作モードの切り替え • メモリの手動リセット(PG、切り替え、MC (モーションコントロール)の変更) • 工場出荷時設定へのリセット	無効 (制限なし)
電源オン後の基本状態	• S7-1200-CPU: ロックは無効で、プログラム(たとえば、スタートアップ OB)で再びこの命令を呼び出す必要があります。 • S7-1500 CPU: 有効(電源オフの前にロックが有効になった場合). パスワードの不許可のオプションは保持型です。
動作モードの移行 RUN/STARTUP/HOLD -> STOP (命令、エラーまたは通信の終了によって)または STOP -> STARTUP/RUN/HOLD	有効 パスワードはまだ使用できません。

パラメータ

次の表に、「パスワードの適正化の制限および有効化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ パラメータのシグナル状態が「0」の場合は、現在設定されているパスワードのシグナル状態が照会されます。
F_PWD	Input	BOOL	I、Q、M、D、L、または定数	フェールセーフなどの読み取り/書き込みアクセス • F_PWD = "0": パスワードを許可しない • F_PWD = "1": パスワードを許可する
FULL_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取り/書き込みアクセス • FULL_PWD = "0": パスワードを許可しない • FULL_PWD = "1": パスワードを許可する
R_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取りアクセス • R_PWD = "0": パスワードを許可しない • R_PWD = "1": パスワードを許可する
HMI_PWD	Input	BOOL	I、Q、M、D、L、または定数	HMI アクセス • HMI_PWD = "0": パスワードを許可しない

				• HMI_PWD = "1": パスワードを許可する
F_PWD_ON	Output	BOOL	I、Q、M、D、L	フェールセーフなどの読み取り/書き込みアクセス • F_PWD_ON = "0": パスワードを許可しない • F_PWD_ON = "1": パスワードを許可する
FULL_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取り/書き込みアクセスステータス • FULL_PWD_ON = "0": パスワードを許可しない • FULL_PWD_ON = "1": パスワードを許可する
R_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取りアクセスステータス • R_PWD_ON = "0": パスワードを許可しない • R_PWD_ON = "1": パスワードを許可する
HMI_PWD_ON	Output	BOOL	I、Q、M、D、L	HMI アクセスステータス • HMI_PWD_ON = "0": パスワードを許可しない • HMI_PWD_ON = "1": パスワードを許可する
RET_VAL	Output	WORD	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	**説明**
0000	エラーは発生していません。
8090	「パスワードの適正化の制限および有効化」命令はサポートされていません
80D0	フェールセーフのパスワードが未設定です。 標準 CPU では、シグナル状態は TRUE であることが必要です。
80D1	読み取り/書き込みアクセスが未設定です
80D2	読み取りアクセスが未設定です
80D3	HMI アクセスが未設定です
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

RE_TRIGR: サイクルタイムモニタのリスタート

説明

「サイクルタイムモニタのリスタート」命令を使用して、CPU のサイクルタイムモニタを再起動することができます。 サイクルモニタリングタイムが、CPU の設定で設定した持続時間内に最初からやり直されます。

「サイクルタイムモニタのリスタート」命令は、すべてのブロック内で優先度に関わらず呼び出し可能です。

ハードウェア割り込み、診断割り込み、周期割り込みなど、さらに優先度の高いブロックで呼び出された場合、この命令は実行されず、許可出力 ENO がシグナル状態「0」にセットされます。

「サイクルタイムモニタのリスタート」命令は、呼び出しの数に関係なく期間内に完全に実行されます (最大プログラムサイクルの 10 倍)。 時間が期限切れになると、プログラムサイクルは延長できなくなります。

パラメータ

「サイクルタイムモニタのリスタート」命令には、パラメータはありません。

STP: プログラム終了

説明

「プログラムの終了」命令を使って、CPU を STOP モードに設定し、プログラムの実行を終了します。 RUN から STOP への変更の影響は CPU の構成によって異なります。

命令の入力の(RLO)が「1」の場合、CPU が STOP モードに移行し、プログラムの実行が終了します。 命令の出力のシグナル状態は評価されません。

命令の入力の RLO が「0」の場合、命令は実行されません。

パラメータ

「プログラムの終了」命令にはパラメータはありません。

GET_ERROR: ローカルでエラーを取得

説明

「ローカルでエラーを取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 システムがブロックの処理中にエラーを報告した場合、命令の最後の実行の後、このエラーがブロックの実行中に発生した最初の該当エラーの場合、詳細情報が、ERROR 出力のオペランドに格納されます。

ERROR 出力では、「ErrorStruct」システムデータタイプのオペランドのみを指定できます。 「ErrorStruct」システムデータタイプは、エラー情報が保存されている正確な構造体を指定します。 追加の命令を使用することで、この構造体とプログラムが適切な応答をするかを評価できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーに関するエラー情報が出力されます。

注記

ERROR 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。
- 許可出力 ENO を照会する。

「ローカルでエラーを取得」命令の許可出力 ENO は、許可入力 EN がシグナル状態「1」を返し、エラー情報が存在する場合のみ設定されます。 これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラーを取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記

「ローカルでエラーを取得」命令は、ブロック内のローカルエラー処理を可能にします。 「ローカルでエラーを取得」がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

パラメータ

次の表に、「ローカルでエラーを取得」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
ERROR	Output	ErrorStruct	D、L	エラー情報

データタイプ「ErrorStruct」

次の表に、「ErrorStruct」データタイプの構造体をリストします。

構造体コンポーネント	データタイプ	説明
ERROR_ID	WORD	エラー ID
FLAGS	BYTE	エラーがブロック呼び出し中に発生したのかどうかを示します。

			16#01: ブロック呼び出し中のエラー 16#00: ブロック呼び出し中にエラーなし
REACTION		BYTE	既定の応答 0: 無視(書き込みエラー) 1: 代替値「0」で続行(読み出しエラー) 2: 命令をスキップ(システムエラー)
CODE_ADDRESS		CREF	ブロックのアドレスおよびタイプに関する情報
	BLOCK_TYPE	BYTE	エラーが発生したブロックのタイプ: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	プログラムブロックの番号
	OFFSET	UDINT	内部メモリへの参照
MODE		BYTE	オペランドのアドレスに関する情報
OPERAND_NUMBER		UINT	マシンコマンドのオペランド番号
POINTER_NUMBER_LOCATION		UINT	(A) 内部ポインタ
SLOT_NUMBER_SCOPE		UINT	(B) 内部メモリの記憶領域
DATA_ADDRESS		NREF	オペランドのアドレスに関する情報
	AREA	BYTE	(C) メモリ領域 L: 16#40～4E、86、87、8E、8F、C0～CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84、85、8A、8B データタイプ DINT の直接編集可能なタグの範囲違反: 16#04
	DB_NUMBER	UINT	(D) データブロックの番号
	OFFSET	UDINT	(E) オペランドの相対アドレス

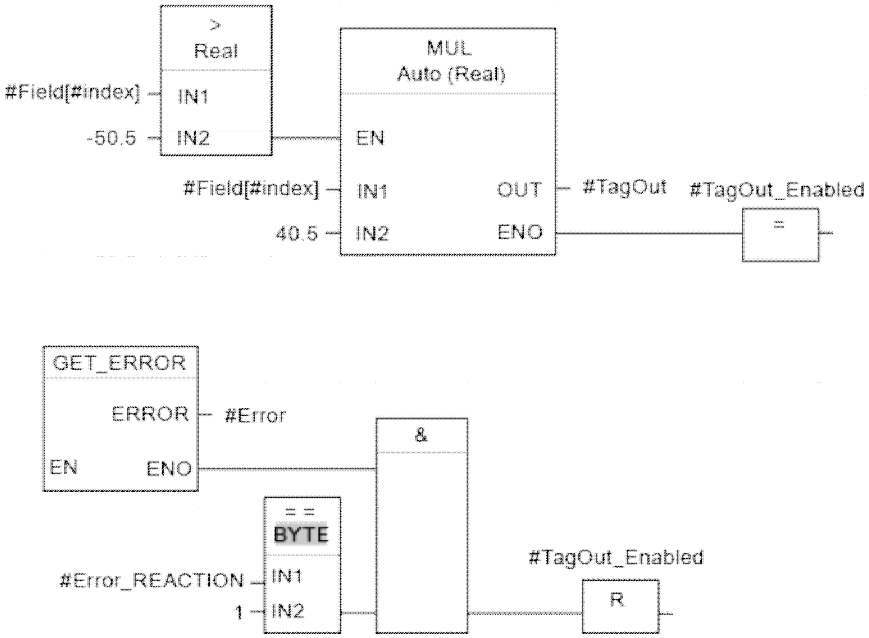
構造体コンポーネント「ERROR_ID」

次の表に、構造体コンポーネント「ERROR_ID」で出力可能な値をリストします。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、またはファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、またはファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー: 入力
2943	10563	書き込みエラー: 出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。		

例

次の例で、命令がどのように動作するかを示します。



#Field[#index]タグへのアクセスでエラーが発生しました。「乗算」命令および#TagOut_Enabled オペランドの許可出力 ENO のシグナル状態は、読み取り/アクセスエラーに関係なく「1」になり、乗算は「0.0」の値を使用して実行されます。 このエラーシナリオが発生した場合は、エラーを取得するために、「乗算」命令の後に「ローカルでエラーを取得」命令をプログラミングすることを推奨します。「ローカルでエラーを取得」命令によって供給されるエラー情報は、比較命令「等しい」を使用して評価されます。「#Error.REACTION」構造コンポーネントの値が「1」の場合、これは読み取り/アクセスエラーが関係しており、#TagOut_Enabled 出力がリセットされます。

GET_ERR_ID: ローカルでエラー ID を取得

説明

「ローカルでエラー ID を取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 この命令の最後の実行の後に、システムがブロック処理中にブロック実行に関するエラーを報告すると、タグで発生した最初のエラーのエラー ID が ID 出力に格納されます。

ID 出力では、WORD データタイプのオペランドのみを指定できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーのエラー ID が出力されます。

注記
ID 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。
- 許可出力 ENO を照会する。

命令「ローカルでエラー ID を取得」の許可出力 ENO は、許可入力 EN がシグナル状態「1」を返し、エラー情報が存在する場合のみ設定されます。 これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラー ID を取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラー ID を取得」命令は、ブロック内のローカルエラー処理を可能にします。「ローカルでエラー ID を取得」命令がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

パラメータ

次の表に、「ローカルでエラー ID を取得」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
ID	Output	WORD	I、Q、M、D、L	エラー ID

ID パラメータ

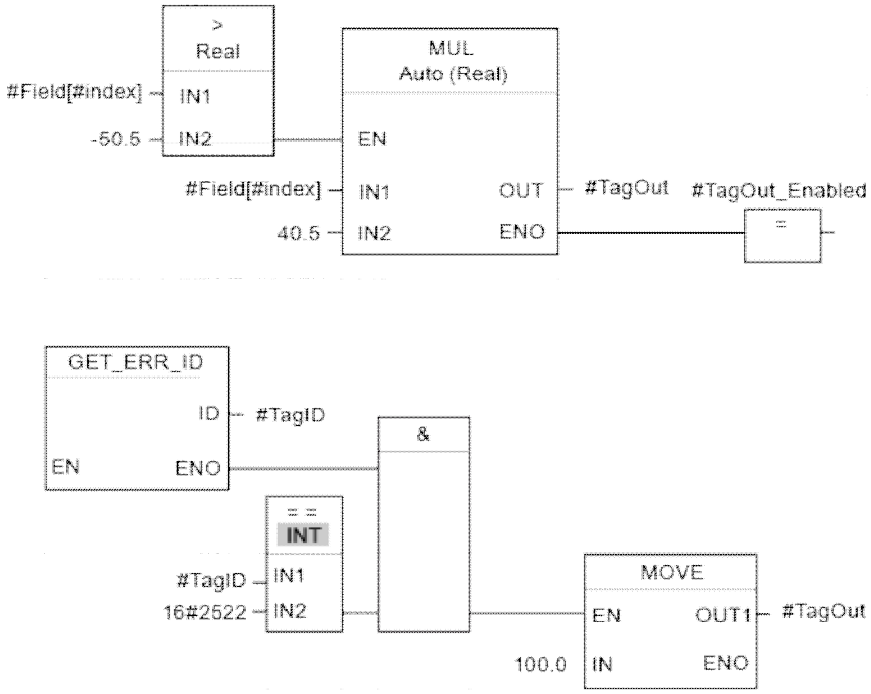
次の表に、ID パラメータで出力できる値をリストします。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外

2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、またはファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、またはファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー:入力
2943	10563	書き込みエラー:出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。		

例

次の例で、命令がどのように動作するかを示します。



#Field[#index]タグへのアクセスでエラーが発生しました。 「乗算」命令および#TagOut_Enabled オペランドの許可出力 ENO のシグナル状態は、読み取り/アクセスエラーに関係なく「1」になり、乗算は「0.0」の値を使用して実行されます。 このエラーシナリオが発生した場合は、エラーを取得するために、「乗算」命令の後に「ローカルでエラー ID を取得」命令をプログラミングすることを推奨します。 「ローカルでエラー ID を取得」命令によって供給されるエラー情報は、比較命令「等しい」を使用して評価されます。 #TagID オペランドが ID 2522 を返す場合、これは読み取り/アクセスエラーが関係しており、値「100.0」が#TagOut 出力に書き込まれます。

INIT_RD: すべての保持データの初期化

説明

「すべての保持データを初期化」命令を使用すると、すべてのデータブロック、ビットメモリ、および SIMATIC タイマとカウンタの保持データが同時にリセットされます。 この命令は、実行するとプログラムサイクル持続時間を超過するため、スタートアップ OB 内でのみ実行が可能です。

パラメータ

次の表に、「すべての保持データを初期化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数	入力「REQ」のシグナル状態が「1」の場合、すべての保持データがリセットされます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

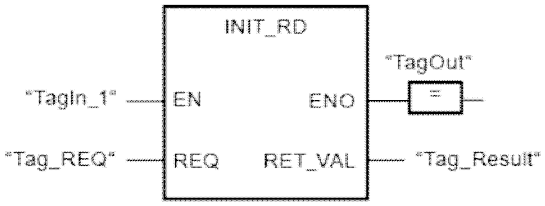
RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B5	この命令はスタートアップ OB 内にプログラムされていないため、実行できません。
一般エラー情報	関連項目:"GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の変更についての情報は、「関連項目」に記載されています。	

例

次の例で、命令がどのように動作するかを示します。



オペランド「TagIn_1」および「Tag_REQ」のシグナル状態が「1」の場合、命令が実行されます。すべてのデータブロック、ビットメモリ、および SIMATIC タイマの保持データとカウンタがリセットされます。 この命令が工ラーなしで実行されると、ENO 許可出力のシグナル状態は「1」となります。

WAIT: 遅延時間の設定

説明

「遅延時間の設定」命令は、指定された時間の間プログラム実行を一時停止します。 この命令の WT パラメータに、マイクロ秒で時間を指定します。

遅延時間は、-32768 から最大+32767 マイクロ秒 (µs)が設定可能です。 設定可能な最短の遅延時間は個々の CPU に依存し、「遅延時間の設定」命令の実行時間に対応します。

この命令の実行には、さらに優先度の高いイベントが割り込む可能性があります。

「遅延時間の設定」命令はエラー情報を返しません。

パラメータ

次の表に、「遅延時間の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
WT	Input	INT	I、Q、M、D、L、P、または定数	遅延時間(µs)

RUNTIME: プログラムランタイムの測定

説明

「プログラムランタイムの測定」命令を使用して、プログラム全体、個々のブロック、またはコマンドシーケンスのランタイムを測定します。

プログラム全体のランタイムを測定する場合は、「プログラムランタイムの測定」命令を OB1 で呼び出します。 最初の呼び出しでランタイムの測定が開始され、2 回目の呼び出し後に出力 RET_VAL がプログラムのランタイムを返します。 測定されたランタイムには、たとえば上位レベルのイベントや通信による割り込みなど、プログラム実行中に発生する可能性のあるすべての CPU プロセスが含まれます。 「プログラムランタイムの測定」命令は CPU の内部カウンタを読み取り、値を IN-OUT パラメータ MEM に書き込みます。 この命令は、内部カウンタ周波数に基づいて現在のプログラムランタイムを計算し、出力 RET_VAL に書き込みます。

個々のブロック、または個々のコマンドシーケンスのランタイムを測定する場合、3 つの個別のネットワークが必要です。 プログラム内の個々のネットワークで、「プログラムランタイムの測定」命令を呼び出します。 命令のこの最初の呼び出しでランタイム測定の開始位置を設定します。 その後、次のネットワーク内で目的のプログラムブロック、またはコマンドシーケンスを呼び出します。 他のネットワーク内で、「プログラムランタイムの測定」命令の 2 回目の呼び出しを行い、命令の最初の呼び出しで行ったように同じメモリを IN-OUT パラメータ MEM に割り当てます。 3 番目のネットワークの「プログラムランタイムの測定」命令は内部 CPU カウンタを読み取り、内部カウンタ周波数に基づいてプログラムブロック、またはコマンドシーケンスの現在のランタイムを計算し、それを出力 RET_VAL に書き込みます。

以下は、ファームウェアバージョン V4.1 未満の S7-1200 が対象です。 「プログラムランタイムの測定」命令は、内部の高周波数カウンタを使用して時間を計算します。 カウンタがオーバーフローする場合(これは 1 分当たり 1 回発生する可能性があります)、命令は値<= 0.0 を返します。 これらのランタイム値は無視してください。

注記
コマンドシーケンスのランタイムは、正確に測定することはできません。これは、コマンドシーケンス内の命令のシーケンスが、プログラムを最適にコンパイルする過程で変更されるためです。

パラメータ

次の表に、「プログラムランタイムの測定」命令のパラメータをリストします。

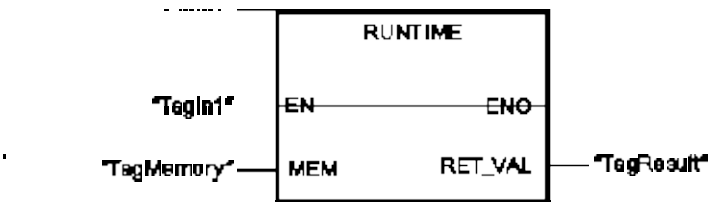
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
MEM	InOut	LREAL	I、Q、M、D、L	ランタイム測定の開始位置を保存します。
RET_VAL	Output	LREAL	I、Q、M、D、L	測定したランタイムを秒単位で返します

有効なデータタイプの追加情報については、「関連項目」を参照してください。

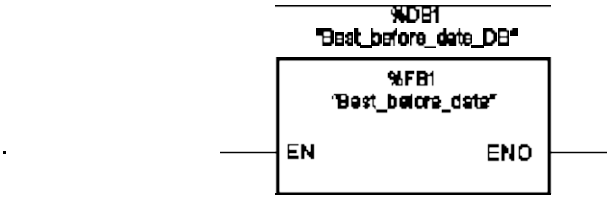
例

次の例は、命令がプログラムブロックのランタイム計算に基づいてどのように動作するかを示します。

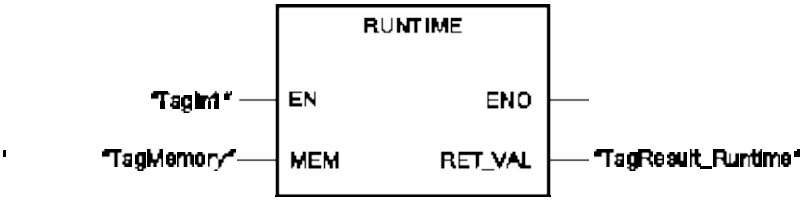
ネットワーク 1:



ネットワーク 2:



ネットワーク 3:



ネットワーク 1 の「TagIn1」オペランドのシグナル状態が「1」の場合、この命令が実行されます。この命令の最初の呼び出しでランタイムの測定が開始され、TagMemory オペランドでの命令の 2 回目の呼び出しの参照先としてバッファリングされます。

ネットワーク 2 で、「Best_before_date」プログラムブロック FB1 が呼び出されます。

FB1 プログラムブロックの実行が終了し、「TagIn1」オペランドのシグナル状態が「1」の場合、ネットワーク 3 の命令が実行されます。 命令の 2 回目の呼び出しにより、プログラムブロックのランタイムが計算され、結果が出力 RET_VAL に書き込まれます。

ワード論理演算

この章には下記に関する情報が記載されています：

AND: AND 演算

説明

「AND 演算」命令を使用し、入力 IN1 の値を入力 IN2 の値に AND ロジックでビットごとにリンクし、結果を出力 OUT で照会します。

この命令が実行されると、入力 IN1 の値のビット 0 が AND ロジックで入力 IN2 の値のビット 0 にリンクされます。 結果は出力 OUT のビット 0 に格納されます。 同じ論理演算が、指定された値の他のすべてのビットに実行されます。

初期状態では、この命令ボックスには 2 つ以上の入力(IN1 および IN2)が含まれています。 入力数は拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。 この命令の実行中に、すべての使用可能な入力パラメータの値が AND ロジックでリンクされます。 結果は出力「OUT」に格納されます。

論理演算の両方のビットのシグナル状態が「1」になる場合のみ、結果ビットのシグナル状態は「1」です。 論理演算の 2 つのビットの 1 つのシグナル状態が「0」の場合、対応する結果ビットはリセットされます。

パラメータ

次の表に、「AND 演算」命令のパラメータを示します。

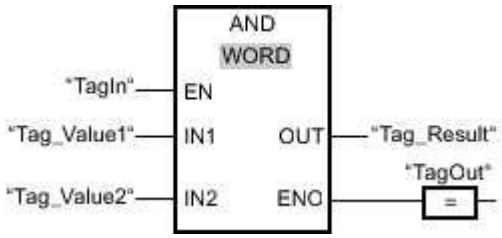
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	論理演算の最初の値
IN2	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	論理演算の 2 番目の値
INn	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	オプションの入力値
OUT	Output	ビット列	I、Q、M、D、L、P	I、Q、M、D、L、P	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0000 0000 0000 0101

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。オペランド「Tag_Value1」の値が、AND によってオペランド「Tag_Value2」の値にリンクされます。 結果はビット単位でマッピングされ、オペランド「Tag_Result」に送信されます。 許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

OR: OR 論理和演算

説明

「OR 演算」命令を使用し、入力 IN1 の値を入力 IN2 の値に OR ロジックでビットごとにリンクし、結果を出力 OUT で照会します。

この命令が実行されると、入力 IN1 の値のビット 0 が OR ロジックで入力 IN2 の値のビット 0 にリンクされます。 結果は出力 OUT のビット 0 に格納されます。 同じ論理演算が、指定されたタグのすべてのビットについて実行されます。

初期状態では、この命令ボックスには 2 つ以上の入力(IN1 および IN2)が含まれています。 入力数は命令ボックスで拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。 この命令の実行中に、すべての使用可能な入力パラメータの値が OR ロジックでリンクされます。 結果は出力「OUT」に格納されます。

論理演算の 2 つのビットのうち少なくとも 1 つのシグナル状態が「1」ならば、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が「0」の場合、対応する結果ビットがリセットされます。

パラメータ

次の表に、「OR 演算」命令のパラメータを示します。

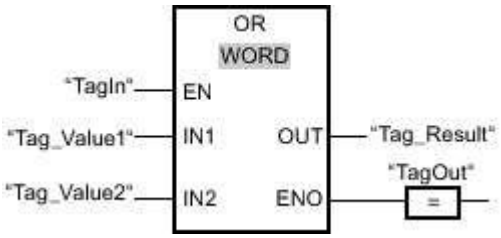
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	論理演算の最初の値
IN2	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	論理演算の 2 番目の値
INn	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	オプションの入力値
OUT	Output	ビット列	I、Q、M、D、L、P	I、Q、M、D、L、P	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1111

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。オペランド「Tag_Value1」の値が、OR によってオペランド「Tag_Value2」の値にリンクされます。結果はビット単位でマッピングされ、オペランド「Tag_Result」に送信されます。許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

XOR: EXCLUSIVE OR 排他的論理和演算

説明

「排他的論理和演算」命令を使用して、入力 IN1 の値を入力 IN2 の値に排他的 OR ロジックでビットごとにリンクし、結果を出力 OUT で照会することができます。

この命令が実行されると、入力 IN1 の値のビット 0 が排他的 OR ロジックで入力 IN2 の値のビット 0 にリンクされます。 結果は出力 OUT のビット 0 に格納されます。 同じ論理演算が、指定された値のすべての他のビットについて実行されます。

初期状態では、この命令ボックスには 2 つ以上の入力(IN1 および IN2)が含まれています。 入力数は命令ボックスで拡張できます。 挿入された入力は、ボックス内で昇順に番号付けされます。 この命令の実行中に、すべての使用可能な入力パラメータの値が排他的 OR ロジックでリンクされます。結果は出力「OUT」に格納されます。

論理演算の 2 つのビットのうち 1 つのシグナル状態が「1」の場合、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が「1」または「0」の場合、対応する結果ビットがリセットされます。

パラメータ

次の表に、「排他的論理和演算」命令のパラメータを示します。

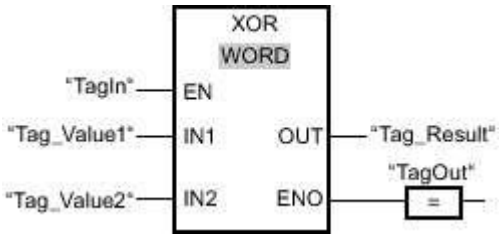
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN1	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	論理演算の最初の値
IN2	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	論理演算の 2 番目の値
INn	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	オプションの入力値
OUT	Output	ビット列	I、Q、M、D、L、P	I、Q、M、D、L、P	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	0101 0101 0101 0101
IN2	Tag_Value2	0000 0000 0000 1111
OUT	Tag_Result	0101 0101 0101 1010

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Value1」の値が、排他的 OR によってオペランド「Tag_Value2」の値にリンクされます。結果はビット単位でマッピングされ、オペランド「Tag_Result」に送信されます。許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

INVERT: 1 の補数の作成

説明

「1 の補数の作成」命令を使用して、入力 IN のビットのシグナル状態を反転します。 この命令が処理されると、16 進数マスク(16 ビットの数では W#16#FFFF、または 32 ビットの数では DW#16#FFFF FFFF)によって入力 IN の値が排他的 OR にリンクされます。 これによって、個々のビットのシグナル状態を反転させ、出力 OUT に格納します。

パラメータ

次の表に、「1 の補数の作成」命令のパラメータを示します。

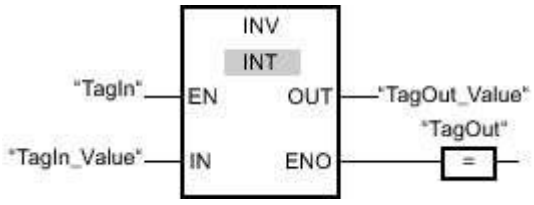
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	ビット列、整数	I、Q、M、D、L、P	I、Q、M、D、L、P	入力 IN の値の 1 の補数

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

パラメータ	オペランド	値	
IN	TagIn_Value	W#16#000F	W#16#7E
OUT	TagOut_Value	W#16#FFF0	W#16#81

オペランド「TagIn」のシグナル状態が「1」の場合、「1 の補数の作成」命令が実行されます。 この命令は、入力「TagIn_Value」の個々のビットのシグナル状態を反転させ、結果を出力「TagOut_Val-

ue」に書き込みます。 許可出力 ENO および出力「TagOut」のシグナル状態が「1」にセットされます。

DECO: デコード

説明

「デコード」命令を使用して、入力値で指定されたビットを出力値にセットします。

「デコード」命令は、IN 入力の値を読み出し、そのビットを出力値にセットします。そのビット位置は、読み出された値に対応します。 出力値の他のビットはゼロで埋められます。 入力 IN の値が 31 よりも大きい場合、モジュロ 32 命令が実行されます。

パラメータ

次の表に、「デコード」命令のパラメータを示します。

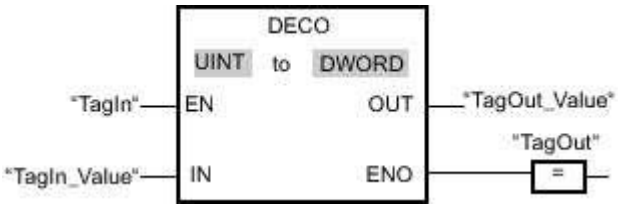
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	UINT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	設定される出力値のビット位置
OUT	Output	ビット列	I、Q、M、D、L、P	I、Q、M、D、L、P	出力値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

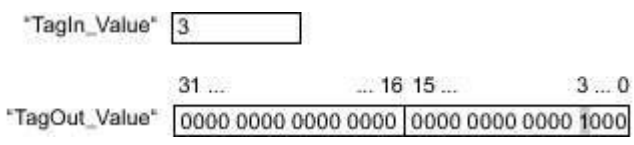
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令はオペランド「TagIn_Value」の値からビット番号「3」を読み取り、3 番目のビットをオペランド「TagOut_Value」の値にセットします。

命令の実行中にエラーが発生しなかった場合、出力 ENO のシグナル状態は「1」となり、出力「TagOut」が設定されます。

ENCO: エンコード

説明

「エンコード」命令を使用し、入力値の最下位ビットのビット番号を読み出し、それを出力 OUT に出力します。

「エンコード」命令は、IN 入力の値の最下位ビットを選択し、そのビット番号を出力 OUT のタグに書き込みます。

パラメータ

次の表に、「エンコード」命令のパラメータを示します。

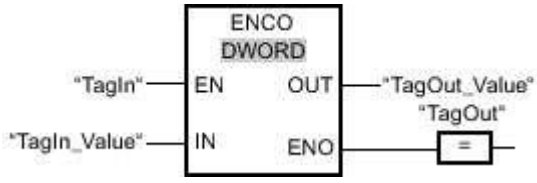
パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
EN	Input	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT	Output	INT	I、Q、M、D、L、P	I、Q、M、D、L、P	出力値

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

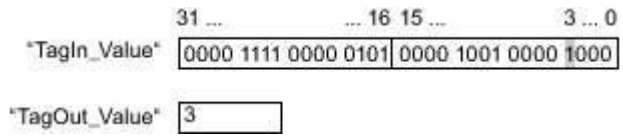
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、ビット位置「3」を入力「TagIn_Value」の最下位ビットとして選択し、値「3」を出力「TagOut_Value」のタグに書き込みます。

この命令がエラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

SEL: 選択

説明

スイッチ(入力 G)のシグナル状態に応じて、「選択」命令は入力 IN0 または IN1 のいずれかを選択し、その内容を出力 OUT に移動します。 入力 G のシグナル状態が「0」の場合、入力 IN0 の値が移動されます。 入力 G のシグナル状態が「1」の場合、入力 IN1 の値が出力 OUT に移動されます。

命令は、許可入力 EN のシグナル状態が「1」であり、かつすべてのパラメータのタグが同じデータタイプである場合にのみ実行できます。

パラメータ

次の表に、「選択」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
G	Input	BOOL	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、T、C、または定数	スイッチ
IN0	Input	ビット列、整数、浮動小数点数、タイマ、TOD、DATE、CHAR、WCHAR	ビット列、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、TDT、CHAR、WCHAR	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	最初の入力値
IN1	Input	ビット列、整数、浮動小数点数、タイマ、TOD、DATE、CHAR、WCHAR	ビット列、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、TDT、CHAR、WCHAR	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	2 番目の入力値
OUT	Output	ビット列、整数、浮動小数点数、タイマ、TOD、DATE、CHAR、WCHAR	ビット列、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、TDT、	I、Q、M、D、L、P	I、Q、M、D、L、P	結果

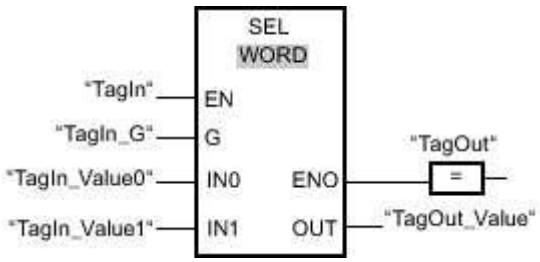
			CHAR、 WCHAR			

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
G	TagIn_G	0	1
IN0	TagIn_Value0	W#16#0000	W#16#4C
IN1	TagIn_Value1	W#16#FFFF	W#16#5E
OUT	TagOut_Value	W#16#0000	W#16#5E

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。入力「TagIn_G」のシグナル状態に応じて、入力「TagIn_Value0」または「TagIn_Value1」の値が選択され、出力「TagOut_Value」にコピーされます。この命令が エラーなしで実行された場合、許可出力 ENO のシグナル状態が「1」となり、「TagOut」出力がセットされます。

MUX: 多重化

説明

「多重化」命令を使用して、選択した入力の内容を出力 OUT にコピーします。 初期状態では、この命令ボックスには 2 つ以上の入力(IN0 および IN1)が含まれています。 命令ボックスの選択可能な入力の数は拡張できます。 最大 32 の入力を宣言できます。

入力はボックスで自動的に割り当てられます。 番号の割り当ては IN0 から開始され、新しい入力ごとに継続的にインクリメントされます。 K パラメータを使用し、内容が OUT 出力にコピーされる入力を定義します。 使用可能な入力数よりもパラメータ K の値が大きい場合、パラメータ ELSE の内容は出力 OUT にコピーされ、許可出力 ENO にシグナル状態「0」が割り当てられます。

「多重化」命令は、すべての入力と OUT 出力のタグが同じデータタイプの場合にのみ実行できます。K パラメータは、指定できるのが整数のみであるため、例外です。

次のいずれかの条件が満たされると、許可出力 ENO がリセットされます。

- 許可入力 EN のシグナル状態が「0」であること。
- K パラメータの入力は、使用可能な入力の外側にあります。 これは、ELSE 入力が使用されるかどうかに関係ない応答です。 OUT 出力の値は変更されません。
- 命令の実行中にエラーが発生していること。

パラメータ

次の表に、「多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
K	Input	整数	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	内容がコピーされる入力を指定します。 • K = 0 ならば、パラメータ IN0 • K = 1 ならば、パラメータ IN1 など
IN0	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	最初の入力値

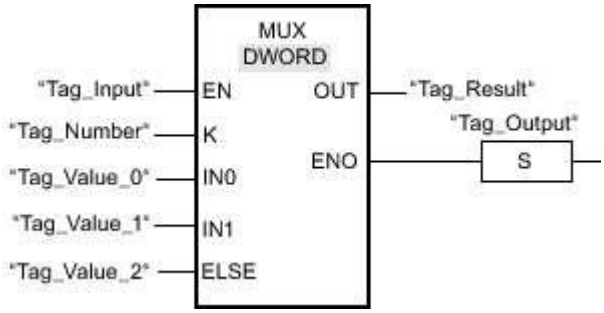
			LTOD、DATE、LDT			
IN1	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	2 番目の入力値
INn	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	オプションの入力値
ELSE	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	K > n の場合コピーする値を指定します。
OUT	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	I、Q、M、D、L、P	値がコピーされる出力

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
K	Tag_Number	1
IN0	Tag_Valuel_0	DW#16#00000000
IN1	Tag_Value_1	DW#16#003E4A7D
ELSE	Tag_Value_2	DW#16#FFFF0000
OUT	Tag_Result	DW#16#003E4A7D

「Tag_Input」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 オペランド「Tag_Number」の値に応じて、入力「Tag_Value_1」の値がコピーされ、出力「Tag_Result」のオペランドに割り当てられます。 この命令の実行中にエラーが発生しなかった場合、出力 ENO および「Tag_Output」が設定されます。

DEMUX: 逆多重化

説明

「逆多重化」命令を使用して、選択した出力に入力 IN の内容をコピーできます。 初期状態では、この命令ボックスには 2 つ以上の出力(OUT0 および OUT1)が含まれています。 選択可能な出力数は命令ボックスで拡張できます。 このボックスで、出力は自動的に番号が付与されます。 番号の割り当ては OUT0 から開始され、新しい出力ごとに継続します。K パラメータを使用して、IN 入力の内容がコピーされる出力を定義します。その他の出力は、変更されません。 K パラメータの値が使用可能な出力の数よりも大きい場合、ELSE パラメータの入力の内容 IN および許可出力 ENO にシグナル状態「0」が割り当てられます。

「逆多重化」命令は、入力 IN とすべての出力のタグが同じデータタイプの場合のみ実行できます。K パラメータは、指定できるのが整数のみであるため、例外です。

次のいずれかの条件が満たされると、許可出力 ENO がリセットされます。

- 許可入力 EN のシグナル状態が「0」であること。
- K パラメータの値が、使用可能な出力の数を超えていること。
- 命令の実行中にエラーが発生していること。

パラメータ

次の表に、「逆多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
K	Input	整数	整数	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値(IN)がコピーされる出力を指定します。 • K = 0 ならば、パラメータ OUT0 • K = 1 ならば、パラメータ OUT1 など
IN	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	I、Q、M、D、L、P、または定数	入力値
OUT0	Output	2 進数、整数、浮動小数点	2 進数、整数、浮動小数	I、Q、M、D、L、P	I、Q、M、D、L、P	最初の出力

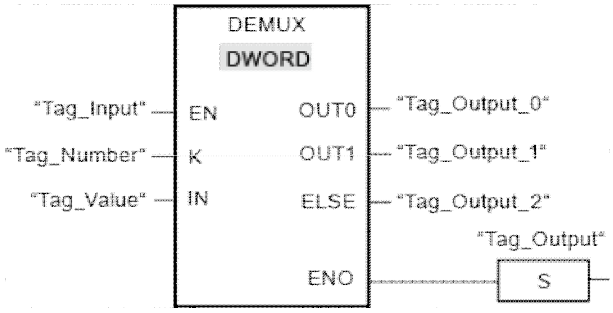
		数、タイマ、CHAR、WCHAR、TOD、DATE	点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT			
OUT1	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	I、Q、M、D、L、P	2 番目の出力
OUTn	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	I、Q、M、D、L、P	オプションの出力
ELSE	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、DATE	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	I、Q、M、D、L、P	K > n の場合に入力値(IN)がコピーされる出力。

命令ボックスの[<???>]ドロップダウンリストから、命令のデータタイプを選択できます。

使用可能なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

ネットワーク実行前の「逆多重化」命令の入力値

パラメータ	オペランド	値	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

ネットワーク実行後の「逆多重化」命令の出力値

パラメータ	オペランド	値	
OUT0	Tag_Output_0	不変	不変
OUT1	Tag_Output_1	DW#16#FFFFFFFF	不変
ELSE	Tag_Output_2	不変	DW#16#003E4A7D

"Tag_Input"入力のシグナル状態が「1」の場合、「逆多重化」命令が実行されます。 オペランド「Tag_Number」の値に応じて、入力「IN」の値が対応する出力にコピーされます。

シフトとローテーション

この章には下記に関する情報が記載されています：

- SHR: 右へシフト (S7-1200, S7-1500)
- SHL: 左へシフト (S7-1200, S7-1500)
- ROR: 右へローテーション (S7-1200, S7-1500)
- ROL: 左へローテーション (S7-1200, S7-1500)

SHR: 右へシフト

説明

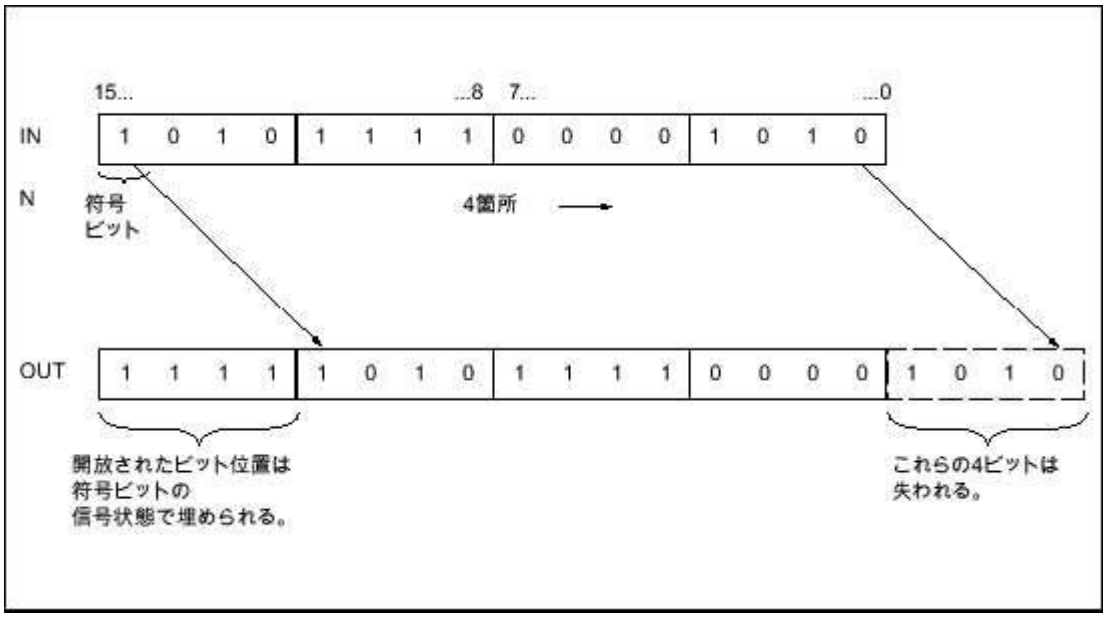
「右へシフト」命令を使用して、入力 IN のオペランドの内容を 1 ビットずつ右へシフトし、結果を OUT 出力で照会できます。 入力 N は、指定した値を移動するビット位置の数の指定に使用されます。

入力 N の値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりも入力 N の値が大きい場合、使用可能なビット位置の数だけ IN 入力のオペランド値が右にシフトされます。

符号なしの値がシフトされると、オペランドの左側の解放されたビット位置はゼロで埋められます。指定された値に符号が存在する場合、解放されたビット位置は符号ビットのシグナル状態で埋められます。

次の図に、整数データタイプのオペランドの内容が、どのように 4 ビット位置右へシフトされるかを示します。



パラメータ

次の表に、「右へシフト」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力

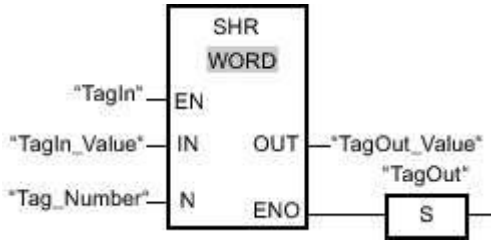
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	シフトされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値がシフトされるビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	I、Q、M、D、L	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	3
OUT	TagOut_Value	0000 0111 1111 0101

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。オペランド「TagIn_Value」の内容が、3 ビット位置右へシフトされます。 結果が「TagOut_Value」出力に出力されます。 この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」になり、「TagOut」出力がセットされます。

SHL: 左へシフト

説明

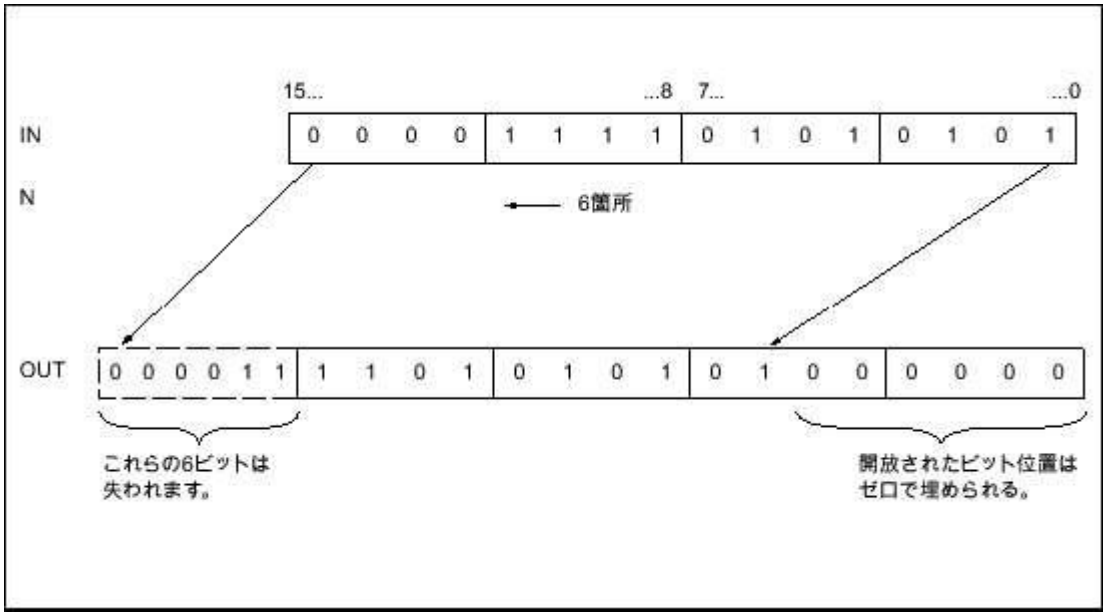
「左へシフト」命令を使用して、入力 IN のオペランドの内容を 1 ビットずつ左へシフトし、結果を OUT 出力で照会できます。 入力 N は、指定した値を移動するビット位置の数の指定に使用されます。

入力 N の値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりも入力 N の値が大きい場合、使用可能なビット位置の数だけ IN 入力のオペランド値が左へシフトされます。

シフトによって解放されたオペランドの右側のビット位置は、ゼロで埋められます。

次の図に、WORD データタイプのオペランドの内容が、どのように 6 ビット位置左へシフトされるかを示します。



パラメータ

次の表に、「左へシフト」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	シフトされる値

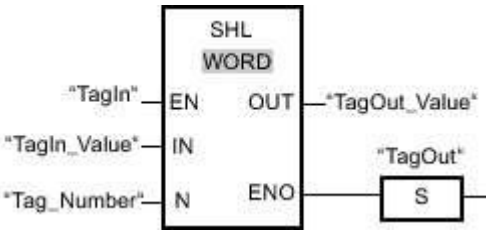
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値がシフトされるビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	I、Q、M、D、L	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0011 1111 1010 1111
N	Tag_Number	4
OUT	TagOut_Value	1111 1010 1111 0000

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。オペランド「TagIn_Value」の内容は、4 ビット位置左へシフトされます。 結果が「TagOut_Value」出力に出力されます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」となり、「TagOut」出力がセットされます。

ROR: 右へローテーション

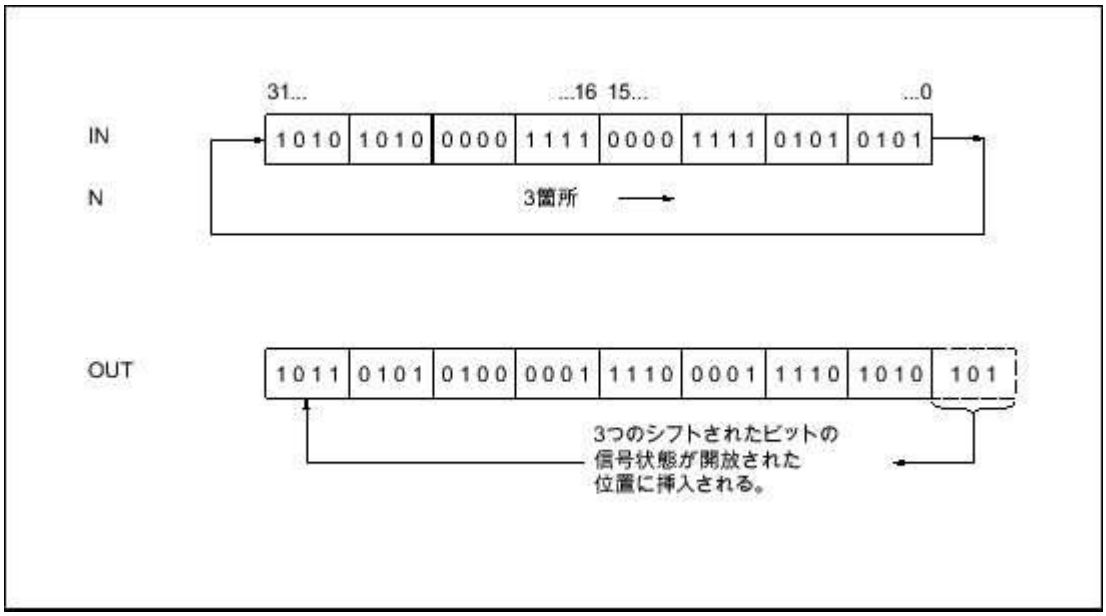
説明

「右へローテーション」命令を使用して、IN 入力のオペランドの内容を 1 ビットずつ右へローテーションし、結果を OUT 出力で照会できます。 入力 N を使って、指定した値をローテーションするビット位置の数を指定します。 ローテーションによって左側に解放されたビット位置は、左側に押し出されたビット位置の実位置で埋められます。

入力 N の値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりも N パラメータの値が大きい場合でも、IN 入力にあるオペランド値は、指定されたビット位置の数でローテーションされます。

次の図に、DWORD データタイプのオペランドの内容が、どのように 3 ビット位置右へローテーションされるかを示します。



パラメータ

次の表に、「右へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	ローテーションされる値

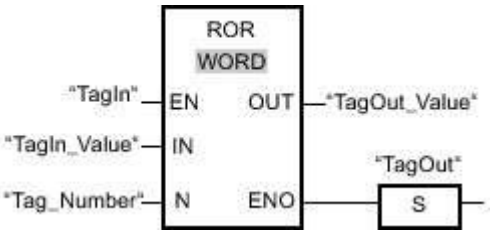
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値がローテーションされるビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	I、Q、M、D、L	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	0000 1111 1001 0101
N	Tag_Number	5
OUT	TagOut_Value	1010 1000 0111 1100

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。オペランド「TagIn_Value」の内容は、5 ビット位置右へローテーションされます。 結果が「TagOut_Value」出力に出力されます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」となり、「TagOut」出力がセットされます。

ROL: 左へローテーション

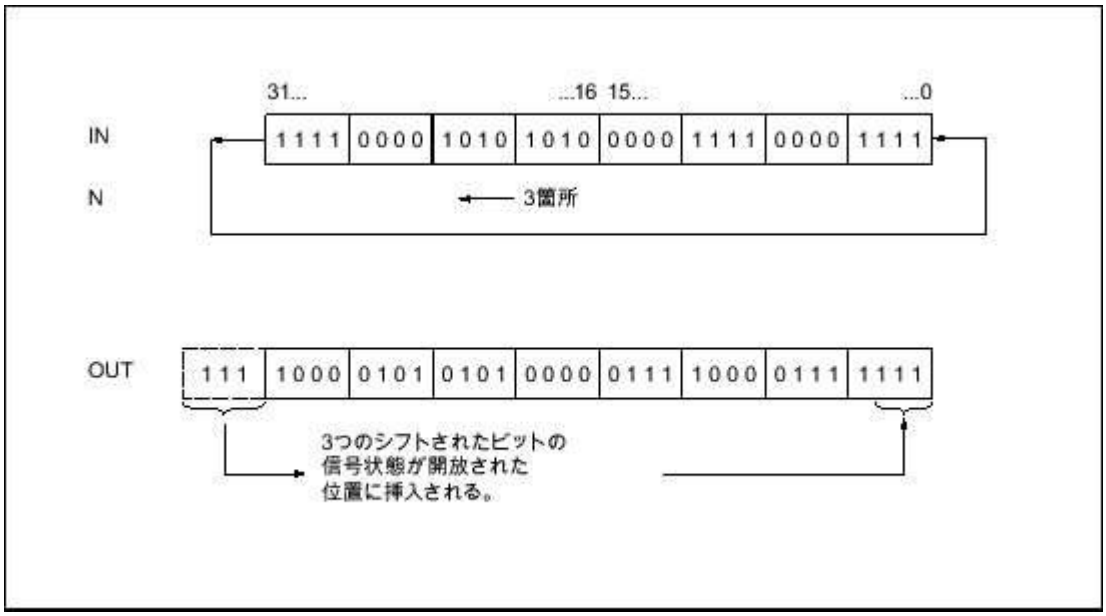
説明

「左へローテーション」命令を使用して、IN 入力のオペランドの内容を 1 ビットずつ左へローテーションし、結果を OUT 出力で照会できます。 入力 N を使って、指定した値をローテーションするビット位置の数を指定します。 ローテーションによって右側に解放されたビット位置は、左側に押し出されたビット位置の実位置で埋められます。

入力 N の値が「0」の場合、IN 入力の値が OUT 出力のオペランドにコピーされます。

使用可能なビット位置の数よりも N パラメータの値が大きい場合でも、IN 入力にあるオペランド値は、指定されたビット位置の数でローテーションされます。

次の図に、DWORD データタイプのオペランドの内容が、どのように 3 ビット位置左へローテーションされるかを示します。



パラメータ

次の表に、「左へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域		説明
		S7-1200	S7-1500	S7-1200	S7-1500	
EN	Input	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	BOOL	I、Q、M、D、L	I、Q、M、D、L	許可出力
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	ローテーションされる値

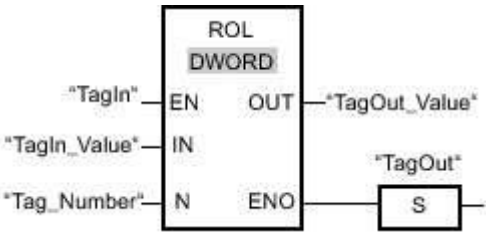
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	値がローテーションされるビット位置の数
OUT	Output	ビット列、整数	ビット列、整数	I、Q、M、D、L	I、Q、M、D、L	命令の結果

命令ボックスの[???]ドロップダウンリストから、命令のデータタイプを選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。



次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_Value	1010 1000 1111 0110
N	Tag_Number	5
OUT	TagOut_Value	0001 1110 1101 0101

「TagIn」のシグナル状態が「1」の場合、この命令が実行されます。オペランド「TagIn_Value」の内容は、左に 5 ビット位置ローテーションされます。 結果が「TagOut_Value」出力に出力されます。この命令がエラーなしで実行された場合、ENO 許可出力のシグナル状態が「1」となり、「TagOut」出力がセットされます。

レガシー

この章には下記に関する情報が記載されています：

DRUM:PLC 実行

説明

「PLC の実装」命令を使用して、対応するステップの OUT_VAL パラメータのプログラム済みの値をプログラム済み出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)に割り当てることができます。 そのため、この命令が特定のステップに留まっている間は、このステップは S_MASK パラメータでプログラムされた許可マスクの条件を満たす必要があります。このステップのイベントが真(TRUE)になっていて現在のステップのプログラムされた時間が経過した場合、または JOG パラメータの値が「0」から「1」に切り替わった場合、この命令は次のステップに進みます。 RESET パラメータのシグナル状態が「1」に切り替わると、この命令はリセットされます。 これによって、プリセットされたステップ(DSP)と現在のステップが等しくなります。

このステップで消費された時間の量は、プリセットタイムベース(DTBP)と各ステップのプリセット力ウンタ値(S_PRESET)の積によって計算されます。 新しいステップの開始時に、この計算値が DCC パラメータにロードされます。このパラメータには、現在のステップの残り時間が含まれています。たとえば、DTBP パラメータの値が「2」で、最初のステップのプリセット値が「100」(100 ミリ秒)の場合、DCC パラメータの値は「200」(200 ミリ秒)になります。

ステップは、タイマ値、イベント、またはその両方を使用してプログラム可能です。 イベントビットおよびタイマ値「0」を持つステップは、イベントビットのシグナル状態が「1」になると直ちに次のステップに進みます。 タイマ値のみによってプログラムされたステップは時間を直ちに開始します。「0」よりも大きいイベントビットおよびタイマ値を使用してプログラムされたステップは、イベントビットのシグナル状態が「1」になると時間を開始します。 イベントビットは、シグナル状態「1」で初期化されます。

プログラムされた最後のステップ(LST_STEP)の実行中にこのステップに割り当てられた時間が経過した場合、Q パラメータのシグナル状態は「1」にセットされ、それ以外の場合は「0」にセットされます。 Q パラメータがセットされると、この命令はリセットされるまでこのステップに留まります。

設定可能なマスク(S_MASK)で、出力ワード(OUT_WORD)の個々のビットを選択し、出力値(OUT_VAL)によって出力ビット(OUT1 ~ OUT16)をセットまたはリセットできます。 設定可能なマスクのビットのシグナル状態が「1」の場合、OUT_VAL 値は対応するビットをセットまたはリセットします。 設定可能なマスクビットのシグナル状態が「0」の場合、対応するビットは変更されません。16 のステップすべてに設定可能なマスクのすべてのビットは、シグナル状態「1」で初期化されます。

OUT1 パラメータの出力ビットは、出力ワード(OUT_WORD)の最下位ビットに対応します。 OUT16 パラメータの出力ビットは、出力ワード(OUT_WORD)の最上位ビットに対応します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成する場合は、データブロックはプロジェクトツリーの[プログラムブロック|システムブロック]パス内の[プログラムリソース」フォルダに保存されます。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「PLC 実装」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力

RESET	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、リセット条件を示します。
JOG	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「0」から「1」に切り替わると、命令が次のステップに進みます。
DRUM_EN	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「1」になると、PLC はイベントおよび時間条件に基づいて処理を進めます。
LST_STEP	Input	BYTE	I、Q、M、D、L、または定数	最後にプログラムされたステップの番号
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I、Q、M、D、L、または定数	イベントビット(i); 初期シグナル状態は「1」。
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I、Q、M、D、L	出力ビット(j)
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
OUT_WORD	Output	WORD	I、Q、M、D、L、P	PLC が出力値を書き込むワードアドレス。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
JOG_HIS	Static	BOOL	I、Q、M、D、L、または定数	JOG パラメータの履歴ビット
EOD	Static	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
DSP	Static	BYTE	I、Q、M、D、L、P、または定数	PLC のプリセットされたステップ
DSC	Static	BYTE	I、Q、M、D、L、P、または定数	PLC の現在のステップ
DCC	Static	DWORD	I、Q、M、D、L、P、または定数	PLC の現在の数値
DTBP	Static	WORD	I、Q、M、D、L、P、または定数	PLC のプリセットタイムベース
PrevTime	Static	TIME	I、Q、M、D、L、または定数	前のシステム時間
S_PRESET	Static	ARRAY[1..16] of WORD	I、Q、M、D、L、または定数	1 クロックパルス = 1 ミリ秒として、各ステップ[1 ~ 16]のカウントをプリセットします。
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの出力値[1 ~ 16, 0 ~ 15]。
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの設定可能なマスク[1 ~ 16, 0 ~ 15]。 初期シグナル状態は「1」。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

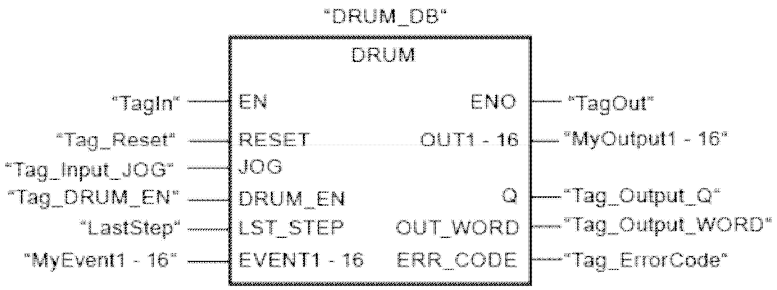
次の表に、ERR_CODE パラメータの値の意味を示します。

ERR_CODE*	説明
W#16#0000	エラーは発生していません。
W#16#000B	LST_STEP パラメータの値が 1 未満であるか、または 16 を超えています。
W#16#000C	DSC パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
W#16#000D	DSP パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例で命令はステップ 1 からステップ 2 に進みます。出力ビット(OUT1～OUT16)および出力ワード(OUT_WORD)はステップ 2 に構成されたマスクおよび OUT_VAL パラメータの値によってセットされます。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータの初期化には以下の値が使用されます。

パラメータ	オペランド	アドレス	値
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE

EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

以下の値が命令の「DRUM_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE

OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

出力パラメータは、命令の実行前に以下の値にセットされます。

パラメータ	オペランド	アドレス	値
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE

OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	アドレス	値
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

命令の実行後、以下の値が命令のインスタンスデータブロック「DRUM_DB」で変更されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002

DCC	DBD16	DW#16#000000C8

DCAT: ディスクリート制御タイマアラーム

説明

「ディスクリート制御タイマアラーム」命令は、CMD パラメータが開くまたは閉じるためのコマンドを発行した時点からの時間を累積するために使用されます。 時間は、指定した時間内でプリセット時間(PT)を超えるか、またはデバイスの開閉に関する情報を受信するまで蓄積します(O_FB または C_FB)。 デバイスの開閉に関する情報を受信する前にプリセット済み時間が経過すると、対応するアラームが有効になります。 プリセット済み時間の前にコマンド入力の信号状態が切り替わると、この時間経過が再開します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「ディスクリート制御タイマアラーム」命令は入力条件に対して以下のように応答します。

- CMD パラメータの信号状態が「0」から「1」に切り替わると、パラメータ Q、CMD_HIS、ET (ET < PT の場合のみ)、OA および CA は以下のように影響されます。
  - Q および CMD_HIS パラメータが「1」にセットされます。
  - ET、OA および CA パラメータが「0」にリセットされます。
- CMD パラメータの信号状態が「1」から「0」に切り替わると、Q、ET (ET < PT の場合のみ)、OA、CA、および CMD_HIS パラメータが「0」にリセットされます。
- CMD および CMD_HIS パラメータの信号状態が「1」で、パラメータ O_FB が「0」にセットされると、命令が最後に実行されてからの時間差(ms)がパラメータ ET の値に加算されます。 パラメータ ET の値がパラメータ PT の値を超えると、パラメータ OA の信号状態が「1」にセットされます。 ET パラメータの値が PT パラメータの値を超えない場合は、OA パラメータの信号状態は「0」にリセットされます。 CMD_HIS パラメータの値が、CMD パラメータの値にリセットされます。
- パラメータ CMD、CMD_HIS、および O_FB の信号状態が「1」にセットされ、パラメータ C_FB の値が「0」の場合、パラメータ OA の信号状態が「0」にセットされます。 パラメータ ET の値が、パラメータ PT の値にセットされます。O_FB パラメータの信号状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 パラメータ CMD_HIS の値が、パラメータ CMD の値にセットされます。
- CMD、CMD_HIS および C_FB パラメータが信号状態「0」を返す場合、命令が最後に実行されてからの時間差(ms)が ET パラメータの値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、パラメータの信号状態が CA「1」にリセットされます。 PT パラメータの値を超えなければ、CA パラメータの信号状態は「0」です。 パラメータ CMD_HIS の値が、パラメータ CMD の値にセットされます。
- CMD、CMD_HIS および O_FB パラメータの信号状態が「0」のときに C_FB パラメータが「1」にセットされると、CA パラメータが「0」にセットされます。 パラメータ ET の値が、パラメータ PT の値にセットされます。C_FB パラメータの信号状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 パラメータ CMD_HIS の値が、パラメータ CMD の値にセットされます。
- O_FB および C_FB パラメータの信号状態が同時に「1」の場合、両方のアラーム出力の信号状態が「1」にセットされます。

「ディスクリート制御タイマアラーム」命令は、エラー情報を返しません。

パラメータ

次の表に、「ディスクリート制御タイマアラーム」命令のパラメータを示します。

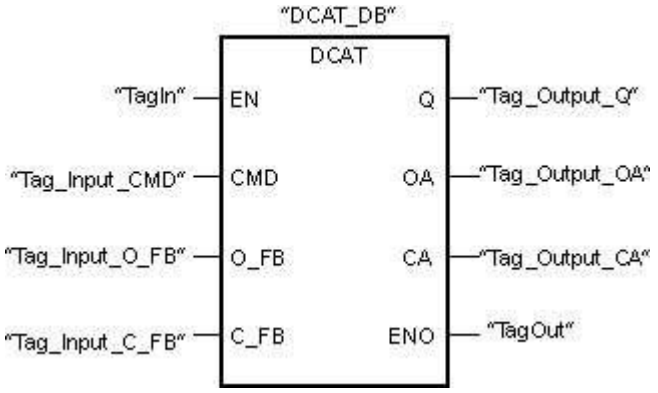
パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
CMD	Input	BOOL	I、Q、M、D、L、または定数	信号状態「0」は、「閉じる」コマンドを示します。 信号状態「1」は、「開く」コマンドを示します。
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
Q	Output	BOOL	I、Q、M、D、L	CMD パラメータのステータスを示します
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
ET	Static	DINT	D、L、または定数	現在の経過時間、1 カウント=1 ミリ秒。
PT	Static	DINT	D、L、または定数	プリセットされたタイマ値、1 カウント = 1 ミリ秒。
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
CMD_HIS	Static	BOOL	D、L、または定数	CMD の履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例では、パラメータ CMD が「0」から「1」に切り替わります。 命令の実行後、パラメータ Q が「1」にセットされ、2 つのアラーム出力 OA および CA の信号状態は「0」になります。 インスタンスデータブロックのパラメータ CMD_HIS が信号状態「1」にセットされ、パラメータ ET が「0」にリセットされます。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令のインスタンスデータブロック「DCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令のインスタンスデータブロック「DCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

MCAT: モータ制御タイマアラーム

説明

「モータ制御タイマアラーム」命令を使用して、コマンド入力(開または閉)の 1 つが有効になる時点から、時間を累積することができます。 時間はプリセット済み時間が経過するまで、または関連のフィードバック入力が、デバイスが指定された時間内に要求された操作を実行したことを示す場合に蓄積されます。 フィードバックが受信される前にプリセット済み時間が経過すると、対応するアラームがトリガされます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「モータ制御タイマアラーム」命令は、エラー情報を返しません。

「モータ制御タイマアラーム」命令の実行

次の表に、さまざまな入力条件に対する「モータ制御タイマアラーム」命令の応答を示します。

入力パラメータ								出力パラメータ								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	ステータス
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening (開き始める)
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open (開く)
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened (開いた)
>= PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm (アラームを開いている)

X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing (閉じ始める)
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close (閉じる)
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed (閉じた)
>= PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm (アラームを閉じている)
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped (停止)

凡例:	
INC	FB から ET への最後に処理されてからの時間差(ms)を追加します。
PT	PT は ET と同じ値にセットされます。
X	使用不可
<PT	ET < PT
>=PT	ET >= PT

入力パラメータ O_HIS および C_HIS のシグナル状態が両方とも「1」の場合、これらのパラメータのシグナル状態が直ちに「0」にセットされます。 この場合、上記の表の最後の行(X)が有効になります。 入力パラメータ O_HIS および C_HIS のシグナル状態が「1」であるかどうかをチェックすることができなくなるため、この場合の出力パラメータは以下のようにセットされます。

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

パラメータ

次の表に、「モータ制御タイマアラーム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	Output	BOOL	I、Q、M、D、L	許可出力
O_CMD	Input	BOOL	I、Q、M、D、L、または定数	[開]コマンド入力
C_CMD	Input	BOOL	I、Q、M、D、L、または定数	[閉]コマンド入力

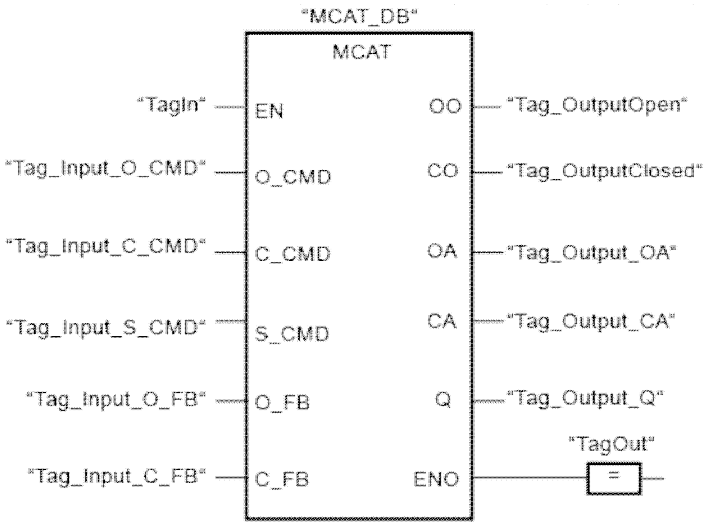
S_CMD	Input	BOOL	I、Q、M、D、L、または定数	[停止]コマンド入力
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
OO	Output	BOOL	I、Q、M、D、L	[開]出力
CO	Output	BOOL	I、Q、M、D、L	[閉]出力
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「0」は、エラー条件を示します。
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒
PT	Static	DINT	D、L、または定数	プリセット時間値、1 クロックパルス = 1 ミリ秒
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
O_HIS	Static	BOOL	D、L、または定数	[開]履歴ビット
C_HIS	Static	BOOL	D、L、または定数	[閉]履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

以下の値が命令の「MCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

以下の値が命令の「MCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

IMC: 入力ビットをマスクビットと比較

説明

「入力ビットをマスクビットと比較」命令を使用して、最大 16 のプログラムされた入力ビット(IN_BIT0 ～IN_BIT15)の信号状態とマスクの対応するビットを比較します。 最大 16 のマスクされたステップをプログラム可能です。 IN_BIT0 パラメータの値が、マスク CMP_VAL[x,0]の値と比較されます。「x」はステップ番号を示しています。 CMP_STEP パラメータで、比較に使用するマスクのステップ番号を指定します。 プログラムされた値は、すべて同じ方法で比較されます。プログラムされていない入力ビットやマスクビットは、デフォルトの信号状態である FALSE になります。

比較で一致が検出されると、OUT パラメータの信号状態が「1」にセットされます。それ以外の場合、OUT パラメータが「0」にセットされます。

CMP_STEP パラメータの値が 15 よりも大きい場合、この命令は実行されません。 ERR_CODE パラメータでエラーメッセージが出力されます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「入力ビットをマスクビットと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。
IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。

IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
CMP_STEP	Input	BYTE	I、Q、M、D、L、P、または定数	比較に使用されるマスクのステップ番号。
OUT	Output	BOOL	I、Q、M、D、L	信号状態「1」は、一致が検出されたことを示します。 信号状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000A	CMP_STEP パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

SMC: スキャンマトリックスの比較

説明

「スキャンマトリックスの比較」命令を使用して、最大 16 のプログラム入力ビット(IN_BIT0 ~ IN_BIT15)と比較マスクの対応するビットの信号状態を各ステップで比較します。処理はステップ 1 から開始し、プログラムされた最後のステップ(LAST)まで、または一致が検出されるまで続行されます。 IN_BIT0 パラメータの入力ビットが、マスク CMP_VAL[x,0]の値と比較されます。「x」はステップ番号を示しています。 プログラムされた値は、すべて同じ方法で比較されます。 一致が検出されると、OUT パラメータの信号状態が「1」にセットされ、一致するマスクのステップ番号が OUT_STEP パラメータに書き込まれます。 プログラムされていない入力ビットやマスクビットは、デフォルトの信号状態である FALSE になります。 複数のステップに一致するマスクが存在する場合、最初に検出されたマスクのみが OUT_STEP パラメータに示されます。 一致が検出されないと、OUT パラメータの信号状態が「0」にセットされます。 この場合、OUT_STEP パラメータの値は LAST パラメータの値よりも「1」だけ大きくなります。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「スキャンマトリックスの比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。
IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。

IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
OUT	Output	BOOL	I、Q、M、D、L	信号状態「1」は、一致が検出されたことを示します。 信号状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT_STEP	Output	BYTE	I、Q、M、D、L、P	一致するマスクを持つステップの番号が含まれているか、または一致が検出されない場合は、LAST パラメータの値よりも「1」だけ大きいステップ番号が含まれています。
LAST	Static	BYTE	I、Q、M、D、L、P、または定数	一致するマスクをスキャンする最後のステップのステップ番号を指定します。
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000E	LAST パラメータの値が 15 よりも大きくなっています。

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。

LEAD_LAG: リード/ラグアルゴリズム

説明

「リード/ラグアルゴリズム」命令を使用して、アナログタグの信号を処理します。GAIN パラメータのゲイン値はゼロよりも大きくなければなりません。「リード/ラグアルゴリズム」命令の結果は、以下の等式を使用して計算されます。

$$OUT = \left[\frac{LG\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_OUT + GAIN \left[\frac{LD\_TIME + SAMPLE\_T}{LG\_TIME + SAMPLE\_T}\right] IN - GAIN \left[\frac{LD\_TIME}{LG\_TIME + SAMPLE\_T}\right] * PREV\_IN$$

「リード/ラグアルゴリズム」命令は、処理を固定プログラムサイクルで行う場合のみ正しい結果が得られます。 LD_TIME、LG_TIME、および SAMPLE_T パラメータには同じ単位を指定する必要があります。LG_TIME > 4 + SAMPLE_T の場合、この命令は以下の関数を実行します。

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

GAIN パラメータの値がゼロ以下の場合、計算は実行されず、ERR_CODE パラメータでエラー情報が出力されます。

「リード/ラグアルゴリズム」命令をダイナミックフィードフォワード制御でループと共に補償器として使用できます。 この命令は、2 つの操作から成ります。「リード」操作は OUT 出力のフェーズをシフトして、出力が入力の前に来るようにします。 一方「ラグ」操作は出力をシフトして、出力が入力の後になるようにします。「ラグ」演算は積分に相当するため、ノイズサプレッサやローパスフィルタとして使用できます。「リード」演算は微分と同等なため、ハイパスフィルタとして使用できます。 これらの 2 つの命令(リードおよびラグ)を併用すると、低周波域では出力フェーズが入力フェーズよりも遅れ、高周波域では出力フェーズが入力フェーズに先行します。 つまり「リード/ラグアルゴリズム」命令は帯域フィルタとして使用できます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

パラメータ

次の表に、「リード/ラグアルゴリズム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	REAL	I、Q、M、D、L、P、または定数	処理対象の現在のサンプル時間(サイクルタイム)の入力値。
SAMPLE_T	Input	INT	I、Q、M、D、L、P、または定数	サンプル時間

OUT	Output	REAL	I、Q、M、D、L、P	命令の結果
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
LD_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のリードタイム。
LG_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のラグタイム。
GAIN	Static	REAL	I、Q、M、D、L、P、または定数	% / %で表されるゲイン(定常状態時の出力の切り替えと入力の切り替えとの比率)。
PREV_IN	Static	REAL	I、Q、M、D、L、P、または定数	前の入力
PREV_OUT	Static	REAL	I、Q、M、D、L、P、または定数	前の出力

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パラメータ ERR_CODE

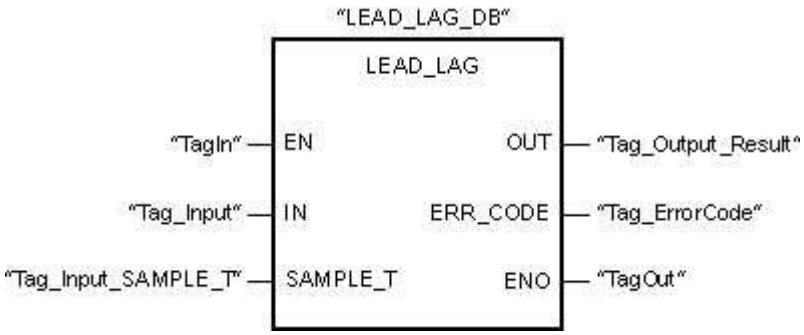
次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード* (W#16#...)	**説明**
0000	エラーは発生していません。
0009	GAIN パラメータの値がゼロ以下である。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。



次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

以下の値が命令のインスタンスデータブロック「LEAD_LAG_DB」に保存されます。

パラメータ	アドレス	値
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output_Result	2.0

以下の値が命令のインスタンスデータブロック「LEAD_LAD_DB」に保存されます。

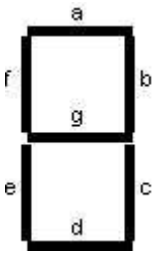
パラメータ	オペランド	値
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

SEG: 7 セグ表示のビットパターン作成

説明

「7 セグメント表示のビットパターンの作成」命令を使用して、指定されたソースワードの 4 つの各 16 進数(IN)を 7 セグメント表示の相当するビットパターンに変換します。 命令の結果は、OUT パラメータにダブルワードで出力されます。

16 進数と 7 セグメント(a、b、c、d、e、f、g)の割り当ての間には、以下の関係があります。

入力する数値 (バイナリ)	セグメントの割り当て -gfedcba	表示 (16 進数)	7 セグメント表示
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

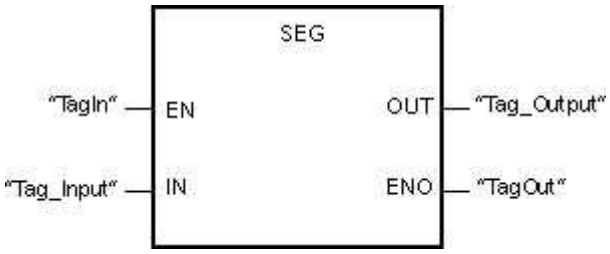
パラメータ

次の表に、「7 セグメント表示のビットパターンの作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	WORD	I、Q、M、D、L、P、または定数	4 つの 16 進数のソースワード
OUT	Output	DWORD	I、Q、M、D、L、P	7 セグメント表示のビットパターン

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値	
		16 進数	2 進数
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	00000110 01011011 01001111 01100110 表示: 1234

BCDCPL: 10 の補数の作成

説明

「10 の補数の作成」命令を使用して、IN パラメータで指定された 7 桁の BCD 数の 10 の補数を作成できます。 この命令は、次の数式を使用して計算します。

10000000 (BCD として)

– 7 桁の BCD 数

---

10 の補数(BCD として)

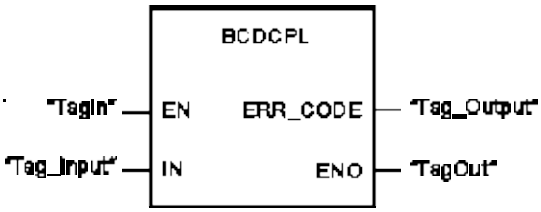
パラメータ

次の表に、「10 の補数の作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	ビット列	I、Q、M、D、L、P、または定数	7 桁の BCD 数
ERR_CODE	Output	DWORD	I、Q、M、D、L、P	命令の結果

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

BITSUM: セットビット数カウント

説明

「セットビット数カウント」命令を使用して、信号状態「1」にセットされるオペランドのビット数をカウントします。ビットをカウントするオペランドは、IN パラメータで指定されます。 命令の結果は、RET_VAL パラメータに出力されます。

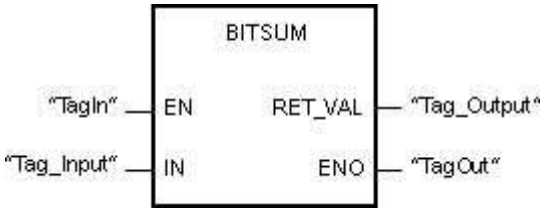
パラメータ

次の表に、「セットビット数カウント」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	Input	BOOL	I、Q、M、D、L、T、C	イネーブル入力
ENO	Output	BOOL	I、Q、M、D、L	イネーブル出力
IN	Input	DWORD	I、Q、M、D、L、P、または定数	セットビット数をカウントするオペランド。
RET_VAL	Output	INT	I、Q、M、D、L、P	セットするビットの数

例

次の例で、命令がどのように機能するかを示します。



次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 ビット)
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

STL

この章には下記に関する情報が記載されています：

- ビット論理演算 (S7-1500)
- タイマの動作 (S7-1500)
- カウンタ演算 (S7-1500)
- 比較演算 (S7-1500)
- 四則演算 (S7-1500)
- 移動操作 (S7-1500)
- 変換操作 (S7-1500)
- プログラム制御演算 (S7-1500)
- ワード論理演算 (S7-1500)
- レガシー (S7-1500)
- STL ニーモニック (S7-1500)

ビット論理演算

この章には下記に関する情報が記載されています：

- R_TRIG: 信号立ち上がりエッジの検出 (S7-1500)
- F_TRIG: 信号立ち下がりエッジの検出 (S7-1500)

R_TRIG: 信号立ち上がりエッジの検出

説明

「信号立ち上がりエッジの検出」命令を使用して、CLK 入力で「0」から「1」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「0」から「1」への状態変化を検出すると、Q 出力で信号立ち上がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

構文

「信号立ち上がりエッジの検出」命令には、以下の構文を使用します。

```
CALL R_TRIG, "R_TRIG_DB"
CLK := <operand>
Q := <operand>
```

パラメータ

次の表に、「信号立ち上がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L、または定数	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL R_TRIG, "R_TRIG_DB"	// 命令が呼び出されます。
CLK := "TagIn"	// 信号立ち上がりエッジが検出される。
Q := "TagOut"	// 信号立ち上がりエッジのシグナル状態が「1」。

CLK 入力のタグでの前の状態は、「R_TRIG_DB」タグに保存されます。 「0」から「1」へのシグナル状態の変化が「TagIn_1」および「TagIn_2」オペランド または「TagIn_3」オペランドで検出されると、「TagOut_Q」出力のシグナル状態は 1 サイクルについて「1」になります。

F_TRIG: 信号立ち下がりエッジの検出

説明

「信号立ち下がりエッジの検出」命令を使用して、CLK 入力で「1」から「0」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「1」から「0」への状態変化を検出すると、Q 出力で信号立ち下がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

構文

「信号立ち下がりエッジの検出」命令には、以下の構文を使用します。

```
CALL F_TRIG, <instance>
CLK  := <operand>
Q  := <operand>
```

パラメータ

次の表に、「信号立ち下がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L、または定数	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL F_TRIG, "F_TRIG_DB"	// 命令が呼び出されます。
CLK := "TagIn"	// 信号立ち下がりエッジが検出される。
Q := "TagOut"	// 信号立ち下がりエッジのシグナル状態が「1」。

CLK 入力のタグでの前の状態は、「F_TRIG_DB」タグに保存されます。 「TagIn」オペランドで「1」から「0」へのシグナル状態の変化が検出されると、TagOut 出力のシグナル状態は 1 サイクルについて「1」になります。

タイマの動作

この章には下記に関する情報が記載されています：

- TP: パルスの生成 (S7-1500)
- TON: オンディレータイマ (S7-1500)
- TOF: オフディレータイマ (S7-1500)
- TONR: タイムアキュムレータ (S7-1500)
- RESET_TIMER: タイマのリセット (S7-1500)
- PRESET_TIMER: 持続時間のロード (S7-1500)

TP: パルスの生成

説明

「パルス生成」命令を使用して、プログラムされた持続時間の出力 Q を設定できます。 この命令は、IN パラメータの論理演算結果(RLO)が「0」から「1」に切り替わるとき(信号立ち上がりエッジ)に開始されます。 命令が開始されると、プログラムされた時間 PT が開始します。 以降の入力信号の切り替えに関わらず、Q パラメータが時間 PT に対してセットされます。 新しい信号立ち上がりエッジが検出されても、時間 PT が継続している限り、Q パラメータのシグナル状態は影響を受けません。

ET 出力で現在時間値を照会できます。 時間値は T#0s で開始し、時間 PT の値に達すると終了します。 持続時間 PT が経過して入力 IN のシグナル状態が「0」の場合、ET 出力がリセットされます。

プログラムコードで「ブロックの呼び出し」(CALL)命令を使用して、「パルス生成」命令を呼び出すことができます。

注記
このタイマが、たとえば、プログラムでスキップされて、呼び出されない場合、ET 出力は時間の期限が切れるとすぐに定数値を返します。

「パルス生成」命令の各呼び出しは、命令データが保存される IEC タイマに割り当てる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TP_TIME」または「TP_LTIME」のローカルタグとしての宣言(#MyTP_TIMER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

オペレーティングシステムは、コールドリスタート中に「パルス生成」命令のインスタンスをリセットします。 ウォームリスタート後に命令のインスタンスを初期化するには、PT パラメータに値「0」をセットして、初期化するインスタンスをスタートアップ OB で呼び出します。 「パルス生成」命令が別のブロック内にある場合は、上位レベルブロックを初期化することによってこれらのインスタンスをリセットできます。

構文

「パルス生成」命令には、以下の構文を使用します。

```
CALL TP, "IEC タイマ"
???
IN := <operand>
PT := <operand>
```

```
Q := <operand>
ET := <operand>
```

パラメータ

次の表に、「パルス生成」命令のパラメータを示します。

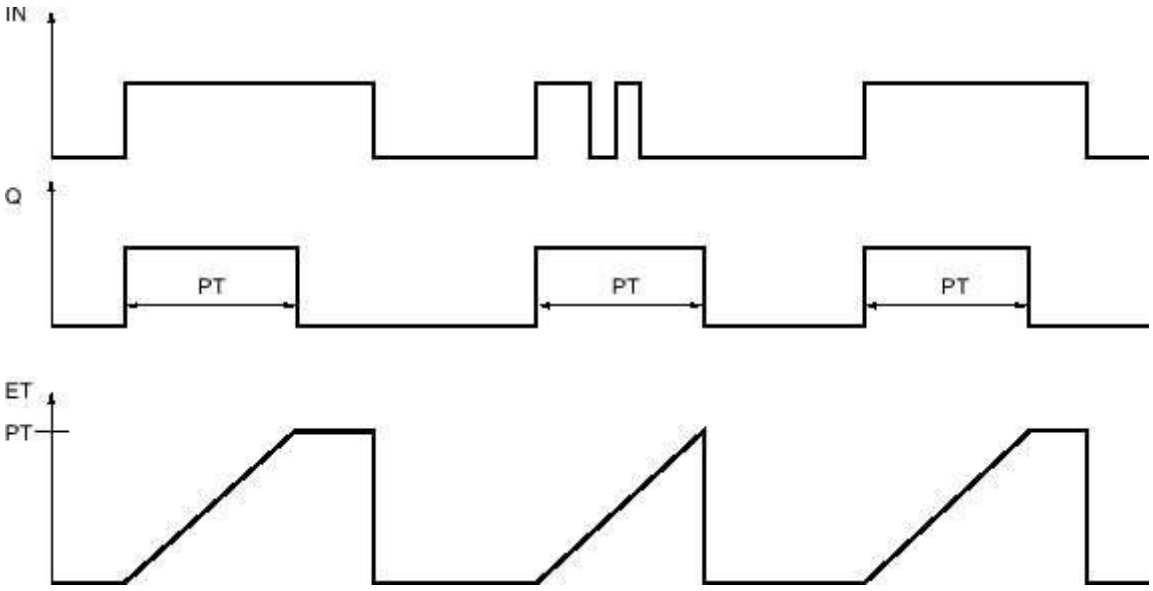
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	パルスの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	パルス出力
ET	Output	TIME, LTIME	I、Q、M、D、L、P	現在の時間値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、開始後の「パルス生成」命令の動作を示します。



例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL TP, "TP_DB"	// 命令が呼び出されます。「TP_DB」データブロックが命令に割り当てられる。
TIME	// 命令のデータタイプ
IN := "Tag_Start"	// 命令がオペランドの信号立ち上がりエッジで実行される。
PT := "Tag_PresetTIME"	// パルスの持続時間
Q := "Tag_Status"	// このオペランドはオペランド「Tag_PresetTIME」によって指定された持続時間にセットされる。
ET := "Tag_ElapsedTIME"	// 現在の時間値

TON: オンディレータイマ

説明

「オンディレータイマ」命令を使用して、プログラムされた時間 PT だけ Q 出力のセットを遅延させることができます。 この命令は、IN パラメータの論理演算結果(RLO)が「0」から「1」に切り替わるとき(信号立ち上がり工ッジ)に開始されます。 命令が開始されると、プログラムされた時間 PT が開始します。 時間 PT が経過すると、Q パラメータのシグナル状態は「1」になります。 Q パラメータは、開始入力 IN が「1」の間はセットされた状態が続きます。 開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータがリセットされます。 開始入力で信号の新しい立ち上がり工ッジが検出されると、タイマファンクションが再開されます。

ET 出力で現在時間値を照会できます。 時間値は T#0s で開始し、時間 PT の値に達すると終了します。 IN パラメータがシグナル状態「0」に切り替わると、ET パラメータは直ちにリセットされます。

注記
このタイマが、たとえば、プログラムでスキップされて、呼び出されない場合、ET 出力は時間の期限が切れるとすぐに定数値を返します。

プログラムコード内で「ブロック呼び出し」(CALL)命令を使用して「オンディレータイマ」命令を呼び出します。

「オンディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TON_TIME」または「TON_LTIME」のローカルタグとしての宣言(#MyTON_TIMER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

オペレーティングシステムは、コールドリスタート中に「オンディレータイマ」命令のインスタンスをリセットします。 ウォームリスタート後に命令のインスタンスを初期化するには、PT パラメータに値「0」をセットして、初期化するインスタンスをスタートアップ OB で呼び出します。 「オンディレータイマ」命令が別のブロック内にある場合は、上位レベルブロックを初期化することによってこれらのインスタンスをリセットできます。

構文

「オンディレータイマ」命令には、以下の構文を使用します。

```
CALL TON, "IEC タイマ"
???
```

```
IN := <operand>
PT := <operand>
Q := <operand>
ET := <operand>
```

パラメータ

次の表に、「オンディレータイマ」命令のパラメータを示します。

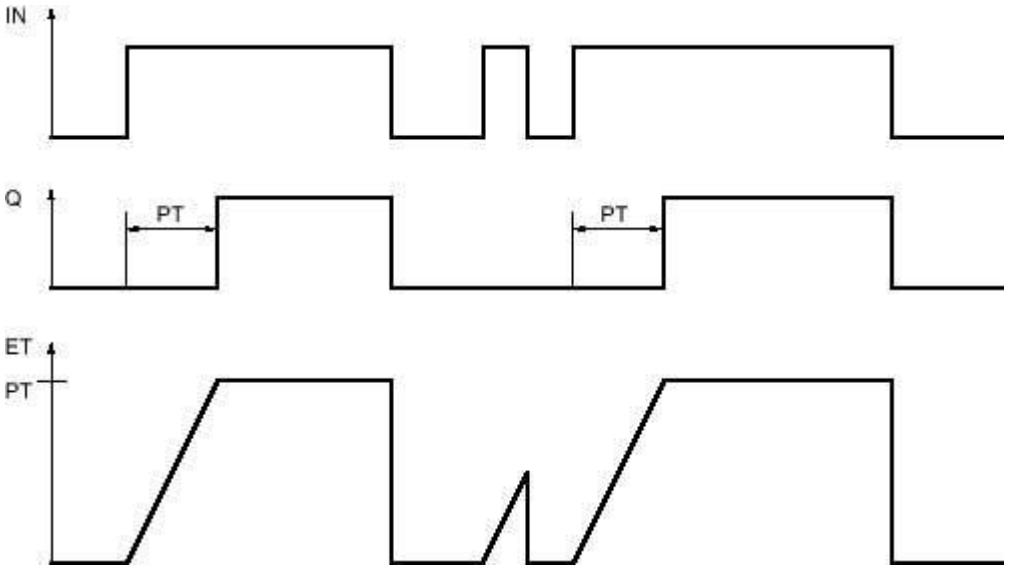
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	オンディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	時間 PT によって遅延されるシグナル状態
ET	Output	TIME, LTIME	I、Q、M、D、L、P	現在の時間値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、開始後の「オンディレータイマ」命令の動作を示します。



例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL "TON", "TON_DB"	// 命令が呼び出されます。「TON_DB」データブロックが命令に割り当てられる。
IN := "Tag_Start"	// 命令がオペランドの信号立ち上がりエッジで実行される。
PT := "Tag_PresetTIME"	// IN パラメータの信号立ち上がりエッジによって時間を指定する。 // オペランドは「Tag_PresetTIME」タグによって指定された持続時間 PT が期限切れになるとセットされる。
Q := "Tag_Status"	// Q パラメータは「Tag_Start」変数が「1」である限り、セットされた状態が継続する。 // 開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータのオペランドがリセットされる。
ET := "Tag_ElapsedTIME"	// 現在の時間値

TOF: オフディレータイマ

説明

「オフディレータイマ」命令を使用して、プログラムされた時間 PT だけ Q 出力のリセットを遅延させることができます。 Q パラメータは、IN パラメータの論理演算結果(RLO)が「0」から「1」(信号の立ち上がりエッジ)に切り替わるときにセットされます。 IN パラメータのシグナル状態が「0」に戻ると、プログラムされた時間 PT が開始します。 時間 PT が継続している限り、Q パラメータはセットされたままです。 時間 PT が経過すると、Q パラメータはリセットされます。 IN のシグナル状態が持続時間 PT が期限切れになる前に「1」に切り替わると、タイマはリセットされます。 Q パラメータのシグナル状態は「1」にセットされた状態が継続します。

現在の時間値は、ET パラメータ内で照会することができます。 時間値は T#0s で開始し、時間 PT の値に達すると終了します。 時間 PT が経過すると、ET パラメータは、IN パラメータが「1」に戻るまで、現在の値がセットされた状態が続きます。 持続時間 PT が経過する前に入力 IN が「1」に切り替わると、ET 出力が値 T#0s にリセットされます。

注記
このタイマが、たとえば、プログラムでスキップされて、呼び出されない場合、ET 出力は時間の期限が切れるとすぐに定数値を返します。

プログラムコード内で「ブロック呼び出し」(CALL)命令を使用して「オフディレータイマ」命令を呼び出します。

「オフディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TOF_TIME」または「TOF_LTIME」のローカルタグとしての宣言(#MyTOF_TIMER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

オペレーティングシステムは、コールドリスタート中に「オフディレータイマ」命令のインスタンスをリセットします。 ウォームリスタート後に命令のインスタンスを初期化するには、PT パラメータに値「0」をセットして、初期化するインスタンスをスタートアップ OB で呼び出します。 「オフディレータイマ」命令が別のブロック内にある場合は、上位レベルブロックを初期化することによってこれらのインスタンスをリセットできます。

構文

「オフディレータイマ」命令には、以下の構文を使用します。

```
CALL TOF, "IEC タイマ"
```

```
???
IN := <operand>
PT := <operand>
Q := <operand>
ET := <operand>
```

パラメータ

次の表に、「オフディレータイマ」命令のパラメータを示します。

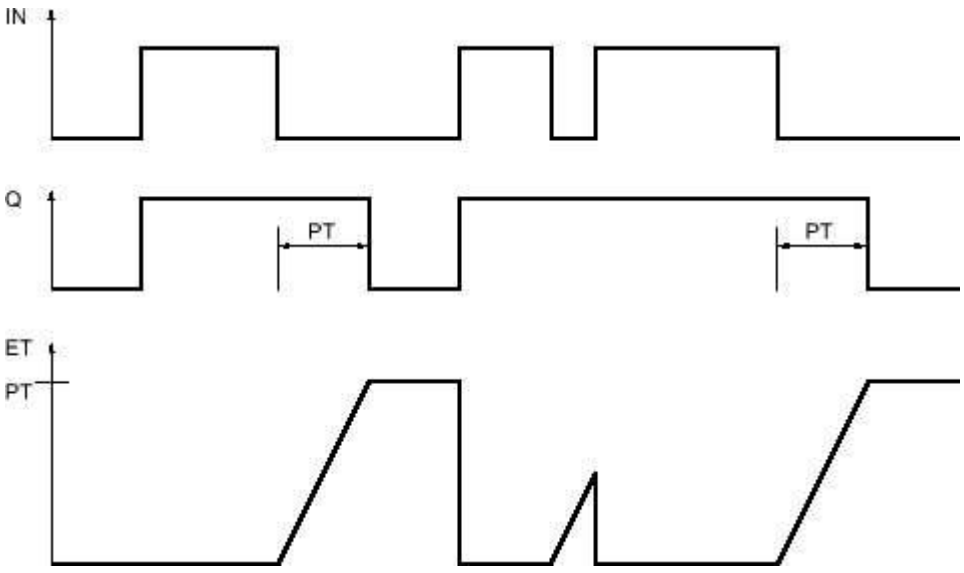
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	オフディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	時間 PT によって遅延されるシグナル状態
ET	Output	TIME, LTIME	I、Q、M、D、L、P	現在の時間値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、開始後の「オフディレータイマ」命令の動作を示します。



例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL TOF, "TOF_DB"	// 命令が呼び出されます。「TOF_DB」データブロックが命令に割り当てられる。
TIME	// 命令のデータタイプ
IN := "Tag_Start"	// 命令がオペランドの信号立ち上がりエッジで実行される。
PT := "Tag_PresetTIME"	// IN パラメータの信号立ち下がりエッジによって時間を指定する。
	// オペランドは、IN パラメータの信号立ち上がりエッジによって命令が開始されるとセットされる。
Q := "Tag_Status"	// IN パラメータの値が「1」から「0」に切り替わると、「Tag_Status」オペランドは、「Tag_PresetTIME」タグによって指定された時間が継続中である限り、セットされた状態のままである。
	// PT パラメータの時間が期限切れになると、オペランドはリセットされる。
ET := "Tag_ElapsedTIME"	// 現在の時間値

TONR: タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用して、PT パラメータによってセットされた時間内の時間値を累積します。 IN 入力のシグナル状態が「0」から「1」に切り替わると(信号立ち上がりエッジ、)命令が実行され、時間 PT が開始します。 時間 PT が経過している間、IN 入力のシグナル状態が「1」のときに記録される時間値が累積されます。 累積時間は ET 出力に書き込まれ、そこで照会できます。 持続時間 PT が経過すると、出力 Q のシグナル状態は「1」になります。 IN パラメータのシグナル状態が「1」から「0」(立ち下がりエッジ)に切り替わった場合でも、Q パラメータは「1」にセットされた状態が継続します。

開始入力のシグナル状態に関わりなく、R 入力は、ET および Q 出力をリセットします。

プログラムコード内で「ブロック呼び出し」(CALL)命令を使用して「タイムアキュムレータ」命令を呼び出します。

「タイムアキュムレータ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TONR_TIME」または「TONR_LTIME」のローカルタグとしての宣言(#MyTONR_TIMER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「タイムアキュムレータ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「タイムアキュムレータ」命令には、以下の構文を使用します。

```
CALL TONR, "IEC タイマ"
???
IN := <operand>
R := <operand>
PT := <operand>
Q := <operand>
ET := <operand>
```

パラメータ

次の表に、「タイムアキュムレータ」命令のパラメータを示します。

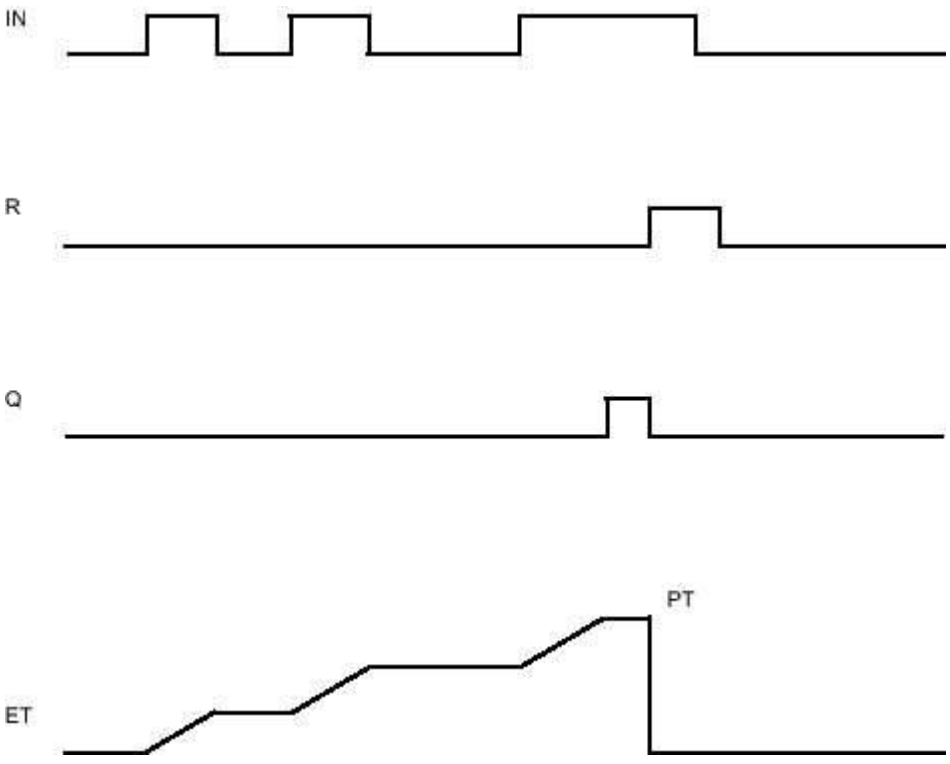
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	リセット入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	時間記録の最大持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	時間 PT の経過時にセットされる出力。
ET	Output	TIME, LTIME	I、Q、M、D、L、P	累積時間

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「タイムアキュムレータ」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL TONR, "TONR_DB"	// 命令が呼び出されます。「TONR_DB」データブロックが命令に割り当てられる。
TIME	// 命令のデータタイプ
IN := "Tag_Start"	// 命令がオペランドの信号立ち上がりエッジで実行され、PT 入力で持続時間が開始される。 時間 PT が経過している間、IN 入力のシグナル状態が「1」のときに記録される時間値が累積されます。
R := "Tag_Reset"	// 開始入力のシグナル状態に関係なく、R 入力が出力 ET と Q をリセットする。
PT := "Tag_PresetTIME"	// 時間値を累積する時間を指定する。 // PT の期限が切れると、オペランドがセットされる。
Q := "Tag_Status"	// IN パラメータのシグナル状態が「1」から「0」(立ち下がりエッジ)に切り替わった場合でさえも、Q パラメータは「1」にセットされた状態となる。
ET := "Tag_ElapsedTIME"	// 累積時間は、出力 ET に書き込まれ、そこで照会可能。

RESET_TIMER: タイマのリセット

説明

「タイマのリセット」命令を使用し、IEC タイマを「0」にリセットすることが可能です。 指定されたデータブロック内のタイマの構造体コンポーネントが「0」にリセットされます。

この命令は、RLO に影響を与えません。 TIMER パラメータで、「タイマのリセット」命令に、プログラムで宣言された IEC タイマが割り当てられます。

命令のデータは命令が呼び出されるときのみ更新され、割り当てられた IEC タイマがアクセスされるたびには更新されません。 データを照会する場合、命令の呼び出しから次の命令の呼び出しまでのみ同一になります。

構文

「タイマのリセット」命令には、以下の構文を使用します。

```
CALL RESET_TIMER
???
TIMER := <operand>
```

パラメータ

次の表に、「タイマのリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
TIMER	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	リセットされる IEC タイマ

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL TON, "TON_DB"	// 命令が呼び出されます。「TON_DB」インスタンスデータブロックが命令に割り当てられる。

```
time_type := TIME                   // 命令のデータタイプ
IN := "Tag_Start"                   // 命令がオペランド「Tag_Start」の信号立ち上がりエッジで実行される。
PT := "Tag_PresetTIME"              // インスタンスデータブロック「TON_DB」に保存された IEC タイマが、オペランド「Tag_PresetTIME」で指定された持続時間で起動する。
                                    オペランド「Tag_Status」は、オペランド「Tag_PresetTIME」によって指定された持続時間 PT が期限切れになるとセットされます。
Q := "Tag_Status"                   // オペランド「Tag_Start」のシグナル状態が「1」である限り、パラメータ Q はセットされた状態となる。
                                    // 開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータのオペランドがリセットされる。
ET := "Tag_ElapsedTIME"             // 現在の時間値

A "Tag_Input_1"                     // オペランド「Tag_Input_1」および
A "Tag_Input_2"                     // オペランド「Tag_Input_2」がシグナル状態が「1」を返す。

CALL RESET_TIMER                    // 「タイマのリセット」命令が呼び出される。
timer_type := IEC_TIMER             // IEC タイマのデータタイプ
TIMER := "TON_DB"                   // IEC タイマがリセットされる。
```

PRESET_TIMER: 持続時間のロード

説明

「持続時間のロード」命令を使用して、IEC タイマの時間を設定することが可能です。 この命令は各サイクルで実行されます。 この命令を実行すると、指定された IEC タイマの構造体に指定された時間を書き込みます。

プログラムで宣言した IEC タイマを「持続時間のロード」命令に割り当てます。

命令のデータは命令が呼び出されるとき、および割り当てられた IEC タイマがアクセスされるたびにのみ更新されます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

注記
命令の実行中に、指定された IEC タイマが動作している場合、指定された IEC タイマの現在の時刻が上書きされます。 これによって、IEC タイマのタイマステータスを変更できます。

構文

「時間の長さのロード」命令には、以下の構文を使用します。

```
CALL PRESET_TIMER
??? ???
PT := <operand>
TIMER := <operand>
```

パラメータ

次の表に、「持続時間のロード」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Time duration>	Input	TIME, LTIME	I、Q、M、D、L、または定数	IEC タイマが動作する時間
<IEC timer>	Output	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	長さがセットされる IEC タイマ

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL TON, "TON_DB"	// 命令が呼び出されます。「TON_DB」インスタンスデータブロックが命令に割り当てられる。
time_type := TIME	// 命令のデータタイプ
IN := "Tag_Start"	// 命令がオペランド「Tag_Start」の信号立ち上がりエッジで実行される。
PT := "Tag_PresetTIME"	// インスタンスデータブロック「TON_DB」に保存された IEC タイマが、オペランド「Tag_PresetTIME」で指定された持続時間で起動する。
	オペランド「Tag_Status」は、オペランド「Tag_PresetTIME」によって指定された持続時間 PT が期限切れになるとセットされます。
Q := "Tag_Status"	// オペランド「Tag_Start」のシグナル状態が「1」である限り、パラメータ Q はセットされた状態となる。
	// 開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータのオペランドがリセットされる。
ET := "Tag_ElapsedTIME"	// 現在の時間値
A "Tag_Input_1"	// オペランド「Tag_Input_1」および
A "Tag_Input_2"	// オペランド「Tag_Input_2」がシグナル状態が「1」を返す。
CALL PRESET_TIMER	// 「タイムアキュムレータ」命令が呼び出される。
time_type := TIME	// 命令のデータタイプ
timer_type := IEC_TIMER	// IEC タイマのデータタイプ
PT := "Tag_PresetTIME_new"	// この命令が持続時間「Tag_PresetTIME_new」をインスタンスデータブロック「TON_DB」に書き込む。これにより、インスタンスデータブロック内のオペランド「Tag_PresetTIME」の時間値が上書きされる。 タイマステータスのシグナル状態は、次の照会時、または "MyTimer".Q または "MyTimer".ET へのアクセス時に変化する可能性があります。
TIMER := "TON_DB"	// IEC タイマがリセットされる。

カウンタ演算

この章には下記に関する情報が記載されています：

- CTU: カウントアップ (S7-1500)
- CTD: カウントダウン (S7-1500)
- CTUD: カウントアップ/カウントダウン (S7-1500)

CTU: カウントアップ

説明

CV パラメータの値のインクリメントには、「カウントアップ」命令を使用できます。 CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わるとこの命令が実行され、CV パラメータのカウンタ現在値が 1 インクリメントされます。 立ち上がりエッジを検出するたびに、CV 出力において指定されたデータタイプの上限値に達するまで、カウンタ値がインクリメントされます。 上限値に達すると、CU パラメータのシグナル状態はこの命令には影響を与えなくなります。

Q パラメータのカウンタステータスを照会できます。 Q パラメータのシグナル状態は、PV パラメータによって決まります。 カウンタ現在値が PV パラメータの値以上の場合、Q パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、Q パラメータのシグナル状態は「0」です。

R パラメータのシグナル状態が「1」に切り替わると、CV パラメータの値はゼロにリセットされます。 R パラメータのシグナル状態が「1」である限り、CU パラメータのシグナル状態はこの命令には影響を与えません。

プログラムコード内で「ブロック呼び出し」(CALL)命令を使用して「カウントアップ」命令を呼び出します。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

システムデータタイプ IEC_<Counter> (共有 DB)のデータブロック

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

ローカルタグ

- CTU_SINT / CTU_USINT
- CTU_INT / CTU_UINT
- CTU_DINT / CTU_UDINT
- CTU_LINT / CTU_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)

- ブロックの「Static」セクションでの CTU_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyCTU_COUNTER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

オペレーティングシステムは、コールドリスタート中に「カウントアップ」命令のインスタンスをリセットします。 ウォームリスタート後に命令のインスタンスを初期化するには、この命令の R パラメータに値「1」をセットして、初期化するインスタンスをスタートアップ OB で呼び出します。「カウントアップ」命令が別のブロック内にある場合は、上位レベルブロックを初期化することによってこれらのインスタンスをリセットできます。

構文

「カウントアップ」命令には、以下の構文を使用します。

```
CALL CTU, "IEC カウンタ"
???
CU := <operand>
R := <operand>
PV := <operand>
Q := <operand>
CV := <operand>
```

パラメータ

次の表に、「カウントアップ」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CU	Input	BOOL	I、Q、M、D、L、または定数	カウント入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	リセット入力
PV	Input	整数	I、Q、M、D、L、P、または定数	Q 出力がセットされる値。
Q	Output	BOOL	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL CTU, "CTU_DB"	// 命令が呼び出されます。「CTU_DB」データブロックが命令に割り当てられる。
INT	// 命令のデータタイプ
CU := "Tag_StartCTU"	//「Tag_StartCTU」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CounterValue」オペランドのカウンタ現在値が 1 インクリメントされます。
	// カウンタ値は、INT の上限値 = 32767 に達するまで、インクリメントされる。
R := "Tag_ResetCounter"	// オペランド「Tag_ResetCounter」のシグナル状態が「1」に切り替わると、オペランド「Tag_CounterValue」は「0」にリセットされる。
PV := "Tag_PresetValue"	// このオペランドは Q パラメータのオペランドがセットされる値を定義する。
Q := "Tag_CounterStatus"	// オペランドは、カウンタ現在値が PV パラメータの値以上である限り、セットされる。
CV := "Tag_CounterValue"	// カウンタ現在値

CTD: カウントダウン

説明

「カウントダウン」命令は、CV パラメータのデクリメントに使用します。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わるとこの命令が実行され、CV パラメータのカウンタ現在値が 1 デクリメントされます。 指定されたデータタイプの下限値に達するまで、信号の立ち上がりエッジが検出されるたびにカウンタがデクリメントされます。 下限値に達すると、CD パラメータのシグナル状態はこの命令には影響を与えなくなります。

Q パラメータのカウンタステータスを照会できます。 カウンタ現在値がゼロ以下の場合、Q パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、Q パラメータのシグナル状態は「0」です。

CV パラメータの値は、LD パラメータのシグナル状態が「1」に切り替わると PV パラメータの値にセットされます。 LD パラメータのシグナル状態が「1」である限り、CD パラメータのシグナル状態はこの命令には影響を与えません。

プログラムコード内で「ブロック呼び出し」(CALL)命令を使用して「カウントダウン」命令を呼び出します。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

システムデータタイプ IEC_<Counter> (共有 DB)のデータブロック

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

ローカルタグ

- CTD_SINT / CTD_USINT
- CTD_INT / CTD_UINT
- CTD_DINT / CTD_UDINT
- CTD_LINT / CTD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)

- ブロックの「Static」セクションでの CTD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyCTD_COUNTER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

オペレーティングシステムは、コールドリスタート中に「カウントダウン」命令のインスタンスをリセットします。 この命令のインスタンスをウォームリスタート後に初期化するには、スタートアップ OB 内でこの命令の LD パラメータに値「1」がセットされた状態で、初期化するインスタンスを呼び出します。この場合、必要な CV パラメータの初期値は PV パラメータで指定します。 「カウントダウン」命令が別のブロック内にある場合は、上位レベルブロックを初期化することによってこれらのインスタンスをリセットできます。

構文

「カウントダウン」命令には、以下の構文を使用します。

```
CALL CTD, "IEC カウンタ"
???
CD := <operand>
LD := <operand>
PV := <operand>
Q := <operand>
CV := <operand>
```

パラメータ

次の表に、「カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CD	Input	BOOL	I、Q、M、D、L、または定数	カウント入力
LD	Input	BOOL	I、Q、M、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	CV 出力が LD = 1 にセットされる値。
Q	Output	BOOL	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL CTD, "CTD_DB"	// 命令が呼び出されます。「CTD_DB」データブロックが命令に割り当てられる。
INT	// 命令のデータタイプ
CD := "Tag_StartCTD"	//「Tag_StartCTD」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CounterValue」オペランドのカウンタ現在値が 1 デクリメントされます。
	// CV パラメータのカウンタ値は、INT の下限値 = -32768 に達するまで、デクリメントされる。
LD := "Tag_LoadPV"	// オペランド「Tag_LoadPV」のシグナル状態が「1」に切り替わると、オペランド「Tag_CounterValue」はオペランド「Tag_PresetValue」の値にセットされる。
PV := "Tag_PresetValue"	// LD パラメータでシグナル状態が「1」のときにセットされるカウンタ値を指定する。
Q := "Tag_CounterStatus"	// カウンタ現在値がゼロ以下になると、オペランドがセットされる。
CV := "Tag_CounterValue"	// カウンタ現在値

CTUD: カウントアップ/カウントダウン

説明

CV パラメータでのカウンタ値のインクリメント、またはデクリメントには、「カウントアップ/カウントダウン」命令を使用します。 CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、カウンタ現在値が 1 インクリメントされ、CV パラメータに保存されます。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、CV パラメータのカウンタ値が 1 デクリメントされます。 プログラムサイクル内の CU および CD 入力で信号の立ち上がりエッジが存在すると、CV パラメータのカウンタ現在値は変化しません。

カウンタ値は、CV パラメータで指定したデータタイプの上限値に達するまでインクリメントすることができます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。 カウンタ値は、指定されたデータタイプの下限値に達するとデクリメントされなくなります。

LD パラメータのシグナル状態が「1」に切り替わると、CV パラメータのカウンタ値が PV パラメータの値にセットされます。 LD パラメータのシグナル状態が「1」である限り、CU 入力および CD 入力のシグナル状態は命令に対する効力を持ちません。

R パラメータのシグナル状態が「1」に切り替わると、カウンタ値は 0 にセットされます。 R パラメータのシグナル状態が「1」である限り、CU、CD パラメータおよび LD パラメータのシグナル状態の変化は「カウントアップ/カウントダウン」命令に影響しません。

QU パラメータでアップカウンタのステータスを照会できます。 カウンタ現在値が PV パラメータの値以上の場合、QU パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、QU パラメータのシグナル状態は「0」です。 PV パラメータには定数も指定できます。

QD パラメータでダウンカウンタのステータスを照会できます。 カウンタ現在値がゼロ以下の場合、QD パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、QD パラメータのシグナル状態は「0」です。

プログラムコード内で「ブロック呼び出し」(CALL)命令を使用して「カウントアップ/カウントダウン」命令を呼び出します。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ/カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

システムデータタイプ IEC_<Counter> (共有 DB)のデータブロック

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

ローカルタグ

- CTUD_SINT / CTUD_USINT
- CTUD_INT / CTUD_UINT

- CTUD_DINT / CTUD_UDINT
- CTUD_LINT / CTUD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTUD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyCTUD_COUNTER など)

[???]ドロップダウンリストからデータタイプを選択した後、[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

オペレーティングシステムは、コールドリスタート中に「カウントアップ/カウントダウン」命令のインスタンスをリセットします。 ウォームリスタート後に命令のインスタンスを初期化する場合、スタートアップ OB の次のパラメータ値で初期化するインスタンスを呼び出す必要があります。

- R パラメータの値は、アップカウンタとして使用する場合、「1」にセットする必要があります。
- LD パラメータの値は、ダウンカウンタとして使用する場合、「1」にセットする必要があります。この場合、CV パラメータの必要な初期値を PV パラメータに指定します。

「カウントアップ/カウントダウン」命令のインスタンスが他のブロックに位置している場合、たとえば上位レベルのブロックを初期化するなどしてこれらのインスタンスをリセットすることができます。

構文

「カウントアップ/カウントダウン」命令には、以下の構文を使用します。

```
CALL CTUD, "IEC カウンタ"
???
CU := <operand>
CD := <operand>
R := <operand>
LD := <operand>
PV := <operand>
```

```
QU := <operand>
QD := <operand>
CV := <operand>
```

パラメータ

次の表に、「カウントアップ/カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CU	Input	BOOL	I、Q、M、D、L、または定数	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L、または定数	カウントダウン入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	リセット入力
LD	Input	BOOL	I、Q、M、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	QU 出力がセットされる値。 /CV 出力が LD = 1 にセットされる値。
QU	Output	BOOL	I、Q、M、D、L	アップカウンタのステータス
QD	Output	BOOL	I、Q、M、D、L	カウントダウンのステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL CTUD, "CTUD_DB"	// 命令が呼び出されます。「CTUD_DB」データブロックが命令に割り当てられる。
INT	// 命令のデータタイプ
CU := "Tag_StartCTU"	//「Tag_StartCTU」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CounterValue」オペランドのカウンタ現在値が 1 インクリメントされます。 // カウンタ値は、CU パラメータの信号立ち上がりエッジで、INT の上限値 = 32767 に達するまでインクリメントされる。
CD := "Tag_StartCTD"	//「Tag_StartCTD」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CounterValue」オペランドのカウンタ現在値が 1 デクリメントされます。 // CV パラメータのカウンタ値は、下限値 = -32768 に達するまで、デクリメントされる。

R := "Tag_ResetCounter"	// オペランド「Tag_ResetCounter」のシグナル状態が「1」に切り替わると、オペランド「Tag_CounterValue」は「0」にリセットされる。
LD := "Tag_LoadPV"	// オペランド「Tag_LoadPV」のシグナル状態が「1」に切り替わると、オペランド「Tag_CounterValue」はオペランド「Tag_PresetValue」の値にセットされる。
PV := "Tag_PresetValue"	// LD パラメータでシグナル状態が「1」のときにセットされるカウンタ値を指定する。
QU := "Tag_CounterStatus"	// オペランドは、カウンタ現在値が PV パラメータの値以上である限り、セットされる。
QD := "Tag_CounterStatus"	// カウンタ現在値がゼロ以下になると、オペランドがセットされる。
CV := "Tag_CounterValue"	// カウンタ現在値

比較演算

この章には下記に関する情報が記載されています：

- EQ_Type: EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較 (S7-1500)
- NE_Type: UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較 (S7-1500)
- EQ_ElemType: EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較 (S7-1500)
- NE_ElemType: UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較 (S7-1500)
- IS_NULL: IS NULL のチェック (S7-1500)
- NOT_NULL: NOT NULL のチェック (S7-1500)
- IS_ARRAY: 配列のチェック (S7-1500)

EQ_Type: EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較

説明

「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグのデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプが、IN2 パラメータのタグのデータタイプと「等しい」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプです。

構文

次の構文は、「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令に使用されます。

```
CALL EQ_Type
IN1 := <operand>
IN2 := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	L	最初のオペランド
IN2	Input	2 進数、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
STL
CALL EQ_Type
IN1 := #Tag_Operand1
IN2 := "Tag_Operand2"
RET_VAL := "Tag_Result"
```

「Tag_Result」出力は、比較命令の条件が満たされると、シグナル状態「1」を返します。つまり、「#Tag_Operand1」オペランドは「Tag_Operand2」に等しくなります。

NE_Type: UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較

説明

「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグが所有しないデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプが、IN2 パラメータのタグのデータタイプと「等しくない」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプです。

構文

次の構文は、「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令に使用されます。

```
CALL NE_Type
IN1 := <operand>
IN2 := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	L	最初のオペランド
IN2	Input	2 進数、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

結果

次の例で、命令がどのように動作するかを示します。

```
STL
CALL NE_Type
IN1 := #Tag_Operand1
IN2 := "Tag_Operand2"
```

```
RET_VAL := "Tag_Result"
```

「Tag_Result」出力は、比較命令の条件が満たされると、シグナル状態「1」を返します。つまり、「#Tag_Operand1」オペランドは「Tag_Operand2」に等しくなくなります。

EQ_ElemType: EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較

説明

「EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグのデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプが、IN2 パラメータのタグのデータタイプと「等しい」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプです。

VARIANT タグのデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

構文

次の構文は、「EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令に使用されます。

```
CALL EQ_ElemType
IN1 := <operand>
IN2 := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	L	最初のオペランド
IN2	Input	2 進数、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

結果

次の例で、命令がどのように動作するかを示します。

```
STL
CALL EQ_ElemType
```

```
IN1 := #Tag_Operand1
IN2 := "Tag_Operand2"
RET_VAL := "Tag_Result"
```

「Tag_Result」出力は、比較命令の条件が満たされると、シグナル状態「1」を返します。つまり、「#Tag_Operand1」オペランドは「Tag_Operand2」に等しくなります。

NE_ElemType: UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較

説明

「UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令を使用して、VARIANT が指すタグが所有しないデータタイプを照会できます。 ブロックインターフェースで宣言した IN1 パラメータのタグのデータタイプが、IN2 パラメータのタグのデータタイプと「等しくない」かどうか比較します。

IN1 パラメータのタグのデータタイプは、VARIANT であることが必要です。 IN2 パラメータのタグは、基本データタイプまたは PLC データタイプです。

VARIANT タグのデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

構文

次の構文は、「UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令に使用されます。

```
CALL NE_ElemType
IN1  := <operand>
IN2  := <operand>
RET_VAL  := <operand>
```

パラメータ

次の表に、「UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	VARIANT	L	最初のオペランド
IN2	Input	2 進数、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、PLC データタイプ	I、Q、M、D、L、P	2 番目のオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

結果

次の例で、命令がどのように動作するかを示します。

```
STL
CALL NE_ElemType
```

```
IN1 := #Tag_Operand1
IN2 := "Tag_Operand2"
RET_VAL := "Tag_Result"
```

「Tag_Result」出力は、比較命令の条件が満たされると、シグナル状態「1」を返します。つまり、「#Tag_Operand1」オペランドは「Tag_Operand2」に等しくなくなります。

IS_NULL: IS NULL のチェック

説明

「IS NULL のチェック」命令を使用して、VARIANT が NULL ポインタをポイントし、そのためオブジェクトをポイントしないかどうかを照会することができます。

OPERAND パラメータのタグのデータタイプは、VARIANT であることが必要です。

注記

VARIANT タグは、ANY ポインタを指します。

VARIANT タグが ANY ポインタを指す場合、この命令は常に(ANY ポインタがゼロである場合でも)、結果 RLO = 「0」を返します。

構文

「IS NULL のチェック」命令には、以下の構文を使用します。

```
CALL IS_NULL
OPERAND := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「IS NULL のチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OPERAND	Input	VARIANT	L	EQUAL ZERO (ゼロに等しい)かどうかで比較されるオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
STL
CALL IS_NULL
OPERAND := #Tag_Operand
RET_VAL := "Tag_Result"
```

「Tag_Result」出力は、比較命令の条件が満たされると、シグナル状態「1」を返します。つまり、#Tag_Operand オペランドはオブジェクトをポイントしません。

NOT_NULL: NOT NULL のチェック

説明

「NOT NULL のチェック」命令を使用して、VARIANT が ZERO ポインタをポイントせず、そのためオブジェクトをポイントするかどうかを照会することができます。

OPERAND パラメータのタグのデータタイプは、VARIANT であることが必要です。

注記

VARIANT タグは、ANY ポインタを指します。

VARIANT タグが ANY ポインタを指す場合、この命令は常に(ANY ポインタがゼロである場合でも)、結果 RLO = 「1」を返します。

構文

「NOT NULL のチェック」命令には、以下の構文を使用します。

```
CALL NOT_NULL
OPERAND := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「NOT NULL のチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OPERAND	Input	VARIANT	L	NOT EQUAL ZERO (ゼロに等しくない)かどうかで比較されるオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
STL
CALL NOT_NULL
OPERAND := #Tag_Operand
RET_VAL := "Tag_Result"
```

「Tag_Result」出力は、比較命令の条件が満たされると、シグナル状態「1」を返します。つまり、#Tag_Operand オペランドはオブジェクトをポイントします。

IS_ARRAY: 配列のチェック

説明

「配列のチェック」命令を使用して、VARIANT が ARRAY データタイプのタグを指すかどうかを照会できます。

OPERAND パラメータのタグのデータタイプが VARIANT です。 照会の結果は、RET_VAL パラメータに出力されます。

構文

「配列のチェック」命令には、以下の構文を使用します。

```
CALL IS_ARRAY
OPERAND := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「配列のチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OPERAND	Input	VARIANT	L	ARRAY に関する照会が行われるオペランド
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL IS_ARRAY	// 命令が呼び出されます。
OPERAND := #Tag_Input	// #Tag_Input オペランドの VARIANT が ARRAY を指した場合、比較命令が実行されます。
RET_VAL := "Tag_Result"	// 命令の結果

四則演算

この章には下記に関する情報が記載されています：

- MIN: 最小値の取得 (S7-1500)
- MAX: 最大値の取得 (S7-1500)
- LIMIT: 制限値の設定 (S7-1500)

MIN: 最小値の取得

説明

「最小値の取得」命令は入力 IN1、IN2 および IN3 の値を比較し、最小値を OUT 出力に書き込みます。 命令は、すべての入力のタグが同じデータタイプの場合にのみ実行されます。

OUT パラメータの値は、次の条件の 1 つが満たされると無効です。

- 指定された複数のタグが同じデータタイプでないこと。
- 浮動小数点数の値が無効であること。

構文

以下の構文が「最小値の取得」命令に使用されます。

```
CALL MIN
???
IN1 := <operand>
IN2 := <operand>
IN3 := <operand>
OUT := <operand>
```

パラメータ

次の表に、「最小値の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	2 番目の入力値
IN3	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	3 番目の入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	結果
IEC チェックが有効でない場合も、等しい長さのビット列または整数を命令のデータタイプとして選択することによって、データタイプ TIME、LTIME、TOD、LTOD、DATE および LDT のタグを使用することができます(TIME => DINT の代わりに、UDINT または DWORD = 32 ビットなど)。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL MIN	// 命令が呼び出されます。
INT	// 命令のデータタイプ
IN1 := "TagIn_Value1"	// 最初の入力値が比較される。
IN2 := "TagIn_Value2"	// 比較される 2 番目の入力値。
IN3 := "TagIn_Value3"	// 比較される 3 番目の入力値。
OUT := "Tag_Minimum"	// 比較の最小値

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Minimum	12222

MAX: 最大値の取得

説明

「最大値の取得」命令は、入力 IN1、IN2 および IN3 の値を比較し、最大値を OUT 出力に書き込みます。 命令は、すべての入力のタグが同じデータタイプの場合にのみ実行されます。

OUT 出力の値は、次の条件の 1 つが満たされると無効です。

- 指定された複数のタグが同じデータタイプでないこと。
- 浮動小数点数の値が無効であること。

構文

以下の構文が「最大値の取得」命令に使用されます。

```
CALL MAX
???
IN1 := <operand>
IN2 := <operand>
IN3 := <operand>
OUT := <operand>
```

パラメータ

次の表に、「最大値の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	2 番目の入力値
IN3	Input	整数、浮動小数点数	I、Q、M、D、L、P、または定数	3 番目の入力値
OUT	Output	整数、浮動小数点数	I、Q、M、D、L、P	結果
IEC チェックが有効でない場合も、等しい長さのビット列または整数を命令のデータタイプとして選択することによって、データタイプ TIME、LTIME、TOD、LTOD、DATE および LDT のタグを使用することができます(TIME => DINT の代わりに、UDINT または DWORD = 32 ビットなど)。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL MAX	// 命令が呼び出されます。
INT	// 命令のデータタイプ
IN1 := "TagIn_Value1"	// 最初の入力値が比較される。
IN2 := "TagIn_Value2"	// 比較される 2 番目の入力値。
IN3 := "TagIn_Value3"	// 比較される 3 番目の入力値。
OUT := "Tag_Maximum"	// 比較の最大値

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
IN3	TagIn_Value3	13333
OUT	Tag_Maximum	14444

LIMIT: 制限値の設定

説明

「制限値の設定」命令を使用して、IN 入力の値を入力 MN と MX の値に制限することができます。 IN 入力の値が MN 条件<= IN <= MX を満たす場合、この値は OUT 出力にコピーされます。 条件が満たされず IN 入力値が下限値 MN を下回ると、OUT 出力は MN 入力の値にセットされます。 上限値 MX を超える場合、OUT 出力は MX 入力の値にセットされます。

命令は、入力のすべてのタグが同じデータタイプの場合のみ実行されます。

OUT 出力の値は、次の条件の 1 つが満たされると無効です。

- 指定された複数のタグが同じデータタイプでないこと。
- オペランドの値が無効である。
- MN パラメータの値が MX パラメータの値よりも大きいこと。

構文

以下の構文が「制限値の設定」命令に使用されます。

```
CALL LIMIT
???
MN  := <operand>
IN  := <operand>
MX  := <operand>
OUT := <operand>
```

パラメータ

次の表に、「制限値の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MN	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	下限値
IN	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	入力値
MX	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	上限値

OUT	Output	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	**説明**
CALL LIMIT	// 命令が呼び出されます。
INT	// 命令のデータタイプ
MN := "Tag_LowLimit"	// 下限値
IN := "Tag_InputValue"	// 入力値
MX := "Tag_HighLimit"	// 上限値
OUT := "Tag_Result"	// 結果

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	**オペランド**	**値**
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

オペランド「Tag_InputValue」の値が定義限界値範囲外のため、オペランド「Tag_Result」は「Tag_LowLimit」の値で書き込まれます。

移動操作

この章には下記に関する情報が記載されています：

MOVE: 値のムーブ

説明

「値のムーブ」命令を使用して、IN 入力のオペランドの内容を OUT 出力のオペランドに転送できます。 入力 IN および出力 OUT のオペランドのデータタイプは同一である必要があります。

次の表に、可能な転送を示します。

転送元(IN)	宛先(OUT)
BYTE	BYTE
WORD	WORD
DWORD	DWORD
LWORD	LWORD
SINT	SINT
USINT	USINT
INT	INT
UINT	UINT
DINT	DINT
UDINT	UDINT
LINT	LINT
ULINT	ULINT
REAL	REAL
LREAL	LREAL
S5TIME	S5TIME
TIME	TIME
LTIME	LTIME
DATE	DATE
DT	DT
LDT	LDT
TOD	TOD
LTOD	LTOD
DTL	DTL
CHAR	CHAR
WCHAR	WCHAR
文字列の文字 1)	文字列の文字
ARRAY	ARRAY
STRUCT	STRUCT
PLC データタイプ(UDT)	PLC データタイプ(UDT)
IEC_TIMER	IEC_TIMER

IEC_LTIMER	IEC_LTIMER
IEC_SCOUNTER	IEC_SCOUNTER
IEC_USCOUNTER	IEC_USCOUNTER
IEC_COUNTER	IEC_COUNTER
IEC_UCOUNTER	IEC_UCOUNTER
IEC_DCOUNTER	IEC_DCOUNTER
IEC_UDCOUNTER	IEC_UDCOUNTER
IEC_LCOUNTER	IEC_LCOUNTER
IEC_ULCOUNTER	IEC_ULCOUNTER
DB_ANY	DB_ANY

1) 「値のムーブ」命令を使用して、文字列の個々の文字を CHAR または WCHAR データタイプのオペランドに転送することもできます。 転送する文字数は、オペランド名の隣の角括弧内に指定されます。 "たとえば「MyString[2]」は、「MyString」文字列の 2 番目の文字を転送します。 データタイプ CHAR のオペランドから文字列の個々の文字を転送することもできます。 文字列の特定の文字を別の文字列の文字と置換することもできます。

「ブロックムーブ」(MOVE_BLK)、「割り込み不可能なブロックムーブ」(UMOVE_BLK)、および「ブロックムーブ」(MOVE_BLK_VARIANT)命令を使用して、ARRAY データタイプのオペランドを移動することができます。 STRING データタイプのオペランドを「文字列移動」命令(S_MOVE)で移動することができます。

構文

以下の構文が「値の移動」命令に使用されます。

```
CALL MOVE
VARIANT
IN := <operand>
OUT := <operand>
```

パラメータ

次の表に、「値の移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BIT 文字列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、IEC データタイプ、PLC データタイプ (UDT)、DB_ANY	I、Q、M、D、L、または定数	宛先アドレスの上書きに使用するエレメント
OUT	Output	BIT 文字列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列、IEC データタイプ、	I、Q、M、D、L	宛先アドレス

		PLC データタイプ (UDT)、DB_ANY		

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL MOVE	// 命令が呼び出されます。
VARIANT	// 命令のデータタイプ
IN := "TagIn_LREAL"	// オペランド「TagIn_LREAL」の内容が移動される。
OUT := "TagOut_LREAL"	// さらに、オペランド「TagOut_LREAL」に転送される。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	TagIn_LREAL	1.23456789098e55
OUT	TagOut_LREAL	1.23456789098e55

「TagIn」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 この命令は、「TagIn_LREAL」オペランドの内容を「TagOut_LREAL」オペランドに移動します。

Deserialize: 逆シリアル化

説明

「逆シリアル化」命令を使用して、PLC データタイプ(UDT)のシーケンシャル表示を PLC データタイプに変換して戻し、その内容全体を表示することができます。

PLC データタイプのシーケンシャル表示を配置するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。変換の前に、十分なメモリ領域が存在することを確認してください。

この命令では、変換された PLC データタイプの複数のシーケンシャル表示をそれらの元の状態に、順を追って変換し直すことができます。

PLC データタイプ(UDT)の単一のシーケンシャル表示のみを変換して元に戻す場合は、命令「TRCV: 通信接続経由のデータ受信」を直接に使用することもできます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

構文

「逆シリアル化」命令には、以下の構文を使用します。

```
CALL Deserialize
SRC_ARRAY := <operand>
RET_VAL := <operand>
DEST_VARIABLE := <operand>
POS := <operand>
```

パラメータ

次の表に、「逆シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_ARRAY	Input	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるグローバルデータブロック
DEST_VARIABLE	InOut	VARIANT	I、Q、M、L	返された変換済み PLC データタイプ(UDT)が格納されるタグ
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_ARRAY パラメータと DEST_VARIABLE パラメータのためのメモリ領域がオーバーラップしています。
8136	DEST_VARIABLE パラメータのデータブロックが標準アクセスによるブロックではありません。
8150	SRC_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8151	SRC_ARRAY パラメータでコード生成エラーが発生しました
8153	SRC_ARRAY パラメータのメモリの空き領域が不足しています。
8250	DEST_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8251	DEST_VARIABLE パラメータでコード生成エラーが発生しました
8254	DEST_VARIABLE パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L 0	// 値「0」がアキュムレータ 1 に読み込まれます。
T #BufferPos	// 値「0」が#BufferPos オペランドに転送される。
CALL Deserialize	// 命令が呼び出されます。
SRC_ARRAY := "Buffer".Field	// 「Buffer」データブロックのカスタマデータが逆シリアル化されます。
RET_VAL := #Error	// エラー情報
DEST_VARIABLE := "Target".Client	// 逆シリアル化されたカスタマデータが、「Target」データブロックに書き込まれます。
POS := #BufferPos	//変換済みカスタマデータで占有されるバイト数は、#BufferPos オペランドに格納されます。
AN BR JC end	// 命令の実行中にエラーが発生すると、プログラムの処理が終了までジャンプします。
CALL Deserialize	// 命令が呼び出されます。

```
                                        // 「Buffer」データブロックのセパレータシートが逆シリアル化されます。 カスタマデータのわかりやすい区別を可能とするために、セパレータシートが「arti」または「Bill」文字形式で挿入されました。
SRC_ARRAY := "Buffer".Field
RET_VAL := #Error                       // エラー情報
DEST_VARIABLE := #Label                 // セパレータシートの逆シリアル化された文字が、「#Label」オペランドに書き込まれます。
POS := #BufferPos                       // 変換済み文字で使用されるバイト数は、#BufferPos オペランドに格納されます。

AN BR                                   // 命令の実行中にエラーが発生すると、プログラムの処理が終了までジャンプします。
JC end

CALL S_COMP                             // 命令が呼び出されます。
src_type := STRING                      // 命令のデータタイプ
relation := EQ
IN1 := 'arti'                           // 文字「arti」と、#Label オペランドに格納された文字の比較。
IN2 := #Label
OUT := #cmp                             // 命令の結果

A #cmp                                  // この比較がシグナル状態「0」を返す場合、#Labelオペランドの文字が「arti」文字でなく、プログラムの処理は「noarticle」ジャンプラベルにジャンプします。
JCN noarticle

CALL Deserialize                        // 比較の結果が「1」の場合、命令を呼び出す。
SRC_ARRAY := "Buffer".Field             // アーティクルデータが逆シリアル化されます。
RET_VAL := #Error                       // エラー情報
DEST_VARIABLE := "Target".Article[#Deliver-
Pos]                                    // 逆シリアル化されたアーティクルデータが、「Target」データブロックに書き込まれます。
POS := #BufferPos                       // 変換済みアーティクルデータで使用されるバイト数は、「#BufferPos」オペランドに格納されます。

AN BR                                   // 処理中にエラーが発生しなかった場合、ブロック終了です。
JC end
BE

noarticle: NOP 0                        //「arti」かどうかの比較の結果が =「0」の場合、プログラムの処理はここに続きます。

CALL S_COMP                             // 命令が呼び出されます。
src_type := STRING                      // 命令のデータタイプ
relation := EQ
IN1 := 'Bill'                           // 文字「Bill」と、#Label オペランドに格納された文字の比較。
IN2 := #Label
```

```
OUT := #cmp                                  // 命令の結果

A #cmp                                       // この比較がシグナル状態「1」を返す場合、#Label
JCN end                                      オペランドの文字は「'Bill」文字で、プログラム
                                             の処理が続行します。

CALL Deserialize                             // 比較の結果が「1」の場合、命令を呼び出す。
SRC_ARRAY := "Buffer".Field                  // ビルデータが逆シリアル化されます。
RET_VAL := #Error                            // エラー情報
DEST_VARIABLE := "Target".Bill[#DeliverPos]  // 逆シリアル化されたカスタマデータが、「Tar-
                                             get」データブロックに書き込まれます。
POS := #BufferPos                            // 変換済みアーティクルデータで使用されるバイ
                                             ト数は、「#BufferPos」オペランドに格納されま
                                             す。

AN BR                                        // 処理中にエラーが発生しなかった場合、ブロッ
JC end                                       ク終了です。
BE

end: CLR                                     // エラーの場合、ジャンプラベル「end」へのジ
                                             ャンプが実行されます。
SAVE                                         // RLO が BR ビットに保存されます。
```

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	
Label	STRING[4]	
cmp	BOOL	

次の表に、PLC データタイプの宣言をリストします。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Target	Client	「クライアント」(PLC データタイプ)
	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
	Bill	INT の配列[0..10]
Buffer	Field	バイトの配列[0..294]

Serialize: シリアル化

説明

「シリアル化」命令を使用して、データタイプの構造体の部分を失うことなく、複数の PLC データタイプ(UDT)を 1 つのシーケンシャル表示に変換することができます。

この命令を使用して、ユーザプログラムの複数の構造体データ項目を一時的にバッファ(グローバルデータブロックを推奨)に保存し、別の CPU に送信することができます。 変換済み PLC データタイプを格納するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。 変換の前に、十分なメモリ領域が存在することを確認してください。

POS パラメータのオペランドには、変換済み PLC データタイプによって使用されるバイト数に関する情報が収納されています。

単一の PLC データタイプ(UDT)を送信する場合、直接に命令「TSEND: 通信接続経由のデータ送信」を呼び出すことができます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

構文

「シリアル化」命令には、以下の構文を使用します。

```
CALL Serialize
SRC_VARIABLE := <operand>
RET_VAL := <operand>
DEST_ARRAY := <operand>
POS := <operand>
```

パラメータ

次の表に、「シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_VARIABLE	Input	VARIANT	I、Q、M、L	シーケンシャル表示に変換される PLC データタイプ (UDT)。
DEST_ARRAY	InOut	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるデータブロック。
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS

				パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_VARIABLE パラメータと DEST_ARRAY パラメータのためのメモリ領域がオーバーラップしています。
8150	SRC_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8152	SRC_VARIABLE パラメータでコード生成エラーが発生しました
8236	DEST_ARRAY パラメータのデータブロックが標準アクセスによるブロックではありません。
8250	DEST_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8252	DEST_ARRAY パラメータでコード生成エラーが発生しました
8253	DEST_ARRAY パラメータのメモリの空き領域が不足しています。
8254	DEST_ARRAY パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L 0	// 値「0」がアキュムレータ 1 に読み込まれます。
T #BufferPos	// 値「0」が#BufferPos オペランドに転送される。
CALL Serialize	// 命令が呼び出されます。
SRC_VARIABLE := "Source".Client	// "Source".Client データブロックのカスタマデータがシリアル化されます。
RET_VAL := #Error	// エラー情報
DEST_ARRAY := "Buffer".Field	// シーケンシャル表示が、"Buffer".Field データブロックに書き込まれます。
POS := #BufferPos	// シーケンシャル表示によって占有されたバイト数は、#BufferPos オペランドに格納されます。
CALL S_MOVE	// 命令が呼び出されます。

```
src_type := STRING                                  // 命令のデータタイプ
dest_type := STRING
IN := 'arti'                                        // 'arti'文字が#Label オペランドにコピーされます。
OUT := #Label

CALL Serialize                                      // 命令が呼び出されます。
SRC_VARIABLE := #Label                              / / 文字がシリアル化されます。
RET_VAL := #Error                                   // エラー情報
DEST_ARRAY := "Buffer".Field                        // シーケンシャル表示が、"Buffer".Field データブロックのカスタマデータの後に書き込まれます。
POS := #BufferPos                                   // 文字が必要とするバイト数が、#BufferPos オペランドに格納済みの数に加算されます。

CALL Serialize                                      // 命令が呼び出されます。
SRC_VARIABLE := "Source".Article[#DeliverPos]       // ランタイム中に計算される、"Source".Article[#DeliverPos]データブロックの特定のアーティクルのデータがシリアル化されます。
RET_VAL := #Error                                   // エラー情報
DEST_ARRAY := "Buffer".Field                        // シーケンシャル表示が、"Buffer".Field データブロックの'arti'文字の後に書き込まれます。
POS := #BufferPos                                   // バイト数
```

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言をリストします。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Source	Client	「クライアント」(PLC データタイプ)
	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
Buffer	Field	バイトの配列[0..294]

MOVE_BLK: ブロックムーブ

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 宛先領域にコピーするエレメントの数は、COUNT パラメータで指定できます。 移動されるエレメントの幅は、IN パラメータのエレメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

OUT 出力の値は、IN パラメータまたは OUT パラメータで使用可能にされたよりも多くのデータが移動された場合、無効です。

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
CALL MOVE_BLK
??? ???
IN := <operand>
COUNT := <operand>
OUT := <operand>
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	ソース領域から宛先領域にコピーするエレメントの数。
OUT [1)]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント
[1)] 指定されたデータタイプは、Array 構造体のエレメントとしてしか使用できません。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL MOVE_BLK	// 命令が呼び出されます。
value_type := INT	// 命令のデータタイプ
count_type := UINT	// COUNT パラメータのデータタイプ
IN := #a_array[2]	// 移動される最初のエレメント。
COUNT := "Tag_Count"	// 移動元領域から移動先領域に移動するエレメントの数。
OUT := #b_array[1]	// 移動先の最初のエレメント。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。 これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。

MOVE_BLK_VARIANT: ブロックムーブ

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 配列全体または配列の工レメントを同じデータタイプの別の配列にコピーすることができます。 ソース配列と宛先配列のサイズ(工レメントの数)は異なる場合があります。配列内の複数工レメントまたは単一工レメントをコピーすることができます。

ブロックの作成時にこの命令を使用する場合、VARIANT ごとにソースおよび宛先が転送されるため、配列はまだ既知である必要はありません。

SRC_INDEX および DEST_INDEX パラメータでのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

コピーされるデータが使用可能なデータよりも多い場合、この命令は実行されません。

注記

BOOL データタイプと接続している VARIANT

VARIANT データタイプのパラメータ(ソース領域または宛先領域)と BOOL データタイプのタグまたは ARRAY of BOOL を相互接続する場合、以下のオプションがあります。

1. シンボリックにアドレス指定することができます。

   例: SRC パラメータ: "Data_block".myArray
2. 任意のポインタによってアドレス指定することができます。 ただし、領域で指定された長さは 8 で割り切れるようにする必要があります。割り切れない場合、この命令は実行されません。

   例: SRC パラメータ: P#DB123.DBX456.0 BOOL 1000

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
CALL MOVE_BLK_VARIANT
SRC := <operand>
COUNT := <operand>
SRC_INDEX := <operand>
DEST_INDEX := <operand>
DEST := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	VARIANT (配列または個々の配列エレメントをポイントする)、<Data_type>の配列	I、Q、M、L	コピー元のソースブロック
COUNT	Input	UDINT	I、Q、M、D、L、または定数	コピーされるエレメント数 SRC が DEST パラメータまたは ARRAY パラメータで指定されていない場合は、COUNT パラメータに値「1」を割り当てます。
SRC_INDEX	Input	DINT	I、Q、M、D、L、または定数	• SRC_INDEX パラメータは、ゼロベースで計算されます。 SRC がパラメータ ARRAY で指定されると、パラメータ SRC_INDEX の整数はコピー元のソース領域内の最初のエレメントを指定します。 宣言された配列の限界値には関係ありません。 • SRC がパラメータ ARRAY で指定されないか、または配列の 1 つの単一エレメントのみが指定された場合、パラメータ SRC_INDEX に値「0」を割り当てます。
DEST_INDEX	Input	DINT	I、Q、M、D、L、または定数	• DEST_INDEX パラメータは、ゼロベースで計算されます。 DEST がパラメータ ARRAY で指定されると、パラメータ DEST_INDEX の整数はコピー先の宛先領域内の最初のエレメントを指定します。 宣言された配列の限界値には関係ありません。 • DEST が ARRAY パラメータで指定されていない場合は、DEST_INDEX パラメータに値「0」を割り当てます。
DEST	Output [1)]	VARIANT	I、Q、M、L	ソースブロックの内容がコピーされる宛先領域。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

1) データがタグに流れるため、DEST パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	データタイプが一致していません
8151	SRC パラメータにアクセスできません。
8152	SRC パラメータのオペランドが入力されていません。
8153	SRC パラメータでコード生成エラーが発生しました
8154	SRC パラメータのオペランドのデータタイプが BOOL です。
8281	COUNT パラメータの値が無効です
8382	SRC_INDEX パラメータの値が VARIANT の限界値外です。
8383	SRC_INDEX パラメータの値が配列の上限値外です
8482	DEST_INDEX パラメータの値が VARIANT の限界値外です。
8483	DEST_INDEX パラメータの値が配列の上限値外です
8534	DEST パラメータが書き込み保護されています
8551	DEST パラメータにアクセスできません。
8552	DEST パラメータのオペランドが入力されていません。
8553	DEST パラメータでコード生成エラーが発生しました
8554	DEST パラメータのオペランドのデータタイプが BOOL です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL MOVE_BLK_VARIANT	// 命令が呼び出されます。
SRC := #SrcField	// 移動される最初のエレメント
COUNT := "Tag_Count"	// 移動元領域から移動先領域に移動するエレメントの数。
SRC_INDEX := "Tag_Src_Index"	// 移動される SRC パラメータの最初のエレメント
DEST_INDEX := "Tag_Dest_Index"	// 移動される DEST パラメータの最初のエレメント
DEST := #DestField	// 移動される移動元領域の最初のエレメント。
RET_VAL := "Tag_Result"	// エラー情報

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
SRC	Input	#SrcField	ローカルオペランド #SrcField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。 ("MOVE_UDT"の ARRAY[0..10]
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	ローカルオペランド #DestField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。 ("MOVE_UDT"の ARRAY[10..20]

UDT の配列の 4 番目の工レメントから始めて、2 つの工レメントが、ソース領域から宛先領域に移動されます。 コピーが、UDT の配列の 4 番目の工レメントから始めて、挿入されます。

UMOVE_BLK: 割り込み不可能なブロックムーブ

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 この命令に割り込みをかけることはできません。 宛先領域にコピーするエレメントの数は、COUNT パラメータで指定できます。 コピーされるエレメントの幅は、IN 入力のエレメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

OUT 出力の値は、IN パラメータまたは OUT パラメータで使用可能にされたよりも多くのデータが移動された場合、無効です。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

構文

以下の構文が「割り込みなしブロック転送」命令に使用されます。

```
CALL UMOVE_BLK
??? ???
IN  := <operand>
COUNT  := <operand>
OUT  := <operand>
```

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	ソース領域から宛先領域にコピーするエレメントの数。
OUT [1)]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント

1) 指定されたデータタイプは、Array 構造体のエレメントとしてしか使用できません。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL UMOVE_BLK	// 命令が呼び出されます。
value_type := INT	// 命令のデータタイプ
count_type := UINT	// COUNT パラメータのデータタイプ
IN := #a_array[2]	// 移動される最初のエレメント。
COUNT := "Tag_Count"	// 移動元領域から移動先領域に移動するエレメントの数。
OUT := #b_array[1]	// 移動先の最初のエレメント。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。 他のオペレーティングシステムのアクティビティによって、コピー操作に割り込みをかけることはできません。

FILL_BLK: フィル命令

説明

「フィル」命令を使用して、IN 入力の値でメモリ領域(宛先領域)を埋めることができます。 宛先領域は、OUT 出力で指定されたアドレスから開始して埋められます。 コピー操作の繰り返し回数は、COUNT パラメータの値で指定されます。 命令が実行されると、IN 入力の値が選択され、COUNT パラメータの値で指定された回数で宛先領域にコピーされます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

OUT 出力の値は、IN 入力または OUT 出力で使用可能にされたよりも多くのデータが移動された場合、無効です。

構文

「フィル」命令には、以下の構文を使用します。

```
CALL FILL_BLK
??? ???
IN  := <operand>
COUNT  := <operand>
OUT  := <operand>
```

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	コピー操作の繰り返し回数
OUT [2)]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	フィル命令が開始する宛先領域のアドレス
[1)] 指定されたデータタイプは、Array 構造体の工レメントとして使用することもできます。 [2)] 指定されたデータタイプは、Array 構造体の工レメントとしてしか使用できません。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL FILL_BLK	// 命令が呼び出されます。
value_type := INT	// 命令のデータタイプ
count_type := UINT	// COUNT パラメータのデータタイプ
IN := a_array[2]	// 移動される エレメント。
COUNT := "Tag_Count"	// コピー操作の繰り返し回数
OUT := b_array[1]	// 宛先領域は、OUT 出力で指定されたアドレスから開始して埋められる。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。 これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。

UFILL_BLK: 割り込み不可能なフィル命令

説明

「フィル」命令を使用して、IN 入力の値でメモリ領域(宛先領域)を埋めることができます。 この命令に割り込みをかけることはできません。 宛先領域は、OUT 出力で指定されたアドレスから開始して埋められます。 コピー操作の繰り返し回数は、COUNT パラメータの値で指定されます。 命令が実行されると、IN 入力の値が選択され、COUNT パラメータの値で指定された回数で宛先領域にコピーされます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込み不可能なフィル」命令の実行中には CPU の割り込み応答時間が長くなります。

OUT 出力の値は、IN 入力または OUT 出力で使用可能にされたよりも多くのデータが移動された場合、無効です。

「割り込み不可能なフィル」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

構文

「割り込み不可能なフィル」命令には、以下の構文を使用します。

```
CALL UFILL_BLK
??? ???
IN  := <operand>
COUNT  := <operand>
OUT  := <operand>
```

パラメータ

次の表に、「割り込み不可能なフィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	コピー操作の繰り返し回数
OUT [2)]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、	D、L	フィル命令が開始する宛先領域のアドレス

		DATE、CHAR、WCHAR		

1) 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。

2) 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL UFILL_BLK	// 命令が呼び出されます。
value_type := INT	// 命令のデータタイプ
count_type := UINT	// COUNT パラメータのデータタイプ
IN := "Tag_IN"	// 移動される工レメント
COUNT := "Tag_Count"	// コピー操作の繰り返し回数
OUT := "Tag_OUT"	// 宛先領域は、OUT 出力で指定されたアドレスから開始して埋められる。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つの工レメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つの工レメントで構成されます。

この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。 他のオペレーティングシステムのアクティビティによって、コピー操作に割り込みをかけることはできません。

配列 DB

この章には下記に関する情報が記載されています：

- ReadFromArrayDB: 配列データブロックからの読み出し (S7-1500)
- WriteToArrayDB: 配列データブロックへの書き込み (S7-1500)
- ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し (S7-1500)
- WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み (S7-1500)

ReadFromArrayDB: 配列データブロックからの読み出し

説明

「配列データブロックからの読み出し」命令を使用し、配列データブロックからデータを読み出して、宛先領域に書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

構文

「ARRAY データブロックからの読み出し」命令には、以下の構文を使用します。

```
CALL ReadFromArrayDB
DB := <operand>
INDEX := <operand>
RET_VAL := <operand>
VALUE := <operand>
```

パラメータ

次の表に、「配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	Output 1)	VARIANT	I、Q、M、L	読み出されて出力される値
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
1) データがタグに流れるため、VALUE パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、書き込み保護されているか、またはロードメモリ内に存在します。
8135	配列データブロックに無効な値が含まれます。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8450	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8452	コード生成エラー
8453	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL ReadFromArrayDB	// 命令の呼び出し
DB := "ArrayDB"	// データの読み出し元データブロック。
INDEX := "ArrayDB".THIS[2]	// データブロックからエレメント[2]が読みだされます。
RET_VAL := "TagRet_Val"	// エラー情報
VALUE := "TargetField[10]".Data2[1]	// 読み出し値は、VALUE パラメータで提供されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	ArrayDB オペランドのデータタイプは、DINT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"TargetField[10]".Data2[1]	「TargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[10 to 20]です。

3 番目のエレメントは「ArrayDB」から読み出され、"TargetField[10]".Data2[1]オペランドに書き込まれます。

WriteToArrayDB: 配列データブロックへの書き込み

説明

「配列データブロックへの書き込み」命令を使用して、配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列での カウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

構文

「配列データブロックへの書き込み」命令には、以下の構文を使用します。

```
CALL WriteToArrayDB
DB := <operand>
INDEX := <operand>
RET_VAL := <operand>
VALUE := <operand>
```

パラメータ

次の表に、「配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内のエレメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。
8134	データブロックが書き込み保護されています。
8135	データブロックが配列データブロックではありません。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8350	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8352	コード生成エラー
8353	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL WriteToArrayDB	// 命令の呼び出し
DB := "ArrayDB"	// データが書き込まれるデータブロック
INDEX := "ArrayDB".THIS[2]	// 値が、データブロックのエレメント[2]に書き込まれます。
RET_VAL := "TagRet_Val"	// エラー情報
VALUE := "SourceField[1]".Data1[6]	// データブロックに書き込まれる値。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	ArrayDB オペランドのデータタイプは、DINT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceField[1]".Data1[6]	「SourceField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。

「SourceField」オペランドから、2 番目のエレメントの「Data1[6]」エレメントが「ArrayDB」に書き込まれます。 3 番目のエレメントは、「ArrayDB」に書き込まれます。

ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し

説明

「ロードメモリの配列データブロックからの読み出し」命令を使用して、ロードメモリの配列データブロックからデータを読み出します。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 この命令は、BUSY パラメータで信号立ち下がりエッジが検出された場合に終了します。 1 プログラムサイクルの間、DONE パラメータのシグナル状態が「1」になり、このサイクル内に、読み取られた値が VALUE パラメータに出力されます。 他のすべてのプログラムサイクルでは、VALUE パラメータの値は変更されません。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

構文

「ロードメモリの ARRAY データブロックからの読み出し」命令には、以下の構文を使用します。

```
CALL ReadFromArrayDBL, "インスタンス DB"
REQ := <operand>
DB := <operand>
INDEX := <operand>
VALUE := <operand>
BUSY := <operand>
DONE := <operand>
ERROR := <operand>
```

パラメータ

次の表に、「ロードメモリの配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」: 配列 DB の読み出しから開始
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	InOut	VARIANT	I、Q、M、L	読み出されて出力される値 TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ読み出し中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL: ロードメモリのデータブロックからの読み出し」および「WRIT_DBL: データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL ReadFromArrayDBL, "ReadFromArrayDBL_DB"	// 命令の呼び出し
REQ := "TagReg"	// この命令は、信号の立ち上がりエッジ時に実行されます。
DB := "ArrayDB"	// データの読み出し元データブロック。
INDEX := "ArrayDB".THIS[2]	// データブロックからエレメント[2]が読みだされます。
VALUE := "SourceTargetField[1]".Data1[6]	// 読み出し値は、VALUE パラメータで提供されます。
BUSY := "TagBusy"	// 命令がまだ処理中であるかどうかを示します。
DONE := "TagDone"	// 命令が終了しているかどうかを示します。
ERROR := "TagError"	// エラー情報

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	ArrayDB オペランドのデータタイプは、DINT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceTargetField[1]".Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

この命令は、「TagReq」オペランドで信号立ち上がりエッジが検出されたときに実行されます。 3 番目のエレメントは「ArrayDB」から読み出され、"SourceTargetField[1]".Data1[6]オペランドに出力されます。 「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値は変更されません。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み

説明

「ロードメモリの配列データブロックへの書き込み」命令を使用して、ロードメモリの配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 BUSY パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、VALUE パラメータの値がデータブロックに書き込まれます。 DONE1 パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、パラメータの値がデータブロックに書き込まれます。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

構文

「ロードメモリの ARRAY データブロックへの書き込み」命令には、以下の構文を使用します。

```
CALL WriteToArrayDBL, "インスタンス DB"
REQ := <operand>
DB := <operand>
INDEX := <operand>
VALUE := <operand>
BUSY := <operand>
DONE := <operand>
ERROR := <operand>
```

パラメータ

次の表に、「ロードメモリの配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」: 配列 DB への書き込みを開始
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内のエレメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値 TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ書き込み中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8234	データブロックが書き込み保護されています。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

命令「READ_DBL: ロードメモリのデータブロックからの読み出し」および「WRIT_DBL: データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL WriteToArrayDBL, "WriteToArrayDBL_DB"	// 命令の呼び出し
REQ := "TagReg"	// この命令は、信号の立ち上がりエッジ時に実行されます。
DB := "ArrayDB"	// データの読み出し元データブロック。
INDEX := "ArrayDB".THIS[2]	// データブロックからエレメント[2]が読みだされます。
VALUE := "SourceTargetField[1]".Data1[6]	// 読み出し値は、VALUE パラメータで提供されます。
BUSY := "TagBusy"	// 命令がまだ処理中であるかどうかを示します。
DONE := "TagDone"	// 命令が終了しているかどうかを示します。
ERROR := "TagError"	// エラー情報

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	ArrayDB オペランドのデータタイプは、DINT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceTargetField[1]".Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

この命令は、「TagReq」オペランドで信号立ち上がりエッジが検出されたときに実行されます。「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値が「ArrayDB」の 3 番目のエレメントに書き込まれます。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

読み取り/書き込みアクセス

この章には下記に関する情報が記載されています：

- PEEK: メモリアドレスの読み取り (S7-1500)
- PEEK_BOOL: メモリビットの読み取り (S7-1500)
- POKE: メモリアドレスの書き込み (S7-1500)
- POKE_BOOL: メモリビットの書き込み (S7-1500)
- POKE_BLK: メモリ領域の書き込み (S7-1500)
- READ_LITTLE: リトルエンディアン形式のデータの読み出し (S7-1500)
- WRITE_LITTLE: リトルエンディアン形式のデータの書き込み (S7-1500)
- READ_BIG: ビッグエンディアン形式のデータの読み出し (S7-1500)
- WRITE_BIG: ビッグエンディアン形式のデータの書き込み (S7-1500)

PEEK: メモリアドレスの読み取り

説明

「メモリアドレスの読み取り」命令は、メモリ領域からのデータタイプを指定しないメモリアドレスの読み取りに使用します。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリアドレスの読み取り」命令には、以下の構文を使用します。

```
CALL PEEK
??? ???
AREA := <operand>
DBNUMBER := <operand>
BYTEOFFSET := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「メモリアドレスの読み取り」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#1: I/O 入力
DBNUMBER	Input	DINT, DB_ANY	I、Q、M、D、L	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	読み取り元のアドレス 16 個の最下位ビットのみが使用されます。
RET_VAL	Output	ビット列	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

注記

入力、出力またはビットメモリ領域からメモリアドレスを読み取る場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL PEEK	// 命令が呼び出されます。
value_type := WORD number_type := DINT	// 命令のデータタイプ
AREA := "Tag_Area"	// データブロック領域が選択されました
DBNUMBER := "Tag_DBNumber"	// データブロックの番号
BYTEOFFSET := "Tag_Byte"	// 読み取り元のアドレス
RET_VAL := "Tag_Result"	// 結果

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
RET_VAL	Tag_Result	バイト値「20」

この命令は、データブロック「5」の「Tag_Byte」オペランドからアドレス「20」の値を読み取り、結果を「Tag_Result」オペランドに返します。

PEEK_BOOL: メモリビットの読み取り

説明

「メモリビットの読み取り」命令は、メモリビットからのデータタイプを指定しないメモリアドレスの読み取りに使用します。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリビットの読み取り」命令には、以下の構文を使用します。

```
CALL PEEK_BOOL
???
AREA := <operand>
DBNUMBER := <operand>
BYTEOFFSET := <operand>
BITOFFSET := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「メモリビットの読み取り」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#1: I/O 入力
DBNUMBER	Input	DINT, DB_ANY	I、Q、M、D、L	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	読み取り元のアドレス 16 個の最下位ビットのみが使用されます。
BITOFFSET	Input	INT	I、Q、M、D、L	読み取り元のビット
RET_VAL	Output	BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

注記
入力、出力またはビットメモリ領域からメモリビットを読み取る場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL PEEK_BOOL	// 命令が呼び出されます。
number_type := DINT	// DBNUMBER パラメータのデータタイプ
AREA := "Tag_Area"	// データブロック領域が選択されました
DBNUMBER := "Tag_DBNumber"	// データブロックの番号
BYTEOFFSET := "Tag_Byte"	// 読み取り元のアドレス
BITOFFSET := "Tag_Bit"	// 読み取り元のビット
RET_VAL := "Tag_Result"	// 結果

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
RET_VAL	Tag_Result	3

この命令は、データブロック「5」のバイト「20」で「Tag_Bit」オペランドからメモリビット「3」の値を読み取り、結果を「Tag_Result」オペランドに返します。

POKE: メモリアドレスの書き込み

説明

「メモリアドレスの書き込み」命令は、メモリ領域へのデータタイプを指定しないメモリアドレスの書き込みに使用します。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリアドレスの書き込み」命令には、以下の構文を使用します。

```
CALL POKE
??? ???
AREA := <operand>
DBNUMBER := <operand>
BYTEOFFSET := <operand>
VALUE := <operand>
```

パラメータ

次の表に、「メモリアドレスの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#2: I/O 出力
DBNUMBER	Input	DINT, DB_ANY	I、Q、M、D、L	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	書き込まれるアドレス 16 個の最下位ビットのみが使用されます。
VALUE	Input	ビット列	I、Q、M、D、L	書き込まれる値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

注記

入力、出力またはビットメモリ領域にメモリアドレスを書き込む場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL POKE	// 命令が呼び出されます。
value_type := WORD number_type := DINT	// 命令のデータタイプ
AREA := "Tag_Area"	// データブロック領域が選択されました
DBNUMBER := "Tag_DBNumber"	// データブロックの番号
BYTEOFFSET := "Tag_Byte"	// 読み取り元のアドレス
VALUE := "Tag_Value"	// 書き込まれる値

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
VALUE	Tag_Value	16#11

この命令は、データブロック「5」内のメモリ領域「20」を値「16#11」で上書きします。

POKE_BOOL: メモリビットの書き込み

説明

「メモリビットの書き込み」命令は、メモリビットへのデータタイプを指定しないメモリアドレスの書き込みに使用します。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリビットの書き込み」命令には、以下の構文を使用します。

```
CALL POKE_BOOL
???
AREA := <operand>
DBNUMBER := <operand>
BYTEOFFSET := <operand>
BITOFFSET := <operand>
VALUE := <operand>
```

パラメータ

次の表に、「メモリビットの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#2: I/O 出力
DBNUMBER	Input	DINT, DB_ANY	I、Q、M、D、L	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	書き込まれるアドレス 16 個の最下位ビットのみが使用されます。
BITOFFSET	Input	INT	I、Q、M、D、L	書き込まれるビット
VALUE	Input	BOOL	I、Q、M、D、L	書き込まれる値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

注記
入力、出力またはビットメモリ領域にメモリビットを書き込む場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL POKE_BOOL	// 命令が呼び出されます。
number_type := DINT	// DBNUMBER パラメータのデータタイプ
AREA := "Tag_Area"	// データブロック領域が選択されました
DBNUMBER := "Tag_DBNumber"	// データブロックの番号
BYTEOFFSET := "Tag_Byte"	// 読み取り元のアドレス
BITOFFSET := "Tag_Bit"	// 読み取り元のビット
VALUE := "Tag_Value"	// 書き込まれる値

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
VALUE	Tag_Value	M0.0

この命令は、バイト「20」内のデータブロック「5」のメモリビット「3」を値「M0.0」で上書きします。

POKE_BLK: メモリ領域の書き込み

説明

「メモリ領域の書き込み」命令は、メモリ領域の内容をデータタイプを指定せずに他のメモリ領域にコピーします。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリ領域の書き込み」命令には、以下の構文を使用します。

```
CALL POKE_BLK
???
AREA_SRC := <operand>
DBNUMBER_SRC := <operand>
BYTEOFFSET_SRC := <operand>
AREA_DEST := <operand>
DBNUMBER_DEST := <operand>
BYTEOFFSET_DEST := <operand>
COUNT := <operand>
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA_SRC	Input	BYTE	I、Q、M、D、L	ソースのメモリ領域で、以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB
DBNUM-BER_SRC	Input	DINT, DB_ANY	I、Q、M、D、L	AREA = DB ならばソースのメモリ領域内のデータブロックの番号、それ以外は「0」
BYTEOFF-SET_SRC	Input	DINT	I、Q、M、D、L	書き込まれるソースのメモリ領域のアドレス 16 個の最下位ビットのみが使用されます。
AREA_DEST	Input	BYTE	I、Q、M、D、L	宛先のメモリ領域で、以下の領域を選択できます。 • 16#81: 入力

				• 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB
DBNUM-BER_DEST	Input	DINT, DB_ANY	I、Q、M、D、L	AREA = DB ならば宛先のメモリ領域内のデータブロックの番号、それ以外は「0」 16 個の最下位ビットのみが使用されます。
BYTEOFF-SET_DEST	Input	DINT	I、Q、M、D、L	書き込まれる宛先のメモリ領域のアドレス 16 個の最下位ビットのみが使用されます。
COUNT	Input	DINT	I、Q、M、D、L	移動されるバイト数

有効なデータタイプの追加情報については、「関連項目」を参照してください。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

注記
入力、出力またはビットメモリ領域にメモリアドレスを書き込む場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL POKE_BLK	// 命令が呼び出されます。
number_type := DINT	// DBNUMBER パラメータのデータタイプ
AREA_SRC := "Tag_Source_Area"	// ソースメモリ領域でデータブロック領域が選択されました
DBNUMBER_SRC := "Tag_Source_DBNumber"	// ソースメモリ領域のデータブロックの番号
BYTEOFFSET_SRC := "Tag_Source_Byte"	// ソースメモリ領域の読み取り元のアドレス
AREA_DEST := "Tag_Source_Area"	// 宛先メモリ領域でデータブロック領域が選択されました
DBNUMBER_DEST := "Tag_Source_DBNumber"	// 宛先メモリ領域のデータブロックの番号
BYTEOFFSET_DEST := "Tag_Source_Byte"	// 宛先メモリ領域の読み取り元のアドレス
COUNT := "Tag_Count"	// 移動されるバイト数

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA_SRC	Tag_Source_Area	16#84
DBNUMBER_SRC	Tag_Source_DBNumber	5
BYTEOFFSET_SRC	Tag_Source_Byte	20

AREA_DEST	Tag_Destination_Area	16#83
DBNUMBER_DEST	Tag_Destination_DBNumber	0
BYTEOFFSET_DEST	Tag_Destination_Byte	30
COUNT	Tag_Count	100

この命令は、アドレス「30」から始まるビットメモリのメモリ領域内で、アドレス「20」から始まるデータブロック「5」から 100 バイトを書き込みます。

READ_LITTLE: リトルエンディアン形式のデータの読み出

説明

「リトルエンディアン形式のデータの読み出し命令を使用し、メモリ領域からデータを読み出して、これをリトルエンディアンバイトシーケンスで単一のタグに書き込みます。 リトルエンディアン形式では、最下位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_ARRAY および DEST_VARIABLE パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 DEST_VARIABLE パラメータの VARIANT は、基本データタイプであることが必要です。 SRC_ARRAY パラメータの VARIANT は読み出し元のメモリ領域を指し、これは ARRAY of BYTE であることが必要です。

POS パラメータのオペランドは、読み出しが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの読み出し

VARIANT が指すタグを読み出す場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを読み出す場合、「スライスアクセス」を使用します。

構文

「リトルエンディアン形式のデータの読み出し」命令には、以下の構文を使用します。

```
CALL READ_LITTLE
SRC_ARRAY := <operand>
OUT := <operand>
DEST_VARIABLE := <operand>
POS := <operand>
```

パラメータ

次の表に、「リトルエンディアン形式のデータの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_ARRAY	Input	ARRAY of BYTE	I、Q、M、D、L	読み出し元のメモリ領域
OUT	Output	INT	I、Q、M、D、L	エラー情報
DEST_VARIABLE	Output	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	読み出し値
POS	InOut	DINT	I、Q、M、D、L	読み出しが開始される位置を決定します。 POS パラメ

				ータは、ゼロベースで計算されます。

OUT パラメータ

次の表に、OUT パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL READ_LITTLE	// 命令が呼び出されます。
SRC_ARRAY := #SourceField	// 整数 1_295_788_826 が、メモリ領域#SourceField から読み取られます。
OUT := "Tag_Error"	// エラー情報
DEST_VARIABLE := #DINTVariable	// 整数が#DINTVariable オペランドに、リトルエンディアン形式で書き込まれます。 DEST_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。
POS := #TagPos	// 個数 4 が「#TagPos」オペランドに保存されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_ARRAY	#SourceField	BYTE の ARRAY [0..3] := 16#1A、16#2B、16#3C、16#4D
DEST_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
POS	#TagPos	0 => 4

WRITE_LITTLE: リトルエンディアン形式のデータの書き込

説明

「リトルエンディアン形式のデータの書き込み」命令を使用して、リトルエンディアンバイトシーケンスの単一のタグのデータをメモリ領域に書き込みます。 リトルエンディアン形式では、最下位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_VARIABLE および DEST_ARRAY パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 SRC_VARIABLE パラメータの VARIANT は、基本データタイプであることが必要です。 DEST_ARRAY パラメータの VARIANT が、データが書き込まれるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、書き込みが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの書き込み

VARIANT が指すタグを書き込む場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを書き込む場合、「スライスアクセス」を使用します。

構文

「リトルエンディアン形式のデータの書き込み」命令には、以下の構文を使用します。

```
CALL WRITE_LITTLE
SRC_VARIABLE := <operand>
OUT := <operand>
DEST_ARRAY := <operand>
POS := <operand>
```

パラメータ

次の表に、「リトルエンディアン形式のデータの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_VARIABLE	Input	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	書き込み元のタグ
OUT	Output	INT	I、Q、M、D、L	エラー情報
DEST_ARRAY	InOut	バイトの配列	I、Q、M、D、L	データが書き込まれるメモリ領域
POS	InOut	DINT	I、Q、M、D、L	書き込みが開始される位置を決定します。 POS パラメ

				ータは、ゼロベースで計算されます。

OUT パラメータ

次の表に、OUT パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL WRITE_LITTLE	// 命令が呼び出されます。
SRC_VARIABLE := #DINTVariable	// #DINTVariable オペランドから整数 1_295_788_826 がメモリ領域#TargetField に、リトルエンディアン形式で書き込まれます。SRC_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。
OUT := "Tag_Error"	// エラー情報
DEST_ARRAY := #TargetField	// メモリ領域
POS := #TagPos	// 個数 4 が「#TagPos」オペランドに保存されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
DEST_ARRAY	#TargetField	BYTE の ARRAY [0..10] = 16#1A、16#2B、16#3C、16#4D
POS	#TagPos	0 => 4

READ_BIG: ビッグエンディアン形式のデータの読み出し

説明

「リトルエンディアン形式のデータの読み出し命令を使用し、メモリ領域からデータを読み出して、これをビッグエンディアンバイトシーケンスで単一のタグに書き込みます。 ビッグエンディアン形式では、最上位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_ARRAY および DEST_VARIABLE パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 DEST_VARIABLE パラメータの VARIANT は、基本データタイプであることが必要です。 SRC_ARRAY パラメータの VARIANT が、データが読み出されるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、読み出しが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの読み出し

VARIANT が指すタグを読み出す場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを読み出す場合、「スライスアクセス」を使用します。

構文

「ビッグエンディアン形式のデータの読み出し」命令には、以下の構文を使用します。

```
CALL READ_BIG
SRC_ARRAY := <operand>
OUT := <operand>
DEST_VARIABLE := <operand>
POS := <operand>
```

パラメータ

次の表に、「ビッグエンディアン形式のデータの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_ARRAY	Input	バイトの配列	I、Q、M、D、L	読み出し元のメモリ領域
OUT	Output	INT	I、Q、M、D、L	エラー情報
DEST_VARIABLE	Output	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	読み出し値
POS	InOut	DINT	I、Q、M、D、L	読み出しが開始される位置を決定します。POS パラメ

				ータは、ゼロベースで計算されます。

OUT パラメータ

次の表に、OUT パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL READ_BIG	// 命令が呼び出されます。
SRC_ARRAY := #SourceField	// 整数 439_041_101 が、メモリ領域#SourceField から読み取られます。
OUT := "Tag_Error"	// エラー情報
DEST_VARIABLE := #DINTVariable	// 整数が#DINTVariable オペランドに、ビッグエンディアン形式で書き込まれます。 DEST_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。
POS := #TagPos	// 個数 4 が「#TagPos」オペランドに保存されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_ARRAY	#SourceField	BYTE の ARRAY [0..10] := 16#1A、16#2B、16#3C、16#4D
DEST_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
POS	#TagPos	0 => 4

WRITE_BIG: ビッグエンディアン形式のデータの書き込み

説明

「ビッグエンディアン形式のデータの書き込み」命令を使用して、ビッグエンディアンバイトシーケンスの単一のタグのデータをメモリ領域に書き込みます。 ビッグエンディアン形式では、最上位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_VARIABLE および DEST_ARRAY パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 SRC_VARIABLE パラメータの VARIANT は、基本データタイプであることが必要です。 DEST_ARRAY パラメータの VARIANT が、データが書き込まれるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、書き込みが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの書き込み

VARIANT が指すタグを書き込む場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを書き込む場合、「スライスアクセス」を使用します。

構文

「ビッグエンディアン形式のデータの書き込み」命令には、以下の構文を使用します。

```
CALL WRITE_BIG
SRC_VARIABLE := <operand>
OUT := <operand>
DEST_ARRAY := <operand>
POS := <operand>
```

パラメータ

次の表に、「ビッグエンディアン形式のデータの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_VARIABLE	Input	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	書き込み元のタグ
OUT	Output	INT	I、Q、M、D、L	エラー情報
DEST_ARRAY	InOut	バイトの配列	I、Q、M、D、L	データが書き込まれるメモリ領域
POS	InOut	DINT	I、Q、M、D、L	書き込みが開始される位置を決定します。POS パラメ

				ータは、ゼロベースで計算されます。

OUT パラメータ

次の表に、OUT パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL WRITE_BIG	// 命令が呼び出されます。
SRC_VARIABLE := #DINTVariable	// #DINTVariable オペランドから整数 439_041_101 がメモリ領域 #TargetField に、ビッグエンディアン形式で書き込まれます。SRC_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。
OUT := "Tag_Error"	// エラー情報
DEST_ARRAY := #TargetField	// メモリ領域
POS := #TagPos	// 個数 4 が「#TagPos」オペランドに保存されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
DEST_ARRAY	#TargetField	BYTE の ARRAY [0..10] = 16#1A、16#2B、16#3C、16#4D
POS	#TagPos	0 => 4

VARIANT

この章には下記に関する情報が記載されています：

- VariantGet: VARIANT タグ値の読み出し (S7-1500)
- VariantPut: VARIANT タグ値の書き込み (S7-1500)
- CountOfElements: 配列要素の数を取得 (S7-1500)

VariantGet: VARIANT タグ値の読み出し

説明

「VARIANT タグ値の読み出し」命令を使用して、SRC パラメータの VARIANT が指すタグの値を読み出し、その値を DST パラメータのタグに書き込むことができます。

SRC パラメータのデータタイプは、VARIANT です。 DST パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

DST パラメータで指定されたタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT 命令: ブロックムーブ」命令を使用できます。 追加情報については、「関連項目」を参照してください。

構文

VARIANT タグ値の読み出し」命令には、以下の構文を使用します。

```
CALL VariantGet
SRC  := <operand>
DST  := <operand>
```

パラメータ

次の表に、「VARIANT タグ値の読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	VARIANT	L	読み出されるタグ
DST	Output	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL VariantGet	// 命令が呼び出されます。
SRC := #TagIn_Source	// #TagIn_Source オペランドの VARIANT が指すタグの値が、読み取られます。

DST := "TagOut_Dest"　　タグの内容が、「TagOut_Dest」オペランドに書き込まれます。

VariantPut: VARIANT タグ値の書き込み

説明

「VARIANT タグ値の書き込み」命令を使用して、SRC パラメータのタグの値を VARIANT が指す DST パラメータのタグに書き込むことができます。

DST パラメータのデータタイプは、VARIANT です。 SRC パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

SRC パラメータのタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT 命令: ブロックムーブ」命令を使用できます。 追加情報については、「関連項目」を参照してください。

構文

VARIANT タグ値の書き込み」命令には、以下の構文を使用します。

```
CALL VariantPut
SRC  := <operand>
DST  := <operand>
```

パラメータ

次の表に、「VARIANT タグ値の書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	読み出されるタグ
DST	Input	VARIANT	L	命令の結果

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL VariantPut	// 命令が呼び出されます。
SRC := "TagIn_Source"	// 「TagIn_Source」オペランドの値が読み取られます...
DST := #TagIn_Dest	// #TagIn_Dest オペランドの VARIANT が指すタグに、書き込まれます。

CountOfElements: 配列要素の数を取得

説明

「ARRAY エレメント数の取得」命令を使用して、VARIANT が指しているタグが所有する ARRAY エレメント数を照会できます。

配列が一次元の ARRAY の場合、上限値と下限値の差 + 1 が結果として出力されます。 配列が多次元の ARRAY の場合、すべての次元の積が結果として出力されます。

IN パラメータのタグのデータタイプが VARIANT です。 VARIANT タグが ARRAY でない場合、結果は「0」になります。

VARIANT が ARRAY of BOOL を指す場合、FILL エレメントはカウントに含まれます。 (たとえば、ブールの配列[0..1]の場合、8 が返されます。)

構文

「ARRAY エレメント数の取得」命令には、以下の構文を使用します。

```
CALL CountOfElements
IN := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「ARRAY エレメント数の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	VARIANT	L	照会するタグ
RET_VAL	Output	UDINT	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL CountOfElements	// 命令が呼び出されます。
IN := #TagIn_Source	// #TagIn_Source オペランドの VARIANT が指すタグの配列要素の数が、読み取られます...
RET_VAL := "TagOut_RetVal"	// 「TagOut_RetVal」オペランドに出力します。

レガシー

この章には下記に関する情報が記載されています：

- BLKMOV: ブロックの移動 (S7-1500)
- UBLKMOV:割り込みなしブロック転送 (S7-1500)
- FILL:ブロックの塗りつぶし (S7-1500)

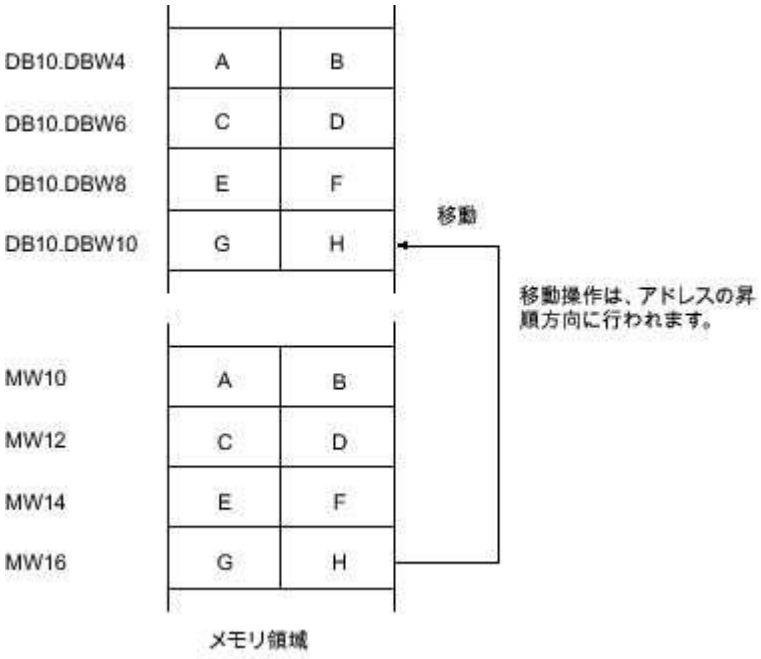
BLKMOV: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

次の図に、ムーブ操作の原理を示します。



ソース領域および宛先領域の整合性

「ブロックムーブ」命令の実行中にソースデータが変更されないことを確認してください。 変更があった場合、宛先データの整合性は保証できません。

割り込み機能

ネストレベルに対する制限はありません。

メモリ領域

「ブロックの移動」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力

- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。 ソース領域と宛先領域の長さが異なる場合、小さな領域の長さのみが移動します。

ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

文字列の移動のルール

STRING データタイプの移動元および移動先の領域の移動に「ブロックの移動」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報も、宛先領域に書き込まれます。 ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。

文字列の最大長および実際の長さに関する情報を移動するには、SRCBLK および DSTBLK パラメータにバイト単位で領域を指定します。

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
CALL BLKMOV
VARIANT
SRCBLK := <Operand>
RET_VAL := <Operand>
DSTBLK := <Operand>
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output [1)]	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみで使用できます。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL BLKMOV	// 「ブロックの移動」命令が呼び出される。
VARIANT	// 命令のデータタイプ
SRCBLK := P#M100.0 BYTE 10	// MB100 から開始して 10 バイト移動される。
RET_VAL := "Tag_ErrorCode"	// ムーブ操作中にエラーが発生した場合、変数「Tag_Error-Code」にエラーコードが出力される。
DSTBLK := P#DB1.DBX0.0 BYTE 10	// 移動されたバイトは DB1 に転送される。

UBLKMOV:割り込みなしブロック転送

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。その結果、「割り込み不可能なブロックムーブ」命令の実行中に、CPU の割り込み応答時間が長くなる場合があります。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

メモリ領域

「割り込みなしブロック転送」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

「割り込みなしブロック転送」命令の実行中は、ソース領域および宛先領域が重複してはなりません。ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

仮パラメータとして定義されたソース領域または宛先領域が、SRCBLK または DSTBLK パラメータで指定された宛先領域またはソース領域よりも小さい場合、データは転送されません。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

文字列の移動のルール

STRING データタイプのソース領域および宛先領域の移動に「割り込みなしブロック転送」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報は、宛先領域には書き込まれません。 ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。 STRING データタイプの領域を移動する場合、領域の長さとして「1」を指定します。

構文

以下の構文が「割り込みなしブロック転送」命令に使用されます。

```
CALL UBLKMOV
VARIANT
SRCBLK := <Operand>
RET_VAL := <Operand>
DSTBLK := <Operand>
```

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output [1)]	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8091	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL UBLKMOV	// 「割り込みなしブロック転送」命令が呼び出される。
VARIANT	// 命令のデータタイプ
SRCBLK := P#M100.0 BYTE 10	// MB100 から開始して 10 バイト移動される。
RET_VAL := "Tag_ErrorCode"	// ムーブ操作中にエラーが発生した場合、変数「Tag_Error-Code」にエラーコードが出力される。
DSTBLK := P#M200.0 BYTE 10	// 移動されるバイトは MB200 から開始して 10 バイトに書き込まれる

FILL:ブロックの塗りつぶし

説明

「フィル」命令を使用して、メモリ領域(宛先領域)を別のメモリ領域(ソース領域)の内容によって埋めることができます。 「フィル」命令は、宛先領域がすべて書き込まれるまでソース領域の内容を宛先領域に移動します。 ムーブ操作は、アドレスの昇順方向に行われます。

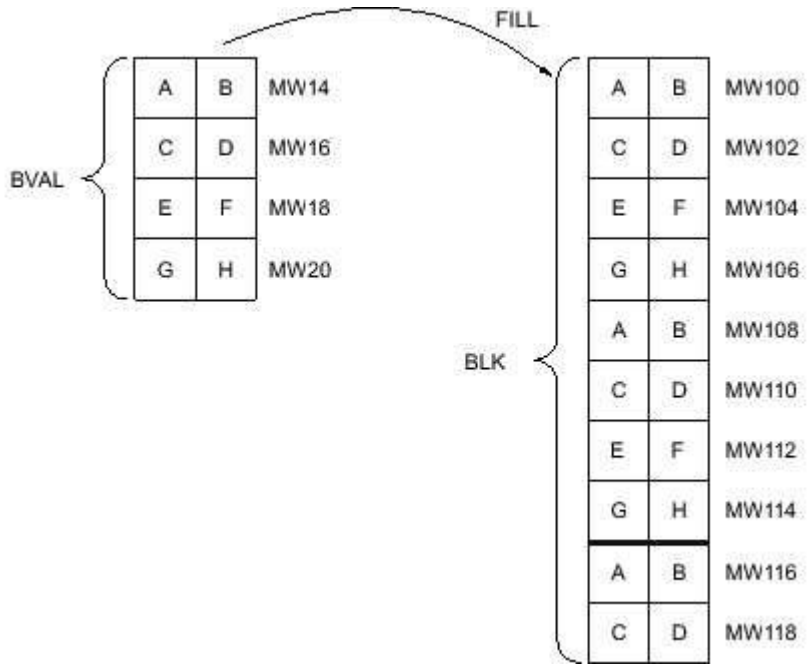
VARIANT を使用して、ソース領域と宛先領域を定義します。

注記

この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

「最適化したブロックアクセス属性を持つブロックの場合、命令「FILL_BLK: フィル」命令を使用できます。

次の図に、ムーブ操作の原理を示します。



例: MW100～MW118 の範囲の内容は、メモリワード MW14～MW20 の内容で事前割り当てされます。

ソース領域および宛先領域の整合性

「フィル」命令の実行中は、宛先データの整合性を保証するため、ソースデータは変更されないことに注意してください。

メモリ領域

「フィル」命令を使用して、以下のメモリ領域を移動することができます。

- データブロックの領域

- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。 プリセットする宛先領域が BVAL 入力パラメータの長さの整数倍ではなくても、宛先領域は最後のバイトまで書き込まれます。

プリセットする宛先領域がソース領域よりも小さい場合は、ソース領域に含まれているデータのうち、宛先領域に書き込める量のみがコピーされます。

実際に存在する宛先領域またはソース領域がソース領域または宛先領域に割り当てられたメモリ領域よりも小さい場合(BVAL、BLK パラメータ)、データは転送されません。

ANY ポインタ(ソースまたは宛先)がデータタイプ BOOL の場合、これを絶対アドレス指定する必要があり、かつ指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行されません。

宛先領域が STRING データタイプの場合、この命令は管理情報を含む文字列全体を書き込みます。

構造体移動のルール

構造体を入力パラメータとして転送するとき、構造体の長さが常に偶数バイト数になっているように注意する必要があります。 構造体を奇数バイトによって宣言する場合は、追加の 1 バイトが必要になります。

構文

「フィル」命令には、以下の構文を使用します。

```
CALL FILL
VARIANT
BVAL := <Operand>
RET_VAL := <Operand>
BLK := <Operand>
```

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
BVAL	Input	VARIANT	I、Q、M、L、P	その内容が BLK パラメータの宛先領域を埋めるために使用されるメモリ領域(ソース領域)の指定。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
BLK	Output [1)]	VARIANT	I、Q、M、L、P	ソース領域の内容で埋められるメモリ領域の指定。

1) データがタグに流れるため、BLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは BVAL パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは BLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL FILL	// 「フィル」命令が呼び出される。
VARIANT	// 命令のデータタイプ
BVAL := P#M14.0 WORD 4	// 領域が MW14 から MW20 に移動される
RET_VAL := "Tag_ErrorCode"	// エラー情報
BLK := P#M100.0 WORD 10	// メモリ領域 MW100 ~ MW118 が BVAL パラメータのメモリ領域に含まれる 4 ワードの内容によって埋められる

変換操作

この章には下記に関する情報が記載されています：

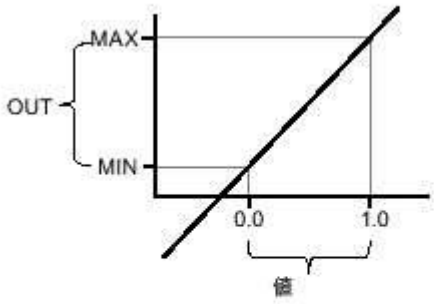
- SCALE_X: スケール (S7-1500)
- NORM_X: 正規化 (S7-1500)
- VARIANT (S7-1500)
- レガシー (S7-1500)

SCALE_X: スケール

説明

「スケール」命令を使用して、VALUE 入力の値を指定した値の範囲にマッピングしてスケーリングすることができます。 「スケール」命令が実行されると、VALUE 入力の浮動小数点値が、MIN および MAX パラメータによって定義された値の範囲にスケーリングされます。 スケーリングの結果は整数になり、RET_VAL 出力に保存されます。

次の図に、値をどのようにスケーリングできるかを示します。



「スケール」命令は、以下の等式で機能します。

OUT = [VALUE * (MAX – MIN)] + MIN

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

構文

「スケール」命令には、以下の構文を使用します。

```
CALL SCALE_X
??? ???
MIN := <operand>
VALUE := <operand>
MAX := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MIN	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE	Input	浮動小数点数	I、Q、M、D、L、または定数	スケーリングされる値 定数を入力する場合は、それを宣言する必要があります。
MAX	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の上限値
RET_VAL	Output	整数、浮動小数点数	I、Q、M、D、L	スケーリングの結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL SCALE_X	// 命令が呼び出されます。
dest_type := INT	// 命令のデータタイプ
src_type := REAL	// VALUE パラメータのデータタイプ
MIN := "Tag_Minimum"	// 下限値
VALUE := "Tag_Value"	// 浮動小数点値
MAX := "Tag_Maximum"	// 上限値
RET_VAL := "Tag_Result"	// 命令の結果

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_Minimum	10
VALUE	Tag_Value	0.5
MAX	Tag_Maximum	30
OUT	Tag_Result	20

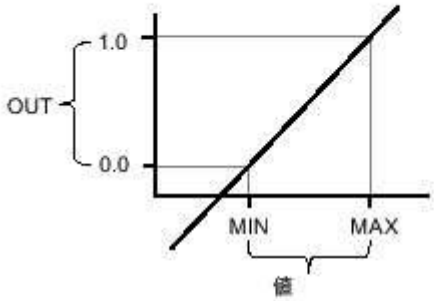
「Tag_Value」入力の値が、「Tag_Minimum」および「Tag_Maximum」入力の値によって定義された値の範囲にスケーリングされます。 結果が「Tag_Result」出力に保存されます。

NORM_X: 正規化

説明

「正規化」命令を使用して、VALUE 入力のタグの値をリニアスケーリングにマッピングして正規化することができます。 MIN および MAX パラメータを使用し、スケーリングに適用する値の範囲の限界を定義できます。 RET_VAL 出力の結果が計算され、正規化される値のこの値の範囲内の位置に応じて、浮動小数点数として保存されます。 正規化される値が MIN 入力の値に等しい場合、OUT 出力の値は「0.0」になります。 正規化する値が入力 MAX の値と等しい場合、出力 OUT が値「1.0」を返します。

次の図に、値が正規化される例を示します。



「正規化」命令は、以下の等式で機能します。

OUT = (VALUE – MIN) / (MAX – MIN)

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

構文

「正規化」命令には、以下の構文を使用します。

```
CALL NORM_X
??? ???
MIN := <operand>
VALUE := <operand>
MAX := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「正規化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MIN 1)	Input	整数、浮動小数点数	I、Q、M、D、L	値の範囲の下限値
VALUE 1)	Input	整数、浮動小数点数	I、Q、M、D、L	正規化する値
MAX 1)	Input	整数、浮動小数点数	I、Q、M、D、L	値の範囲の上限値
RET_VAL	Output	浮動小数点数	I、Q、M、D、L	正規化の結果
1) これらの 3 つのパラメータで定数を使用する場合は、そのいずれかを宣言するのみで済みます。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL NORM_X	// 命令が呼び出されます。
src_type := INT	// 命令のデータタイプ
dest_type := REAL	// RET_VAL パラメータのデータタイプ
MIN := "Tag_Minimum"	// 下限値
VALUE := "Tag_Value"	// 浮動小数点値
MAX := "Tag_Maximum"	// 上限値
RET_VAL := "Tag_Result"	// 命令の結果

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_Minimum	10
VALUE	Tag_Value	20
MAX	Tag_Maximum	30
RET_VAL	Tag_Result	0.5

「Tag_Value」入力の値が、「Tag_Minimum」および「Tag_Maximum」入力の値によって定義された値の範囲にマッピングされます。 「Tag_Value」入力のタグ値が、定義された値の範囲に正規化されます。 結果が、浮動小数点数として「Tag_Result」出力に保存されます。

VARIANT

この章には下記に関する情報が記載されています：

- VARIANT_TO_DB_ANY: VARIANT の DB_ANY への変換 (S7-1500)
- DB_ANY_TO_VARIANT: DB_ANY の VARIANT への変換 (S7-1500)

VARIANT_TO_DB_ANY: VARIANT の DB_ANY への変換

説明

「VARIANT の DB_ANY への変換」命令を使用して、IN パラメータがアドレス指定するオペランドのデータブロック番号を照会します。 インスタンスデータブロックまたは配列データブロックにすることはできません。 IN パラメータのオペランドはデータタイプ VARIANT です。つまり、プログラムの作成時に照会する番号のデータブロックのデータタイプが既知である必要があります。 データブロック番号はランタイム中に読み取られ、RET_VAL パラメータで指定されたオペランドに書き込まれます。

必要条件

条件が満たされる場合は、命令が実行されます。 条件が満たされない場合は、データブロック番号として「0」が出力されます。

出力タグ...	原点復帰...	変換オプション
VARIANT	... データブロックであり、PLC データタイプまたはシステムデータタイプ(SDT)のインスタンスデータブロックです。	出力タグをデータブロック番号に変換できます。
VARIANT	... 配列 DB であるデータブロック。	出力タグをデータブロック番号に変換できます。
VARIANT	... 基本データタイプのタグ。	データブロックを 1 つの基本データタイプのみで構成することはできないため、出力タグをデータベース番号に変換できません。
VARIANT	... データブロック内の構造体。	これはデータブロック内の一部にすぎないため、出力タグをデータベース番号に変換できません。

構文

「VARIANT の DB_ANY への変換」命令には、以下の構文を使用します。

```
CALL VARIANT_TO_DB_ANY
IN := <operand>
RET_VAL := <operand>
ERR := <operand>
```

パラメータ

次の表に、「VARIANT の DB_ANY への変換」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	VARIANT	I、Q、M、L	読み出されるタグ
RET_VAL	Output	DB_ANY	I、Q、M、D、L	結果: DB の番号
ERR	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR パラメータ

次の表に、ERR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
252C	IN パラメータの VARIANT データタイプの値は「0」であり、CPU が STOP モードに切り替わります。
80B4	配列データブロックに保存された工レメントのデータタイプが、VARIANT で転送される工レメントのデータタイプと一致しません。
8131	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。
8132	データブロックが小さすぎるか、または配列データブロックではありません。
8150	IN パラメータのデータタイプ VARIANT が、値「0」を提供します。 このエラーメッセージを受信するには、「ブロック内でのエラー処理」ブロックプロパティが有効であることが必要です。 有効でない場合、CPU が STOP モードに切り替わり、エラーコード 16#252C を送信します。
8153	IN パラメータの VARIANT データタイプが配列データブロックの開始を指していないか、または VARIANT の長さがデータブロックの長さと一致しません。
8154	データブロックのデータタイプが不正です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL VARIANT_TO_DB_ANY	// 命令が呼び出されます。
IN := #InputTag	// #InputTag オペランドで指定されたデータブロックの番号が読み取られます。 オペランドのデータタイプが VARIANT であるため、プログラムの作成中にタグのデータタイプを知っている必要はありません。
RET_VAL := "OutputDBNumber"	// この番号が、データタイプが DB_ANY である「OutputDBNumber」タグに書き込まれます。
ERR := "Tag_Error"	// エラー情報

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
IN	Input	#InputTag	-
RET_VAL	Output	OutputDBNumber	11

DB_ANY_TO_VARIANT: DB_ANY の VARIANT への変換 

説明

「DB_ANY の VARIANT への変換」命令を使用して、以下にリストした要件を満たすデータブロックから VARIANT タグを生成します。 IN パラメータのオペランドはデータタイプ DB_ANY です。つまり、プログラムの作成時にデータブロックが既知である必要はありません。 データブロックが番号がランタイム中に読み取られます。

必要条件

条件が満たされる場合は、命令が実行されます。 要件が満たされない場合またはデータブロックが存在しない場合、RET_VAL パラメータで値 NULL が出力されます。 RET_VAL タグによるその他すべてのアクセスは失敗します。

データタイプの入力タグ...	原点復帰...	変換オプション
DB_ANY	...データブロックであり、PLC データタイプまたはシステムデータタイプ(SDT)のインスタンスデータブロック。	変換が可能です。
DB_ANY	...配列 DB であるデータブロック。	変換が可能です。
DB_ANY	...ファンクションブロックのインスタンスデータブロックまたはグローバルデータブロックであるデータブロック。	変換が不可能です。

構文

「DB_ANY の VARIANT への変換」命令には、以下の構文を使用します。

```
CALL DB_ANY_TO_VARIANT
IN := <operand>
RET_VAL := <operand>
ERR := <operand>
```

パラメータ

次の表に、「DB_ANY の VARIANT への変換」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	DB_ANY	I、Q、M、D、L	番号が読み出されるデータブロック
RET_VAL	Output [1)]	VARIANT	L	データブロックの番号
ERR	Output	INT	I、Q、M、D、L	エラー情報

1) データがタグに流れるため、RET_VAL パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR パラメータ

次の表に、ERR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8130	データブロックの番号が「0」です
8131	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。
8132	データブロックが小さすぎるか、または配列データブロックではありません。
8134	データブロックが書き込み保護されています。
8154	データブロックのデータタイプが不正です。
8155	データブロックのデータタイプが不明です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL DB_ANY_TO_VARIANT	// 命令が呼び出されます。
IN := "InputDB"	//「InputDB」オペランドで指定された任意のデータブロックの番号が、そのデータブロックをアドレス指定するデータタイプ VARIANT のタグの生成に使用されます。 IN パラメータのオペランドのデータタイプは DB_ANY であるため、ランタイム中に使用されるデータブロックはプログラムの作成時に既知である必要はありません(データブロックの名前も番号も不明でかまいません)。
RET_VAL := #tempVARIANT	// RET_VAL パラメータのオペランドのデータタイプは VARIANT であるため、プログラムの作成時にデータブロックのデータタイプが既知である必要はありません。
ERR := "Tag_Error"	// エラー情報

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
IN	Input	InputDB	11

RET_VAL	Temp	#tempVARIANT	-

レガシー

この章には下記に関する情報が記載されています：

- SCALE: スケール (S7-1500)
- UNSCALE:スケール解除 (S7-1500)

SCALE: スケール

説明

「スケール」命令を使用して、IN パラメータの整数を下限値と上限値間の物理単位でスケーリングが可能な浮動小数点数に変換することができます。 LO_LIM および HI_LIM パラメータを使用して、入力値をスケーリングする範囲の下限値と上限値を指定することができます。 命令の結果は、OUT パラメータに出力されます。

「スケール」命令は、以下の等式で機能します。

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は-27648.0 となり、定数「K2」の値は+27648.0 となります。
- シグナル状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は 0.0 となり、定数「K2」の値は+27648.0 となります。

IN パラメータの値が定数値「K2」よりも大きい場合、命令の結果が上限値(HI_LIM)にセットされ、エラーが出力されます。

IN パラメータの値が定数値「K1」未満の場合、命令の結果が下限値(LO_LIM)の値にセットされ、エラーが出力されます。

指示された下限値が上限値よりも大きい(LO_LIM >HI_LIM)場合、結果は入力値に反比例してスケーリングされます。

構文

「スケール」命令には、以下の構文を使用します。

```
CALL SCALE
IN := <operand>
HI_LIM := <operand>
LO_LIM := <operand>
BIPOLAR := <operand>
RET_VAL := <operand>
OUT := <operand>
```

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	INT	I、Q、M、D、L、P、または定数	スケーリングする入力値

HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
RET_VAL	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT	Output	REAL	I、Q、M、D、L、P	命令の結果

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	**説明**
0000	エラーは発生していません。
0008	IN パラメータの値が定数値「K2」を超えているか、または定数値「K1」未満になっています。
一般エラー情報	関連項目:"GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	**説明**
CALL SCALE	// 命令が呼び出されます。
IN := "Tag_InputValue"	// 変換およびスケーリングされる値を指定する
HI_LIM := "Tag_HighLimit"	// 上限値
LO_LIM := "Tag_LowLimit"	// 下限値
BIPOLAR := "Tag_Bipolar"	// IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示す。
RET_VAL := "Tag_ErrorCode"	// エラー情報
OUT := "Tag_OutputValue"	// 命令の結果

次の表に、実行前の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

次の表に、実行後の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

UNSCALE:スケール解除

説明

「スケール解除」命令は、IN パラメータの浮動小数点数を下限値と上限値間の物理単位にスケーリング解除し、それを整数に変換します。 LO_LIM および HI_LIM パラメータを使用して、入力値をスケーリング解除する範囲の下限値と上限値を指定することができます。 命令の結果は、OUT パラメータに出力されます。

「スケール解除」命令は、以下の等式で機能します。

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は-27648.0 となり、定数「K2」の値は+27648.0 となります。
- シグナル状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は 0.0 となり、定数「K2」の値は+27648.0 となります。

IN パラメータの値が定数値「HI_LIM」よりも大きい場合、この命令の結果が定数値(K2)にセットされ、エラーが出力されます。

IN パラメータの値が下限値の定数値「LO_LIM」未満の場合、この命令の結果が定数値(K1)にセットされ、エラーが出力されます。

構文

以下の構文が「スケール解除」命令に使用されます。

```
CALL UNSCALE
IN := <operand>
HI_LIM := <operand>
LO_LIM := <operand>
BIPOLAR := <operand>
RET_VAL := <operand>
OUT := <operand>
```

パラメータ

次の表に、「スケール解除」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	REAL	I、Q、M、D、L、P、または定数	整数値にスケーリング解除する入力値
HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	上限値

LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
RET_VAL	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT	Output	INT	I、Q、M、D、L、P	命令の結果

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	**説明**
0000	エラーは発生していません。
0008	IN パラメータの値が上限値(HI_LIM)を超えているか、または下限値(LO_LIM)未満になっています。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	**説明**
CALL UNSCALE	// 命令が呼び出されます。
IN := "Tag_InputValue"	// スケーリング解除する値を指定する
HI_LIM := "Tag_HighLimit"	// 上限値
LO_LIM := "Tag_LowLimit"	// 下限値
BIPOLAR := "Tag_Bipolar"	// IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示す。
RET_VAL := "Tag_ErrorCode"	// エラー情報
OUT := "Tag_OutputValue"	// 命令の結果

次の表に、実行前の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

次の表に、実行後の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

プログラム制御演算

この章には下記に関する情報が記載されています：

- ランタイム制御 (S7-1500)

ランタイム制御

この章には下記に関する情報が記載されています：

- ENDIS_PW: パスワードの適正化の制限および有効化 (S7-1500)
- RE_TRIGR: サイクルタイムモニタのリスタート (S7-1500)
- STP: プログラム終了 (S7-1500)
- GET_ERROR: ローカルでエラーを取得 (S7-1500)
- GET_ERR_ID: ローカルでエラー ID を取得 (S7-1500)
- INIT_RD: すべての保持データの初期化 (S7-1500)
- WAIT: 遅延時間の設定 (S7-1500)
- RUNTIME: プログラムランタイムの測定 (S7-1500)

ENDIS_PW: パスワードの適正化の制限および有効化

説明

「パスワード正当性の制限および有効化」命令を使用して、設定されたパスワードを CPU に対して正当化できるかどうかを指定します。 このため、正しいパスワードがわかっている場合でも、正規の接続を防止することができます。

命令呼び出し時に REQ パラメータのシグナル状態が「0」の場合、現在設定されている状態が、出力パラメータに表示されます。 入力パラメータに変更を加えても、変更は出力パラメータに転送されません。

命令呼び出し時に REQ パラメータのシグナル状態が「1」の場合、シグナル状態は、入力パラメータ(F_PWD、FULL_PWD、R_PWD、HMI_PWD)から取得されます。FALSE は、パスワードごとの正当化が許可されていないことを意味します。TRUE は、パスワードを使用できることを意味します。

パスワードの有効化または無効化を個別に許可または禁止することができます。 たとえば、フェールセーフパスワードを除く、すべてのパスワードを禁止することができます。 したがって、アクセスオプションを小さいユーザーグループに制限することができます。 出力パラメータ(F_PWD_ON、FULL_PWD_ON、R_PWD_ON、HMI_PWD_ON)は、REQ パラメータに関係なく、常にパスワード使用の現在のステータスを示します。

設定済みでないパスワードの場合、入力のシグナル状態が TRUE でなければならず、出力にシグナル状態 TRUE を返します。 フェールセーフパスワードは、F-CPU の場合のみ設定できるため、標準 CPU では、常にシグナル状態 TRUE と相互接続する必要があります。 命令がエラーを返す場合、この呼び出しには効果がなく、直前のロックがまだ有効です。

無効なパスワードは、以下の条件で再度有効にすることができます。

- CPU が工場出荷時設定にリセットされること。
- S7-1500 CPU のフロントパネルが、パスワードを再度有効にすることができる適切なメニューに移動することができるダイアログをサポートしていること。
- 「パスワードの適正化の制限および有効化」命令を呼び出す場合、目的のパスワードの入力パラメータのシグナル状態が「1」であること。
- モードセレクタを STOP に設定すること。 このスイッチの設定を RUN に戻すと直ちに、パスワードの適正化に対する制限が再確立されます。
- S7-1200 CPU への空きメモリカード(転送モジュールまたはプログラムカード)の差し込み。
- S7-1200 CPU では、電源オフから電源オンへの移行によって、保護が無効になります。 この場合、プログラム(たとえば、スタートアップ OB)で、再度、「パスワード正当性の制限および有効化」命令を呼び出す必要があります。

注記
「パスワード正当性の制限および有効化」命令は、HMI パスワードが有効にされていない場合、HMI システムからのアクセスをブロックします。

注記
既存の適正化された接続はアクセス権を保持し、「パスワードの適正化の制限および有効化」命令を使用してそれらを制限することはできません。

S7-1500 CPU の偶発的なロックアウトの防止

CPU のフロントパネルで設定を行うことができ、CPU は最新の設定を保存します。

偶発的なロックアウトを防止するには、S7-1500 CPU でモードセレクタを STOP に設定することによって保護を無効にすることができます。 モードセレクタを RUN に設定することによって、「パスワ

ードの適正化の制限および有効化」命令を再度呼び出したり、フロントパネルで追加の操作を実行したりせずに、保護が自動的に再度有効になります。

S7-1200 CPU の偶発的なロックアウトの防止

S7-1200 CPU にはモードスイッチが存在しないため、電源オフから電源オンへの移行時の保護は無効です。 このため、ユーザープログラム内の特定のプログラムシーケンスによって偶発的なロックアウトを防止することをお奨めします。

これを行うには、サイクリック割り込み OB、またはメイン OB (OB 1)のタイマのいずれかを使用して、時間制御をプログラミングします。 これによって、電源オフから電源オンへの移行と、関連する保護の無効化の後、比較的迅速に、各 OB (たとえば、OB 1 または OB 35)内で「パスワード正当性の制限および有効化」命令を呼び出すオプションが可能になります。 スタートアップ OB (OB 100)でこの命令を呼び出して、この命令が無効で、パスワード正当化の制限が存在しない時間帯を比較的小さなものに保ちます。 この手順は、未許可のアクセスに対する可能な最大限の保護を提供します。

偶発的なロックアウトが発生した場合は、スタートアップ OB での呼び出しをスキップし(たとえば、入力パラメータを照会することによって)、ロックが再び有効になる前に、CPU へ接続確立する時間(たとえば、10 秒～1 分)を設定することができます。

ユーザープログラムコードにタイマが存在しない場合にロックアウトが発生した場合は、空の転送カードまたはプログラムカードを CPU に挿入します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 この後、ユーザープログラムを再び STEP 7 から CPU にダウンロードする必要があります。

S7-1200 CPU でパスワードが失われた場合の手順

パスワード保護された S7-1200 CPU のパスワードを紛失した場合は、空の転送カードまたはプログラムカードを使用して、パスワード保護されたプログラムを削除します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 その後、新しいユーザープログラムを STEP 7 Basic から CPU にロードすることができます。

警告

空の転送カードの挿入

ランタイム中に CPU に転送カードを挿入すると、CPU は STOP モードに切り替わります。 動作状態が不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期しない動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

転送カードを取り出すと、転送カードの内容が内部ロードメモリで使用可能になります。 この時点に、カードにプログラムが含まれていないことを確認してください。

警告

空のプログラムカードの挿入

ランタイム中に CPU にプログラムカードを挿入すると、CPU は STOP モードに移行します。 動作状態が不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期しない動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

プログラムカードが空であることを確認してください。 内部ロードメモリが空のプログラムカードにコピーされます。 直前に空であったプログラムカードを取り出すと、内部ロードメモリは空になります。

CPU を RUN モードに切り替える前に、転送カードまたはプログラムカードを取り出す必要があります。

動作モードへのパスワードの使用の影響

次の表に、「パスワードの適正化の制限および有効化」命令によるパスワードの使用が動作モードと対応するユーザーの操作に与える影響を示します。

操作	命令によるパスワードの保護
その後の基本状態 • STOP への動作モードの切り替え • メモリの手動リセット(PG、切り替え、MC (モーションコントロール)の変更) • 工場出荷時設定へのリセット	無効 (制限なし)
電源オン後の基本状態	• S7-1200-CPU: ロックは無効で、プログラム(たとえば、スタートアップ OB)で再びこの命令を呼び出す必要があります。 • S7-1500 CPU: 有効(電源オフの前にロックが有効になった場合). パスワードの不許可のオプションは保持型です。
動作モードの移行 RUN/STARTUP/HOLD -> STOP (命令、エラーまたは通信の終了によって)または STOP -> STARTUP/RUN/HOLD	有効 パスワードはまだ使用できません。

構文

「パスワード正当性の制限および有効化命令には、以下の構文を使用します。

```
CALL ENDIS_PW
REQ := <operand>
F_PWD := <operand>
FULL_PWD := <operand>
R_PWD := <operand>
HMI_PWD := <operand>
F_PWD_ON := <operand>
FULL_PWD_ON := <operand>
R_PWD_ON := <operand>
HMI_PWD_ON := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「パスワードの適正化の制限および有効化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ パラメータのシグナル状態が「0」の場合は、現在設定されているパスワード

				のシグナル状態が照会されます。
F_PWD	Input	BOOL	I、Q、M、D、L、または定数	フェールセーフなどの読み取り/書き込みアクセス • F_PWD = "0": パスワードを許可しない • F_PWD = "1": パスワードを許可する
FULL_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取り/書き込みアクセス • FULL_PWD = "0": パスワードを許可しない • FULL_PWD = "1": パスワードを許可する
R_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取りアクセス • R_PWD = "0": パスワードを許可しない • R_PWD = "1": パスワードを許可する
HMI_PWD	Input	BOOL	I、Q、M、D、L、または定数	HMI アクセス • HMI_PWD = "0": パスワードを許可しない • HMI_PWD = "1": パスワードを許可する
F_PWD_ON	Output	BOOL	I、Q、M、D、L	フェールセーフなどの読み取り/書き込みアクセス • F_PWD_ON = "0": パスワードを許可しない • F_PWD_ON = "1": パスワードを許可する
FULL_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取り/書き込みアクセスステータス • FULL_PWD_ON = "0": パスワードを許可しない • FULL_PWD_ON = "1": パスワードを許可する
R_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取りアクセスステータス • R_PWD_ON = "0": パスワードを許可しない • R_PWD_ON = "1": パスワードを許可する
HMI_PWD_ON	Output	BOOL	I、Q、M、D、L	HMI アクセスステータス • HMI_PWD_ON = "0": パスワードを許可しない • HMI_PWD_ON = "1": パスワードを許可する
RET_VAL	Output	WORD	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8090	「パスワードの適正化の制限および有効化」命令はサポートされていません
80D0	フェールセーフのパスワードが未設定です。 標準 CPU では、シグナル状態は TRUE であることが必要です。
80D1	読み取り/書き込みアクセスが未設定です
80D2	読み取りアクセスが未設定です
80D3	HMI アクセスが未設定です
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

RE_TRIGR: サイクルタイムモニタのリスタート

説明

「サイクルタイムモニタのリスタート」命令を使用して、CPU のサイクルタイムモニタを再起動することができます。 サイクルタイムモニタが、CPU の設定で設定した時間で再起動されます。

「サイクルタイムモニタのリスタート」命令は、すべてのブロック内で優先度に関わらず呼び出し可能です。

ハードウェア割り込み、診断割り込み、または周期割り込みなどの上位の優先度のブロック内でこの命令が呼び出される場合、この命令は実行されません。

「サイクルタイムモニタのリスタート」命令は、呼び出しの数に関係なく期間内に完全に実行されます (最大プログラムサイクルの 10 倍)。 時間が期限切れになると、プログラムサイクルは延長できなくなります。

構文

「サイクルタイムモニタのリスタート」命令には、以下の構文を使用します。

```
CALL RE_TRIGR
```

パラメータ

「サイクルタイムモニタのリスタート」命令にはパラメータがなく、エラー情報は返されません。

STP: プログラム終了

説明

「プログラムの終了」命令を使用して、CPU を STOP モードに設定し、その結果、プログラムの実行を終了することができます。 RUN から STOP への変更の影響は CPU の構成によって異なります。

構文

以下の構文が「プログラムの終了」命令に使用されます。

```
CALL STP
```

パラメータ

「プログラムの終了」命令にはパラメータがなく、エラー情報は返されません。

GET_ERROR: ローカルでエラーを取得

説明

「ローカルでエラーを取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 システムがブロックの処理中にエラーを報告した場合、命令の最後の実行の後、このエラーがブロックの実行中に発生した最初の該当エラーの場合、詳細情報が、OUT 出力のオペランドに格納されます。

OUT 出力では、「ErrorStruct」システムデータタイプのオペランドのみを指定できます。 「ErrorStruct」システムデータタイプは、エラー情報が保存されている正確な構造体を指定します。 追加の命令を使用することで、この構造体とプログラムが適切な応答をするかを評価できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーに関するエラー情報が出力されます。

注記

OUT 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記

「ローカルでエラーを取得」命令は、ブロック内のローカルエラー処理を可能にします。 「ローカルでエラーを取得」がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

構文

「ローカルでエラーを取得」命令には、以下の構文を使用します。

```
CALL GET_ERROR
OUT := <operand>
```

パラメータ

次の表に、「ローカルでエラーを取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OUT	Output	ErrorStruct	D、L	エラー情報

データタイプ「ErrorStruct」

次の表に、「ErrorStruct」データタイプの構造体を示します。

構造体コンポーネント		データタイプ	説明
ERROR_ID		WORD	エラー ID
FLAGS		BYTE	エラーがブロック呼び出し中に発生したのかどうかを示します。 16#01: ブロック呼び出し中のエラー 16#00: ブロック呼び出し中にエラーなし
REACTION		BYTE	既定の応答 0: 無視(書き込みエラー) 1: 代替値「0」で続行(読み出しエラー) 2: 命令をスキップ(システムエラー)
CODE_ADDRESS		CREF	ブロックのアドレスおよびタイプに関する情報
	BLOCK_TYPE	BYTE	エラーが発生したブロックのタイプ: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	プログラムブロックの番号
	OFFSET	UDINT	内部メモリへの参照
MODE		BYTE	オペランドのアドレスに関する情報
OPERAND_NUMBER		UINT	マシンコマンドのオペランド番号
POINTER_NUMBER_LOCATION		UINT	(A) 内部ポインタ
SLOT_NUMBER_SCOPE		UINT	(B) 内部メモリの記憶領域
DATA_ADDRESS		NREF	オペランドのアドレスに関する情報
	AREA	BYTE	(C) メモリ領域 L: 16#40～4E、86、87、8E、8F、C0～CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84、85、8A、8B データタイプ DINT の直接編集可能なタグの範囲違反: 16#04

	DB_NUMBER	UINT	(D) データブロックの番号
	OFFSET	UDINT	(E) オペランドの相対アドレス

構造体コンポーネント「ERROR_ID」

次の表に、構造体コンポーネント「ERROR_ID」で出力可能な値を示します。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、またはファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、またはファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー: 入力
2943	10563	書き込みエラー: 出力

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

	説明
LABEL: L #Field[#index]	// #Field[#index]タグへのアクセス
L 40.5	// 値 40.5 の読み込み
*R	// タグの値と値 40.5 の乗算を行う。
T#TagOut	// 積をオペランド#TagOut に転送。
CALL GET_ERROR	// 命令が呼び出されます。
OUT := #Error	// エラーが発生すると、詳細情報が#Error オペランドに格納される。
T#Error.REACTION	// 構造体コンポーネント#Error.REACTION が読み出される。
L 1	// 値 1 の読み込み
==I	// 構造体コンポーネントの値が値 1 と比較される。
JC LABEL	// 2 つの値の比較が正の場合、プログラム実行はジャンプラベル LABEL に戻ります。

#Field[#index]タグへのアクセスでエラーが発生しました。 #TagOut オペランドは、読み取り/アクセスエラーに関わらずシグナル状態「1」を返し、乗算は値「0.0」によって実行されます。 I このエラーシナリオが発生する場合は、エラーを取得するために、乗算の後に、「ローカルでエラーを取得」命令をプログラムすることを推奨します。 「ローカルでエラーを取得」命令によって供給されるエラー情報は、比較を使用して評価されます。 #Error.REACTION 構造コンポーネントの値が「1」の場合、読み取り/アクセスエラーが発生しており、プログラム実行はジャンプラベル LABEL で再び開始します。

GET_ERR_ID: ローカルでエラー ID を取得

説明

「ローカルでエラー ID を取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 この命令の最後の実行の後に、システムがブロック処理中にブロック実行に関するエラーを報告すると、タグで発生した最初のエラーのエラー ID が RET_VAL 出力に格納されます。

RET_VAL 出力では、WORD データタイプのオペランドのみを指定できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーのエラー ID が出力されます。

注記
RET_VAL 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。

「ローカルでエラー ID を取得」命令の出力は、エラー情報が存在する場合のみ設定されます。 これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラー ID を取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラー ID を取得」命令は、ブロック内のローカルエラー処理を可能にします。「ローカルでエラー ID を取得」命令がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

構文

「ローカルでエラー ID を取得」命令には、以下の構文を使用します。

```
CALL GET_ERR_ID
RET_VAL := <operand>
```

パラメータ

次の表に、「ローカルでエラー ID を取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Output	WORD	I、Q、M、D、L	エラー ID

RET_VAL パラメータ

次の表に、RET_VAL パラメータで出力できる値を示します。

RET_VAL* (16 進数)	RET_VAL* (10 進数)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、またはファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、またはファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー: 入力
2943	10563	書き込みエラー: 出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

例

次の例で、命令がどのように動作するかを示します。

	説明
L#Field[#index]	// #Field[#index]タグへのアクセス

```
L    40.5                    // 値 40.5 の読み込み
*R                           // タグの値と値 40.5 の乗算を行う。
T#TagOut                     // 積をオペランド#TagOut に転送。

CALL GET_ERR_ID              // 命令が呼び出されます。
RET_VAL := #TagID            // エラーが発生すると、エラー ID が#TagID オペランドに格納される。

T#TagID                      // #TagID オペランドのエラー ID が読み出される。
L 16#2522                    // 値 16#2522 の読み込み
==I                          // エラー ID の値が値 16#2522 と比較される。
CALL MOVE                    // 命令が呼び出されます。
VARIANT                      // 命令のデータタイプ
IN := 100.0                  // 値 100.0 が#TagOut オペランドにコピーされる。
OUT := #TagOut               // 命令の出力
```

#Field[#index]タグへのアクセスでエラーが発生しました。 #TagOut オペランドは、読み取り/アクセスエラーに関わらずシグナル状態「1」を返し、乗算は値「0.0」によって実行されます。 I このエラーシナリオが発生する場合は、エラーを取得するために、乗算の後に、「ローカルでエラー ID を取得」命令をプログラムすることを推奨します。 「ローカルでエラー ID を取得」命令によって供給されるエラー ID は、比較を使用して評価されます。 #TagID オペランドが ID 16#2522 を返す場合、読み取り/アクセスエラーが発生しており、値「100.0」が#TagOut 出力に移動します。

INIT_RD: すべての保持データの初期化

説明

「すべての保持データを初期化」命令を使用すると、すべてのデータブロック、ビットメモリ、および SIMATIC タイマとカウンタの保持データが同時にリセットされます。 この命令は、実行するとプログラムサイクル持続時間を超過するため、スタートアップ OB 内でのみ実行が可能です。

構文

「すべての保持データを初期化」命令には、以下の構文を使用します。

```
CALL INIT_RD
REQ := <operand>
RET_VAL := <operand>
```

パラメータ

次の表に、「すべての保持データを初期化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	入力「REQ」のシグナル状態が「1」の場合、すべての保持データがリセットされます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B5	この命令はスタートアップ OB 内にプログラムされていないため、実行できません。
一般エラー情報	関連項目:"GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL INIT_RD	// 命令が呼び出されます。
REQ := "Tag_REQ"	オペランド「Tag_REQ」のシグナル状態が「1」の場合、すべての保持データがリセットされます。
RET_VAL := "Tag_RET_VAL"	// エラーコード

「Tag_REQ」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 すべてのデータブロック、ビットメモリ、および SIMATIC タイマの保持データとカウンタがリセットされます。

WAIT: 遅延時間の設定

説明

「遅延時間の設定」命令は、指定された時間の間プログラム実行を一時停止します。 この命令の WT パラメータにマイクロ秒単位で時間を指定します。

遅延時間は、-32768 から最大 32767 マイクロ秒 (µs)が設定可能です。設定可能な最短の遅延時間は個々の CPU に依存し、「遅延時間の設定」命令の実行時間に対応します。

この命令の実行には、さらに優先度の高いイベントが割り込む可能性があります。

「遅延時間の設定」命令はエラー情報を返しません。

構文

以下の構文が「遅延時間の設定」命令に使用されます。

```
CALL WAIT
WT := <operand>
```

パラメータ

次の表に、「遅延時間の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
WT	Input	INT	I、Q、M、D、L、P、または定数	遅延時間(µs)

RUNTIME: プログラムランタイムの測定

説明

「プログラムランタイムの測定」命令を使用して、プログラム全体、個々のブロック、またはコマンドシーケンスのランタイムを測定します。

プログラム全体のランタイムを測定する場合は、「プログラムランタイムの測定」命令を OB1 で呼び出します。 最初の呼び出しでランタイムの測定が開始され、2 回目の呼び出し後に出力 RET_VAL がプログラムのランタイムを返します。 測定されたランタイムには、たとえば上位レベルのイベントや通信による割り込みなど、プログラム実行中に発生する可能性のあるすべての CPU プロセスが含まれます。 「プログラムランタイムの測定」命令は CPU の内部カウンタを読み取り、値を IN-OUT パラメータ MEM に書き込みます。 この命令は、内部カウンタ周波数に基づいて現在のプログラムランタイムを計算し、出力 RET_VAL に書き込みます。

個々のブロック、または個々のコマンドシーケンスのランタイムを測定する場合、3 つの個別のネットワークが必要です。 プログラム内の個々のネットワークで、「プログラムランタイムの測定」命令を呼び出します。 命令のこの最初の呼び出しでランタイム測定の開始位置を設定します。 その後、次のネットワーク内で目的のプログラムブロック、またはコマンドシーケンスを呼び出します。 他のネットワーク内で、「プログラムランタイムの測定」命令の 2 回目の呼び出しを行い、命令の最初の呼び出しで行ったように同じメモリを IN-OUT パラメータ MEM に割り当てます。 3 番目のネットワークの「プログラムランタイムの測定」命令は内部 CPU カウンタを読み取り、内部カウンタ周波数に基づいてプログラムブロック、またはコマンドシーケンスの現在のランタイムを計算し、それを出力 RET_VAL に書き込みます。

注記
コマンドシーケンスのランタイムは、正確に測定することはできません。これは、コマンドシーケンス内の命令のシーケンスが、プログラムを最適にコンパイルする過程で変更されるためです。

構文

以下の構文が「プログラムランタイムの測定」命令に使用されます。

```
CALL RUNTIME
RET_VAL := <operand>
MEM := <operand>
```

パラメータ

次の表に、「プログラムランタイムの測定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Output	LREAL	I、Q、M、D、L	測定したランタイムを秒単位で返します
MEM	InOut	LREAL	I、Q、M、D、L	ランタイム測定の開始位置を保存します

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がプログラムブロックのランタイム計算に基づいてどのように動作するかを示します。

STL	説明
CALL RUNTIME	// 命令が呼び出されます。
RET_VAL := "TagResult"	// 中間結果
MEM := "TagMemory"	// ランタイム測定の開始位置を保存する。
CALL "Best_before_date", "Best_before_date_DB"	// 「Best_before_date」ブロックが呼び出され、処理される。
CALL RUNTIME	// 命令が再び呼び出されます。
RET_VAL := "TagResult"	// 命令の結果
MEM := "TagMemory"	// ランタイム測定の基準として、ランタイム測定の開始点を使用する。

この命令の最初の呼び出しでランタイムの測定が開始され、TagMemory オペランドでの命令の 2 回目の呼び出しの参照先としてバッファリングされます。

「Best_before_date」プログラムブロック FB1 が呼び出されます。

プログラムブロック FB1 が処理されると、命令の 2 回目の実行が行われます。 命令の 2 回目の呼び出しにより、プログラムブロックのランタイムが計算され、結果が RET_VAL 出力の TagResult オペランドに書き込まれます。

ワード論理演算

この章には下記に関する情報が記載されています：

- DECO: デコード (S7-1500)
- ENCO: エンコード (S7-1500)
- SEL: 選択 (S7-1500)

DECO: デコード

説明

「デコード」命令を使用して、入力値で指定されたビットを出力値にセットします。

「デコード」命令は、IN パラメータの値を読み取り、読み取った値とビット位置が一致する OUT パラメータにそのビットをセットします。 出力値の他のビットはゼロで埋められます。 IN パラメータの値が 31 よりも大きい場合、モジュロ 32 演算が実行されます。

構文

「デコード」命令には、以下の構文を使用します。

```
CALL DECO
UINT ???
IN := <operand>
OUT := <operand>
```

パラメータ

次の表に、「デコード」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	UINT	I、Q、M、D、L、P、または定数	設定される出力値のビット位置
OUT	Output	ビット列	I、Q、M、D、L、P	出力値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

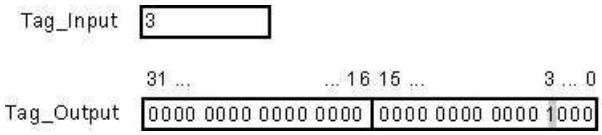
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL DECO	// 命令が呼び出されます。
UINT WORD	// 命令のデータタイプ
IN := "Tag_Input"	// 出力値にセットされるビットを指定する入力値
OUT := "Tag_Output"	// 結果

次の表に、命令が特定の値を使用してどのように動作するかを示します。



この命令はオペランド「Tag_Input」の値からビット番号「3」を読み取り、3 番目のビットをオペランド「Tag_Output」の値にセットします。

ENCO: エンコード

説明

「エンコード」命令を使用して、入力値の最下位ビットのビット番号を読み出し、それをパラメータ OUT に出力します。

「エンコード」命令は IN パラメータの値の最下位ビットを選択し、そのビット番号を OUT パラメータのオペランドに書き込みます。 IN パラメータに値 DW#16#00000001 または DW#16#00000000 が含まれる場合、値「0」が OUT パラメータに出力されます。

構文

「エンコード」命令には、以下の構文を使用します。

```
CALL ENCO
???
IN  := <operand>
OUT  := <operand>
```

パラメータ

次の表に、「エンコード」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	ビット列	I、Q、M、D、L、P、または定数	入力値
OUT	Output	INT	I、Q、M、D、L、P	出力値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

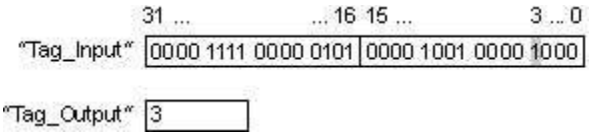
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL ENCO	// 命令が呼び出されます。
WORD	// 命令のデータタイプ
IN := "Tag_Input"	// 最下位ビットの入力値が読み取られる。
OUT := "Tag_Output"	// 最下位ビットのビット番号が入力値にセットされる。

次の表に、命令が特定の値を使用してどのように動作するかを示します。



この命令は変数「Tag_Input」のセットされた最下位ビットを選択し、そのビット位置「3」を変数「Tag_Output」に書き込みます。

SEL: 選択

説明

「選択」命令を使用して、スイッチ(G 入力)に応じて、2 つの入力 IN0 または IN1 のいずれかを選択し、その内容を OUT 出力に移動することができます。 G 入力のシグナル状態が「0」の場合、IN0 入力の値が移動されます。 G 入力のシグナル状態が「1」の場合、IN1 入力の値が OUT 出力に移動されます。

命令は、すべてのパラメータのタグが同じデータタイプの場合のみ実行されます。

構文

「選択」命令には、以下の構文を使用します。

```
CALL SEL
???
G  := <operand>
IN0  := <operand>
IN1  := <operand>
OUT  := <operand>
```

パラメータ

次の表に、「選択」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
G	Input	BOOL	I、Q、M、D、L、または定数	スイッチ
IN0	Input	ビット列、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、DT、LDT	I、Q、M、D、L、P、または定数	最初の入力値
IN1	Input	ビット列、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD, DATE、DT、LDT	I、Q、M、D、L、P、または定数	2 番目の入力値
OUT	Output	ビット列、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD, DATE、DT、LDT	I、Q、M、D、L、P	結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL SEL	// 命令が呼び出されます。
DWORD	// 命令のデータタイプ
G := "Tag_Input_G"	// スイッチのシグナル状態
IN0 := "Tag_Input0"	// 最初の入力値
IN1 := "Tag_Input1"	// 2 番目の入力値
OUT := "Tag_Output"	// 選択した入力の値

次の表に、命令が特定の値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

レガシー

この章には下記に関する情報が記載されています：

DRUM:PLC 実行

説明

「PLC の実装」命令を使用して、プログラムされた出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)に関連するステップの OUT_VAL パラメータのプログラム値を割り当てます。そのため、この命令が特定のステップに留まっている間は、このステップは S_MASK パラメータでプログラムされた許可マスクの条件を満たす必要があります。 このステップのイベントが真(TRUE)になっていて現在のステップのプログラムされた時間が経過した場合、または JOG パラメータの値が「0」から「1」に切り替わった場合、この命令は次のステップに進みます。 RESET パラメータのシグナル状態が「1」に切り替わると、この命令はリセットされます。 これによって、プリセットされたステップ(DSP)と現在のステップが等しくなります。

このステップで消費された時間の量は、プリセットタイムベース(DTBP)と各ステップのプリセットカウンタ値(S_PRESET)の積によって計算されます。 新しいステップの開始時に、この計算値が DCC パラメータにロードされます。このパラメータには、現在のステップの残り時間が含まれています。たとえば、DTBP パラメータの値が「2」で、最初のステップのプリセット値が「100」(100 ミリ秒)の場合、DCC パラメータの値は「200」(200 ミリ秒)になります。

ステップは、時間値、イベント、またはその両方を使用してプログラム可能です。イベントビットおよび時間値「0」を持つステップは、イベントビットのシグナル状態が「1」になると直ちに次のステップに進みます。時間値のみを使用してプログラムされたステップは、時間を直ちに開始します。「0」よりも大きなイベントビットおよび時間値を使用してプログラムされたステップは、イベントビットのシグナル状態が「1」になると時間を開始します。 イベントビットは、シグナル状態「1」で初期化されます。

プログラムされた最後のステップ(LST_STEP)の実行中にこのステップに割り当てられた時間が経過した場合、Q パラメータのシグナル状態は「1」にセットされ、それ以外の場合は「0」にセットされます。 Q パラメータがセットされると、この命令はリセットされるまでこのステップに留まります。

設定可能なマスク(S_MASK)で、出力ワード(OUT_WORD)の個々のビットを選択し、出力値(OUT_VAL)によって出力ビット(OUT1 ~ OUT16)をセットまたはリセットできます。 設定可能なマスクのビットのシグナル状態が「1」の場合、OUT_VAL 値は対応するビットをセットまたはリセットします。 設定可能なマスクビットのシグナル状態が「0」の場合、対応するビットは変更されません。16 のステップすべてに設定可能なマスクのすべてのビットは、シグナル状態「1」で初期化されます。

OUT1 パラメータの出力ビットは、出力ワード(OUT_WORD)の最下位ビットに対応します。OUT16 パラメータの出力ビットは、出力ワード(OUT_WORD)の最上位ビットに対応します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「PLC の実装」命令には、以下の構文を使用します。

```
CALL DRUM, "インスタンス DB"
RESET := <Operand>
JOG := <Operand>
```

```
DRUM_EN := <Operand>
LST_STEP := <Operand>
EVENT1 - 16 := <Operand>
OUT1 - 16 := <Operand>
Q := <Operand>
OUT_WORD := <Operand>
ERR_CODE := <Operand>
```

パラメータ

次の表に、「PLC 実装」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RESET	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、リセット条件を示します。
JOG	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「0」から「1」に切り替わると、命令が次のステップに進みます。
DRUM_EN	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「1」になると、PLC はイベントおよび時間条件に基づいて進むことができます。
LST_STEP	Input	BYTE	I、Q、M、D、L、または定数	プログラムされた最後のステップの番号。
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I、Q、M、D、L、または定数	イベントビット(i); 初期シグナル状態は「1」。
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I、Q、M、D、L	出力ビット(j)
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
OUT_WORD	Output	WORD	I、Q、M、D、L、P	PLC が出力値を書き込むワードアドレス。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
JOG_HIS	Static	BOOL	I、Q、M、D、L、または定数	JOG パラメータの履歴ビット
EOD	Static	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
DSP	Static	BYTE	I、Q、M、D、L、P、または定数	PLC のプリセットされた最初のステップ(1 ~ 16)
DSC	Static	BYTE	I、Q、M、D、L、P、または定数	PLC の現在のステップ

DCC	Static	DWORD	I、Q、M、D、L、P、または定数	現在のステップの残り処理時間
DTBP	Static	WORD	I、Q、M、D、L、P、または定数	PLC のプリセットタイムベース
PrevTime	Static	TIME	I、Q、M、D、L、または定数	前回の呼び出しのシステム時刻
S_PRESET	Static	ARRAY[1..16] of WORD	I、Q、M、D、L、または定数	1 クロックパルス = 1 ミリ秒の場合の各ステップ[1 ~ 16]のプリセットされたカウンタ値。
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの出力値[1 ~ 16, 0 ~ 15]。
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの設定可能なマスク[1 ~ 16, 0 ~ 15]。 初期シグナル状態は「1」。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

ERR_CODE*	説明
W#16#0000	エラーは発生していません。
W#16#000B	LST_STEP パラメータの値が 1 未満であるか、または 16 を超えています。
W#16#000C	DSC パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
W#16#000D	DSP パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例で命令はステップ 1 からステップ 2 に進みます。出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)はステップ 2 に構成されたマスクおよび OUT_VAL パラメータの値によってセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

STL	説明
CALL DRUM, "DRUM_DB"	// 「PLC の実装」命令が呼び出され、インスタンスデータブロック「DRUM_DB」が作成される。
RESET := "Tag_Reset"	// リセット入力
JOG := "Tag_Input_Jog"	// 信号立ち上がりエッジで命令は次のステップに進む。

DRUM_EN := "Tag_Input_DrumEN"	// シグナル状態が「1」になると、PLC はイベントおよび時間条件に基づいて進むことができます。
LST_STEP := "Tag_Number_LastStep"	// プログラムされた最後のステップの番号
EVENT1 := "MyTag_Event_1"	// イベントビット 1
EVENT2 := "MyTag_Event_2"	// イベントビット 2
EVENT3 := "MyTag_Event_3"	// イベントビット 3
EVENT4 := "MyTag_Event_4"	// イベントビット 4
EVENT5 := "MyTag_Event_5"	// イベントビット 5
EVENT6 := "MyTag_Event_6"	// イベントビット 6
EVENT7 := "MyTag_Event_7"	// イベントビット 7
EVENT8 := "MyTag_Event_8"	// イベントビット 8
EVENT9 := "MyTag_Event_9"	// イベントビット 9
EVENT10 := "MyTag_Event_10"	// イベントビット 10
EVENT11 := "MyTag_Event_11"	// イベントビット 11
EVENT12 := "MyTag_Event_12"	// イベントビット 12
EVENT13 := "MyTag_Event_13"	// イベントビット 13
EVENT14 := "MyTag_Event_14"	// イベントビット 14
EVENT15 := "MyTag_Event_15"	// イベントビット 15
EVENT16 := "MyTag_Event_16"	// イベントビット 16
OUT1 := "MyTag_Output_1"	// 出力ビット 1
OUT2 := "MyTag_Output_2"	// 出力ビット 2
OUT3 := "MyTag_Output_3"	// 出力ビット 3
OUT4 := "MyTag_Output_4"	// 出力ビット 4
OUT5 := "MyTag_Output_5"	// 出力ビット 5
OUT6 := "MyTag_Output_6"	// 出力ビット 6
OUT7 := "MyTag_Output_7"	// 出力ビット 7
OUT8 := "MyTag_Output_8"	// 出力ビット 8
OUT9 := "MyTag_Output_9"	// 出力ビット 9
OUT10 := "MyTag_Output_10"	// 出力ビット 10
OUT11 := "MyTag_Output_11"	// 出力ビット 11
OUT12 := "MyTag_Output_12"	// 出力ビット 12
OUT13 := "MyTag_Output_13"	// 出力ビット 13
OUT14 := "MyTag_Output_14"	// 出力ビット 14
OUT15 := "MyTag_Output_15"	// 出力ビット 15
OUT16 := "MyTag_Output_16"	// 出力ビット 16
Q := "Tag_Output_Q"	// シグナル状態「1」は、最後のステップの時間が経過したことを示します。
OUT_WORD := "Tag_OutputWord"	// PLC が出力値を書き込むワードアドレス
ERR_CODE := "Tag_ErrorCode"	// エラー情報

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータの初期化には以下の値が使用されます。

パラメータ	オペランド	アドレス	値
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

以下の値が命令の「DRUM_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE

OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

出力パラメータは、命令の実行前に以下の値にセットされます。

パラメータ	オペランド	アドレス	値
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE

OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	アドレス	値
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

命令の実行後、以下の値が命令のインスタンスデータブロック「DRUM_DB」で変更されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

DCAT:ディスクリート制御タイマアラーム

説明

「ディスクリート制御タイマアラーム」を使用して、CMD パラメータが開くまたは閉じるためのコマンドを発行した時点からの時間をカウントします。 時間は、指定した時間内でプリセット時間(PT)を超えるか、またはデバイスの開閉に関する情報を受信するまで蓄積します(O_FB または C_FB)。 デバイスの開閉に関する情報を受信する前にプリセット済み時間が経過すると、対応するアラームが有効になります。プリセット済み時間の前にコマンド入力の信号状態が切り替わると、この時間経過が再開します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

「ディスクリート制御タイマアラーム」命令は、入力条件に対して以下のような影響があります。

- CMD パラメータの信号状態が「0」から「1」に切り替わると、パラメータ Q、CMD_HIS、ET (ET < PT の場合のみ)、OA、および CA の信号状態は、以下のような影響を受けます。
  - Q および CMD_HIS パラメータが「1」にセットされます。
  - ET、OA および CA パラメータが「0」にリセットされます。
- CMD パラメータの信号状態が「1」から「0」に切り替わると、Q、ET (ET < PT の場合のみ)、OA、CA、および CMD_HIS パラメータが「0」にリセットされます。
- CMD および CMD_HIS パラメータの信号状態が「1」で、パラメータ O_FB が「0」にセットされると、命令が最後に実行されてからの時間差(ミリ秒)がパラメータ ET の値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、OA パラメータの信号状態が「1」にセットされます。ET パラメータの値が PT パラメータの値を超えなければ、OA パラメータの信号状態が「0」にリセットされます。 CMD_HIS パラメータの値が、CMD パラメータの値にリセットされます。
- CMD、CMD_HIS、および O_FB パラメータの信号状態が「1」にセットされ、C_FB パラメータの値が「0」の場合、OA パラメータの信号状態が「0」にセットされます。ET パラメータの値が、PT パラメータの値にセットされます。O_FB パラメータの信号状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。CMD_HIS パラメータの値が、CMD パラメータの値にセットされます。
- CMD、CMD_HIS および C_FB パラメータの値が「0」の場合、命令が最後に実行されてからの時間差(ms)が ET パラメータの値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、CA パラメータの信号状態が「1」にセットされます。 PT パラメータの値を超えなければ、CA パラメータの信号状態は「0」です。CMD_HIS パラメータの値が、CMD パラメータの値にセットされます。
- CMD、CMD_HIS および O_FB パラメータの信号状態が「0」のときに C_FB パラメータが「1」にセットされると、CA パラメータが「0」にセットされます。ET パラメータの値が、PT パラメータの値にセットされます。C_FB パラメータの信号状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。CMD_HIS パラメータの値が、CMD パラメータの値にセットされます。
- O_FB および C_FB パラメータの信号状態が同時に「1」の場合、両方のアラーム出力の信号状態が「1」にセットされます。

「ディスクリート制御タイマアラーム」命令は、エラー情報を返しません。

構文

「ディスクリート制御タイマアラーム」命令には、以下の構文を使用します。

```
CALL DCAT, "インスタンス DB"
CMD := <Operand>
O_FB := <Operand>
C_FB := <Operand>
Q := <Operand>
OA := <Operand>
CA := <Operand>
```

パラメータ

次の表に、「ディスクリート制御タイマアラーム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CMD	Input	BOOL	I、Q、M、D、L、または定数	信号状態「0」は、「閉じる」コマンドを示します。 信号状態「1」は、「開く」コマンドを示します。
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
Q	Output	BOOL	I、Q、M、D、L	CMD パラメータのステータスを示します
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒
PT	Static	DINT	D、L、または定数	プリセットタイマ値(1 クロックパルス = 1 ミリ秒)
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
CMD_HIS	Static	BOOL	D、L、または定数	CMD の履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例では、パラメータ CMD が「0」から「1」に切り替わります。 命令の実行後、パラメータ Q が「1」にセットされ、2 つのアラーム出力 OA および CA の信号状態は「0」になります。 インスタンスデータブロックのパラメータ CMD_HIS が信号状態「1」にセットされ、パラメータ ET が「0」にリセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

STL	説明
CALL DCAT, "DCAT_DB"	//「ディスクリート制御タイマアラーム」命令が呼び出されインスタンスデータブロック「DCAT_DB」が作成される。
CMD := "Tag_Input_CMD"	// デバイスを開くまたは閉じる
O_FB := "Tag_Input_O_FB"	// 開くときにフィードバック
C_FB := "Tag_Input_C_FB"	// 閉じるときにフィードバック
Q := "Tag_Output_Q"	// パラメータのステータスを示す。CMD
OA := "Tag_Output_OA"	// 開くときにアラーム出力
CA := "Tag_Output_CA"	// 閉じるときにアラーム出力

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令の「DCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令の「DCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0

CMD_HIS	DBX16.0	TRUE

MCAT: モータ制御タイマアラーム

説明

「モータ制御タイマアラーム」命令を使用して、コマンド入力(開または閉)の 1 つが有効になる時点から、時間を累積します。 時間はプリセット済み時間が経過するまで、または関連のフィードバック入力が、デバイスが指定された時間内に要求された操作を実行したことを示す場合に蓄積されます。 フィードバックが受信される前にプリセット済み時間が経過すると、対応するアラームがトリガされます。

プログラムに命令を挿入すると、[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存するか、ブロックインターフェースにローカルタグ(マルチインスタンス)として保存するかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「モータ制御タイマアラーム」命令の実行

次の表に、さまざまな入力条件に対する「モータ制御タイマアラーム」命令の応答を示します。

入力パラメータ								出力パラメータ								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	状態
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening (開き始める)
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open (開く)
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened (開いた)
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm (アラームを開いている)
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing

															(閉じ始める)
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close (閉じる)
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed (閉じた)
>= PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm (アラームを閉じている)
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped (停止)

凡例:

INC	FB から ET への最後に処理されてからの時間差(ms)を追加します。
PT	PT は ET と同じ値にセットされます。
X	使用不可
<PT	ET < PT
>=PT	ET >= PT

入力パラメータ O_HIS および C_HIS のシグナル状態が両方とも「1」の場合、これらのパラメータのシグナル状態が直ちに「0」にセットされます。 この場合、上記の表の最後の行(X)が有効になります。 入力パラメータ O_HIS および C_HIS のシグナル状態が「1」であるかどうかをチェックすることができないため、この場合の出力パラメータは以下のようにセットされます。

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

構文

以下の構文が、「モータ制御タイマアラーム」命令に使用されます。

```
CALL MCAT, "インスタンス
DB"
O_CMD := <operand>
C_CMD := <operand>
S_CMD := <operand>
O_FB := <operand>
C_FB := <operand>
OO := <operand>
CO := <operand>
```

OA := <operand>

CA := <operand>

Q := <operand>

パラメータ

次の表に、「モータ制御タイマアラーム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
O_CMD	Input	BOOL	I、Q、M、D、L、または定数	[開]コマンド入力
C_CMD	Input	BOOL	I、Q、M、D、L、または定数	[閉]コマンド入力
S_CMD	Input	BOOL	I、Q、M、D、L、または定数	[停止]コマンド入力
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
OO	Output	BOOL	I、Q、M、D、L	[開]出力
CO	Output	BOOL	I、Q、M、D、L	[閉]出力
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「0」は、エラー条件を示します。
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒
PT	Static	DINT	D、L、または定数	プリセットタイマ値(1 クロックパルス = 1 ミリ秒)
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
O_HIS	Static	BOOL	D、L、または定数	[開]履歴ビット
C_HIS	Static	BOOL	D、L、または定数	[閉]履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。

STL	説明
CALL MCAT, "MCAT_DB"	//「モータ制御タイマアラーム」命令が呼び出されインスタンスデータブロック「MCAT_DB」が作成される。
O_CMD := "Tag_Input_O_CMD"	// [開]コマンド入力

```
C_CMD := "Tag_Input_C_CMD"      // [閉]コマンド入力
S_CMD := "Tag_Input_S_CMD"      // [停止]コマンド入力
O_FB := "Tag_Input_O_FB"        // 開くときにフィードバック
C_FB := "Tag_Input_C_FB"        // 閉じるときにフィードバック
OO := "Tag_OutputOpen"          // [開]出力
CO := "Tag_OutputClosed"        // [閉]出力
OA := "Tag_Output_OA"           // 開くときにアラーム出力
CA := "Tag_Output_CA"           // 閉じるときにアラーム出力
Q := "Tag_Output_Q"             // エラー情報
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

以下の値が命令の「MCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE

CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

以下の値が命令の「MCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

IMC:入力ビットをマスクビットと比較

説明

「入力ビットをマスクビットと比較」命令を使用して、対応するマスクビットと最大 16 のプログラム済み入力ビット(IN_BIT0 ~ IN_BIT15)の信号状態を比較できます。 最大 16 のマスクされたステップをプログラム可能です。IN_BIT0 パラメータの値が、CMP_VAL[x,0]マスクの値と比較されます。「x」はステップ番号を示しています。 CMP_STEP パラメータで、比較に使用するマスクのステップ番号を指定します。 プログラムされた値は、すべて同じ方法で比較されます。プログラムされていない入力ビットやマスクビットは、デフォルトの信号状態である FALSE になります。

比較で一致が検出されると、OUT パラメータの信号状態が「1」にセットされます。それ以外の場合、OUT パラメータが「0」にセットされます。

CMP_STEP パラメータの値が 15 よりも大きい場合、この命令は実行されません。ERR_CODE パラメータでエラーメッセージが出力されます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

構文

「入力ビットをマスクビットと比較」命令には、以下の構文を使用します。

```
CALL IMC, "インスタンス DB"
IN_BIT0 - 15 := <Operand>
CMP_STEP := <Operand>
OUT := <Operand>
ERR_CODE := <Operand>
```

パラメータ

次の表に、「入力ビットをマスクビットと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。

IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
CMP_STEP	Input	BYTE	I、Q、M、D、L、P、または定数	比較に使用されるマスクのステップ番号。
OUT	Output	BOOL	I、Q、M、D、L	信号状態「1」は、一致が検出されたことを示します。 信号状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報

CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000A	CMP_STEP パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例では、すべての 16 の入力ビットがステップ 2 のマスクと比較されます。入力ビットがステップ 2 のマスクに一致するため、OUT パラメータの信号状態は TRUE にセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

STL	説明
CALL IMC, "IMC_DB"	//「入力ビットをマスクビットと比較」命令が呼び出されインスタンスデータブロック「IMC_DB」が作成される。
IN_BIT0 := "Tag_Input_BIT0"	// 入力ビット 0
IN_BIT1 := "Tag_Input_BIT1"	// 入力ビット 1
IN_BIT2 := "Tag_Input_BIT2"	// 入力ビット 2
IN_BIT3 := "Tag_Input_BIT3"	// 入力ビット 3
IN_BIT4 := "Tag_Input_BIT4"	// 入力ビット 4
IN_BIT5 := "Tag_Input_BIT5"	// 入力ビット 5
IN_BIT6 := "Tag_Input_BIT6"	// 入力ビット 6
IN_BIT7 := "Tag_Input_BIT7"	// 入力ビット 7
IN_BIT8 := "Tag_Input_BIT8"	// 入力ビット 8
IN_BIT9 := "Tag_Input_BIT9"	// 入力ビット 9
IN_BIT10 := "Tag_Input_BIT10"	// 入力ビット 10
IN_BIT11 := "Tag_Input_BIT11"	// 入力ビット 11
IN_BIT12 := "Tag_Input_BIT12"	// 入力ビット 12
IN_BIT13 := "Tag_Input_BIT13"	// 入力ビット 13
IN_BIT14 := "Tag_Input_BIT14"	// 入力ビット 14
IN_BIT15 := "Tag_Input_BIT15"	// 入力ビット 15

```
CMP_STEP := "Tag_CMP_STEP"   // マスクのステップ番号
OUT := "Tag_Output"          // 一致が検出されたかどうかを示す
ERR_CODE := "Tag_ErrorCode"  // エラー情報
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

ステップ 2 のマスクの以下の値が命令の「IMC_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE

CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

SMC:スキャンマトリックスの比較

説明

「スキャンマトリックスの比較」命令を使用して 16 のプログラム済み入力ビット(IN_BIT0～IN_BIT15)の信号状態を各ステップの比較マスクの対応するビットと比較します。 処理はステップ 1 から開始し、プログラムされた最後のステップ(LAST)まで、または一致が検出されるまで続行されます。IN_BIT0 パラメータの入力ビットが、CMP_VAL[x,0]マスクの値と比較されます。「x」はステップ番号を示しています。 プログラムされた値は、すべて同じ方法で比較されます。 一致が検出されると、OUT パラメータの信号状態が「1」にセットされ、一致するマスクのステップ番号が OUT_STEP パラメータに書き込まれます。プログラムされていない入力ビットやマスクビットは、デフォルトの信号状態である FALSE になります。 複数のステップに一致するマスクが存在する場合、最初に検出されたマスクのみが OUT_STEP パラメータに示されます。 一致が検出されないと、OUT パラメータの信号状態が「0」にセットされます。 この場合、OUT_STEP パラメータの値は LAST パラメータの値よりも「1」だけ大きくなります。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

構文

「スキャンマトリックスの比較」命令には、以下の構文を使用します。

```
CALL SMC, "インスタンス DB"
IN_BIT0 - 15 := <Operand>
OUT := <Operand>
OUT_STEP := <Operand>
ERR_CODE := <Operand>
```

パラメータ

次の表に、「スキャンマトリックスの比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。

IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
OUT	Output	BOOL	I、Q、M、D、L	信号状態「1」は、一致が検出されたことを示します。 信号状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT_STEP	Output	BYTE	I、Q、M、D、L、P	一致するマスクを持つステップの番

				号が含まれているか、または一致が検出されない場合は、LAST パラメータの値よりも「1」だけ大きいステップ番号が含まれています。
LAST	Static	BYTE	I、Q、M、D、L、P、または定数	一致するマスクをスキャンする最後のステップのステップ番号を指定します。
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0～15, 0～15]:インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000E	LAST パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

例

この例で、すべての 16 個の入力ビットは一致が検出されるまでステップ 0～5 までのマスクと比較されます。 ステップ 2 が入力ビットと一致したので、ステップ 0～2 のマスクのみがスキャンされました。

注記
静的なパラメータをデータブロックで初期化できます。

STL	説明
CALL SMC, "SMC_DB"	//「スキャンマトリックスの比較」命令が呼び出されインスタンスデータブロック「SMC_DB」が作成される
IN_BIT0 := "Tag_Input_BIT0"	// 入力ビット 0
IN_BIT1 := "Tag_Input_BIT1"	// 入力ビット 1
IN_BIT2 := "Tag_Input_BIT2"	// 入力ビット 2
IN_BIT3 := "Tag_Input_BIT3"	// 入力ビット 3

```
IN_BIT4 := "Tag_Input_BIT4"          // 入力ビット 4
IN_BIT5 := "Tag_Input_BIT5"          // 入力ビット 5
IN_BIT6 := "Tag_Input_BIT6"          // 入力ビット 6
IN_BIT7 := "Tag_Input_BIT7"          // 入力ビット 7
IN_BIT8 := "Tag_Input_BIT8"          // 入力ビット 8
IN_BIT9 := "Tag_Input_BIT9"          // 入力ビット 9
IN_BIT10 := "Tag_Input_BIT10"        // 入力ビット 10
IN_BIT11 := "Tag_Input_BIT11"        // 入力ビット 11
IN_BIT12 := "Tag_Input_BIT12"        // 入力ビット 12
IN_BIT13 := "Tag_Input_BIT13"        // 入力ビット 13
IN_BIT14 := "Tag_Input_BIT14"        // 入力ビット 14
IN_BIT15 := "Tag_Input_BIT15"        // 入力ビット 15
OUT := "Tag_Output"                  // 一致が検出されたかどうかを示す
OUT_STEP := "Tag_Out-
put_STEP"                            // 関連するマスクのステップ番号を含む
ERR_CODE := "Tag_ErrorCode"          // エラー情報
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

ステップ 2 のマスクの以下の値が命令の「SMC_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output	TRUE
OUT_STEP'	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

LEAD_LAG:リード/ラグアルゴリズム

説明

「リード/ラグアルゴリズム」命令を使用してアナログタグによってシグナルを処理できます。 GAIN パラメータのゲイン値はゼロよりも大きくなければなりません。「リード/ラグアルゴリズム」命令の結果は、以下の等式を使用して計算されます。

$$OUT = \left[\frac{LG\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_OUT + GAIN \left[\frac{LD\_TIME + SAMPLE\_T}{LG\_TIME + SAMPLE\_T}\right] IN - GAIN \left[\frac{LD\_TIME}{LG\_TIME + SAMPLE\_T}\right] * PREV\_IN$$

「リード/ラグアルゴリズム」命令は、処理を固定プログラムサイクルで行う場合のみ正しい結果が得られます。 LD_TIME、LG_TIME、および SAMPLE_T パラメータには同一のユニットを指定する必要があります。 LG_TIME > 4 + SAMPLE_T の場合、この命令は以下の関数を実行します。

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

GAIN パラメータの値がゼロ以下の場合、計算は実行されず、ERR_CODE パラメータでエラー情報が出力されます。

「リード/ラグアルゴリズム」命令を動的フィードフォワード制御でループと共に補償器として使用できます。 この命令は、2 つの操作で構成されています。「Lead」操作は OUT 出力のフェーズをシフトして、出力が入力の前に来るようにします。 一方「Lag」操作は出力をシフトして、出力が入力の後になるようにします。「Lag」演算は積分に相当するため、ノイズサプレッサやローパスフィルタとして使用できます。「Lead」演算は微分と同等なため、ハイパスフィルタとして使用できます。2 つの演算(Lead および Lag)をまとめて実行すると、低周波域では出力フェーズが入力フェーズよりも遅れ、高周波域では出力フェーズが入力フェーズに先行します。 つまり「リード/ラグアルゴリズム」命令は帯域フィルタとして使用できます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

構文

「リード/ラグアルゴリズム」命令には、以下の構文を使用します。

```
CALL LEAD_LAG, "インスタン
ス DB"
IN := <Operand>
SAMPLE_T := <Operand>
OUT := <Operand>
ERR_CODE := <Operand>
```

パラメータ

次の表に、「リード/ラグアルゴリズム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	REAL	I、Q、M、D、L、P、または定数	処理対象の現在のサンプル時間(サイクルタイム)の入力値
SAMPLE_T	Input	INT	I、Q、M、D、L、P、または定数	サンプル時間
OUT	Output	REAL	I、Q、M、D、L、P	命令の結果
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
LD_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のリードタイム。
LG_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のラグタイム。
GAIN	Static	REAL	I、Q、M、D、L、P、または定数	% / %で表されるゲイン(定常状態時の出力の切り替えと入力の切り替えとの比率)。
PREV_IN	Static	REAL	I、Q、M、D、L、P、または定数	前の入力
PREV_OUT	Static	REAL	I、Q、M、D、L、P、または定数	前の出力

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0009	GAIN パラメータの値がゼロ以下です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。

STL	説明
CALL LEAD_LAG, "LEAD_LAG_DB"	// 「リード/ラグアルゴリズム」命令が呼び出されインスタンスデータブロック「LEAD_LAG_DB」が作成される。
IN := "Tag_Input"	// 処理対象の現在のサンプル時間(サイクルタイム)の入力値。
SAMPLE_T := "Tag_Input_SAMPE_T"	// サンプル時間
OUT := "Tag_Output_Result"	// 命令の結果
ERR_CODE := "Tag_ErrorCode"	// エラー情報

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータで以下の値が使用されます。

パラメータ	オペランド	値
IN	Tag_Input	2.0
SAMPLE_T	Tag_InputSampleTime	10

以下の値が命令の「LEAD_LAG_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output_Result	2.0

以下の値が命令の「LEAD_LAD_DB」インスタンスデータブロックに保存されます。

パラメータ	オペランド	値
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

SEG:7 セグメント表示のビットパターン作成

説明

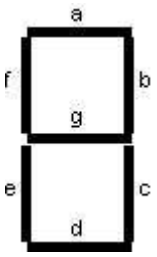
「7 セグメント表示のビットパターンの作成」命令は、指定されたソースワード(IN)内の 16 進数の 4 桁を 1 つずつ 7 セグメント表示用の 4 つの同等のコードに変換し、それを出力(OUT)のダブルワードに書き込みます。

「7 セグメント表示のビットパターンの作成」は、いずれのエラー条件も認識しません。

入力の 16 進数値と出力のビットパターンの関係は以下のようになります。

桁	-gfedcba	表示
0000	00111111	0
0001	00000110	1
0010	01011011	2
0011	01001111	3
0100	01100110	4
0101	01101101	5
0110	01111101	6
0111	00000111	7
1000	01111111	8
1001	01100111	9
1010	01110111	A
1011	01111100	B
1100	00111001	C
1101	01011110	D
1110	01111001	E
1111	01110001	F

7 セグメント表示



構文

「7 セグメント表示のビットパターンの作成」命令には、以下の構文を使用します。

```
CALL SEG
IN := <Operand>
OUT := <Operand>
```

パラメータ

次の表に、「7 セグメント表示のビットパターンの作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	IN	WORD	I、Q、M、D、L、P、または定数	4 つの 16 進数のソースワード。
OUT	OUT	DWORD	I、Q、M、D、L、P	変換先の 4 バイトのビットパターン。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

STL	説明
CALL SEG	//「7 セグメント表示のビットパターンの作成」命令が呼び出されます。
IN := "Tag_Input"	// ソースワード
OUT := "Tag_Output"	// 7 セグメント表示のビットパターン

次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値	
		16 進数	2 進数
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16#065B4F66	00000110 01011011 01001111 01100110 表示: 1234

BCDCPL:10 の補数の作成

説明

「10 の補数の作成」命令を使用して、IN パラメータで指定された 7 桁の BCD 数の 10 の補数を作成します。 この命令は、次の数式を使用して計算します。

10000000 (BCD として)

– 7 桁の BCD 数

---

10 の補数(BCD として)

構文

「10 の補数の作成」命令には、以下の構文を使用します。

```
CALL BCDCPL
IN := <Operand>
ERR_CODE := <Operand>
```

パラメータ

次の表に、「10 の補数の作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	ビット列	I、Q、M、D、L、P、または定数	7 桁の BCD 数
ERR_CODE	Output	DWORD	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

STL	説明
CALL BCDCPL	// 「10 の補数の作成」命令が呼び出されます。
IN := "Tag_Input"	// 7 桁の BCD 数
ERR_CODE := "Tag_Output"	// 指定された BCD 数の 10 の補数

次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

BITSUM:セットビット数カウント

説明

「セットビット数カウント」命令を使用して、信号状態「1」にセットされるオペランドのビット数をカウントします。 ビットをカウントするオペランドは、IN パラメータで指定されます。命令の結果は、RET_VAL パラメータに出力されます。

構文

「セットビット数カウント」命令には、以下の構文を使用します。

```
CALL BITSUM
IN := <Operand>
RET_VAL := <Operand>
```

パラメータ

次の表に、「セットビット数カウント」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	DWORD	I、Q、M、D、L、P、または定数	セットビット数をカウントするオペランド
RET_VAL	Output	INT	I、Q、M、D、L、P	セットビット数

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

STL	説明
CALL BITSUM	// 「セットビット数カウント」命令が呼び出される。
IN := "Tag_Input"	// セットビット数をカウントするオペランド。
RET_VAL := "Tag_Output"	// オペランド「Tag_Input」のセットビット数。

次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 ビット)
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

STL ニーモニック

この章には下記に関する情報が記載されています：

- ビット論理演算 (S7-1500)
- タイマの動作 (S7-1500)
- カウンタ演算 (S7-1500)
- 比較演算 (S7-1500)
- 四則演算 (S7-1500)
- ロードと転送 (S7-1500)
- 変換操作 (S7-1500)
- プログラム制御演算 (S7-1500)
- ワード論理演算 (S7-1500)
- シフトとローテーション (S7-1500)
- 追加の命令 (S7-1500)

ビット論理演算

この章には下記に関する情報が記載されています：

A: AND 論理演算

説明

「論理積演算」命令を使用してバイナリオペランドのシグナル状態が「1」であるかどうかを照会し、照会結果を論理演算の結果(RLO)のシグナル状態によって論理積演算します。 これによって、照会結果はチェック済みオペランドのシグナル状態と同一になります。

リンクされたシグナル状態が両方とも「1」ならば、命令の実行後の RLO は「1」です。 リンクされたシグナル状態の 1 つが「0」ならば、命令の実行後の RLO は「0」です。

命令を連続して複数回使用すると共通論理演算の結果も「1」であるため、すべての照会結果は「1」になるはずです。

さらに「論理積演算」命令を使用してステータスワードのシグナル状態も照会できます。 これを実行するには、関連ステータスビット(==0、<>0、>0、<0, >=0、<=0、OV、OS、UO、BR)を命令のパラメータとして指定します。

構文

「AND 演算」命令には、以下の構文を使用します。

```
A <operand>
```

パラメータ

次の表に、「AND 演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L、T、C	シグナル状態を問い合わせるオペランド。

例

次の例で、命令がどのように動作するかを示します。

AN: 否定 AND 論理演算

説明

「否定論理積演算」命令を使用してバイナリオペランドのシグナル状態が「0」であるかどうかを照会し、照会結果を論理演算の結果(RLO)のシグナル状態によって論理積演算します。 これによって、照会結果はチェック済みオペランドの否定されたシグナル状態になります。 オペランドのシグナル状態が「0」の場合、照会結果は「1」です。 オペランドのシグナル状態が「1」の場合、照会結果は「0」です。

命令の実行前に照会結果および RLO のシグナル状態が「1」の場合、命令の実行後 RLO は「1」です。

命令の実行前に照会結果または RLO のシグナル状態が「0」の場合、命令の実行後 RLO は「0」です。

さらに「否定論理積演算」命令を使用してステータスワードのシグナル状態も照会できます。 これを実行するには、関連ステータスビット(==0、<>0、>0、<0, >=0、<=0、OV、OS、UO、BR)を命令のパラメータとして指定します。

構文

「否定論理積演算」命令には、以下の構文を使用します。

```
AN <operand>
```

パラメータ

次の表に、「否定論理積演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L、T、C	シグナル状態を問い合わせるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STLプログラム		リレー回路図	
		パワーレール	
A	"Tag_Input_1"	信号状態0:	a接点
AN	"Tag_Input_2"	信号状態1:	b接点
=	"Tag_Output"	信号状態0:	コイル

O: OR 論理和演算

説明

「論理和演算」命令を使用してバイナリオペランドのシグナル状態が「1」であるかどうかを照会し、照会結果を論理演算の結果(RLO)のシグナル状態によって論理和演算します。 これによって、照会結果はチェック済みオペランドのシグナル状態と同一になります。

リンクされたシグナル状態の 1 つが「1」ならば、命令の実行後の RLO は「1」です。 リンクされた両方のシグナル状態が「0」ならば、命令の実行後の RLO は「0」です。

命令を連続して複数回使用すると共通論理演算の結果も「1」であるため、照会結果の 1 つが「1」になることが十分あります。

さらに「論理和演算」命令を使用してステータスワードのシグナル状態も照会できます。 これを実行するには、関連ステータスビット(==0、<>0、>0、<0, >=0、<=0、OV、OS、UO、BR)を命令のパラメータとして指定します。

構文

「OR 演算」命令には、以下の構文を使用します。

```
O <operand>
```

パラメータ

次の表に、「OR 演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L、T、C	シグナル状態を問い合わせるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STLプログラム	リレー回路図
	パワーレール
O "Tag_Input_1"	信号状態1 a接点 (「Tag_Input_1」)
O "Tag_Input_2"	信号状態0 a接点 (「Tag_Input_2」)
= "Tag_Output"	信号状態1: コイル
	閉じたスイッチを示す

ON: 否定論理和演算

説明

「否定論理和演算」命令を使用してバイナリオペランドのシグナル状態が「0」であるかどうかを照会し、照会結果を論理演算の結果(RLO)のシグナル状態によって論理和演算します。 これによって、照会結果はチェック済みオペランドの否定されたシグナル状態になります。 オペランドのシグナル状態が「0」の場合、照会結果は「1」です。 オペランドのシグナル状態が「1」の場合、照会結果は「0」です。

命令の実行前に照会結果または RLO のシグナル状態が「1」の場合、命令の実行後 RLO は「1」です。 リンクされた両方のシグナル状態が「0」ならば、命令の実行後の RLO も「0」です。

命令を連続して複数回使用すると共通論理演算の結果も「1」であるため、照会結果の 1 つが「1」になることが十分あります。

さらに「否定論理和演算」命令を使用してステータスワードのシグナル状態も照会できます。 これを実行するには、関連ステータスビット(==0、<>0、>0、<0, >=0、<=0、OV、OS、UO、BR)を命令のパラメータとして指定します。

構文

「否定論理和演算」命令には、以下の構文を使用します。

```
ON <operand>
```

パラメータ

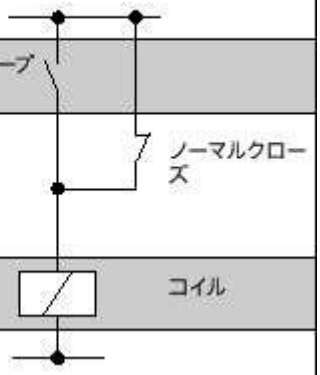
次の表に、「否定論理和演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L、T、C	シグナル状態がチェックされるオペランド

例

次の例で、命令がどのように動作するかを示します。

STLプログラム		リレー回路図	
		パワーレール	
O	"Tag_Input_1"	信号状態0	ノーマルオープン
オン	"Tag_Input_2"	信号状態1	ノーマルクローズ
=	"Tag_Output"	信号状態1	コイル



X: EXCLUSIVE OR 排他的論理和演算

説明

「排他的論理和演算」命令を使用してバイナリオペランドのシグナル状態が「1」であるかどうかを照会し、照会結果を論理演算の結果(RLO)のシグナル状態によって排他的論理和演算します。 これによって、照会結果はチェック済みオペランドのシグナル状態と同一になります。 リンクされたシグナル状態が等しくなければ、命令の実行後 RLO は「1」です。 両方のシグナル状態が等しい場合、RLO は「0」です。

たとえば、命令の実行前 RLO が「1」で照会されたオペランドが「1」の場合、RLO が「0」にリセットされます。 RLO が「1」かつオペランドのシグナル状態が「0」の場合、命令実行後 RLO は「1」にセットされます。

命令を連続して複数回使用すると共通の RLO の結果も「1」であるため、照会結果の 1 つが「1」になることが十分あります。 照会結果が「1」または「0」の場合、RLO が「0」にリセットされます。

さらに「排他的論理和演算」命令を使用して、ステータスワードのシグナル状態も照会できます。 これを実行するには、関連ステータスビット(==0、<>0、>0、<0, >=0、<=0、OV、OS、UO、BR)を命令のパラメータとして指定します。

構文

「排他的論理和演算」命令には、以下の構文を使用します。

```
X <operand>
```

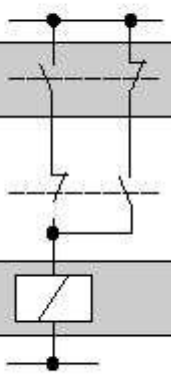
パラメータ

次の表に、「排他的論理和演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L、T、C	シグナル状態を問い合わせるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STLプログラム		リレー回路図	
		パワーレール	
X	"Tag_Input_1"	連絡先	
X	"Tag_Input_2"	連絡先	
=	"Tag_Output"	コイル	

XN: 否定排他的論理和演算

説明

「否定排他的論理和演算」命令を使用してバイナリオペランドのシグナル状態が「0」であるかどうかを照会し、照会結果を論理演算の結果(RLO)のシグナル状態によって論理和演算します。 これによって、照会結果はチェック済みオペランドの否定されたシグナル状態になります。 リンクされたシグナル状態が等しくなければ、命令の実行後 RLO は「1」です。 両方のシグナル状態が等しい場合、RLO は「0」です。

たとえば、照会するオペランドのシグナル状態が「0」の場合、照会結果は「1」です。 照会結果と RLO「1」を排他的論理和演算すると、シグナル状態は「0」となります。 RLO「0」で論理演算すると、シグナル状態は「1」となります。

さらに「否定排他的論理和演算」命令を使用して、ステータスワードのシグナル状態も照会できます。 これを実行するには、関連ステータスビット(==0、<>0、>0、<0, >=0、<=0、OV、OS、UO、BR)を命令のパラメータとして指定します。

構文

「否定排他的論理和演算」命令には、以下の構文を使用します。

```
XN <operand>
```

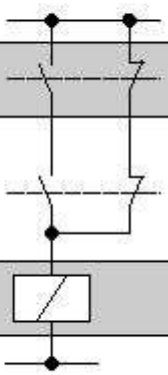
パラメータ

次の表に、「否定排他的論理和演算」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L、T、C	シグナル状態を問い合わせるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STLプログラム		リレー回路図	
		パワーレール	
X	"Tag_Input_1"	連絡先	
XN	"Tag_Input_2"	連絡先	
=	"Tag_Output"	コイル	

O: AND ファンクションの論理和演算

説明

「AND ファンクションの論理和演算」命令を使用して、論理和演算の前に複数の論理積演算を実行します。 論理積演算の全体の結果が一時的に保存され、以降の信号照会と論理和演算されます。

構文

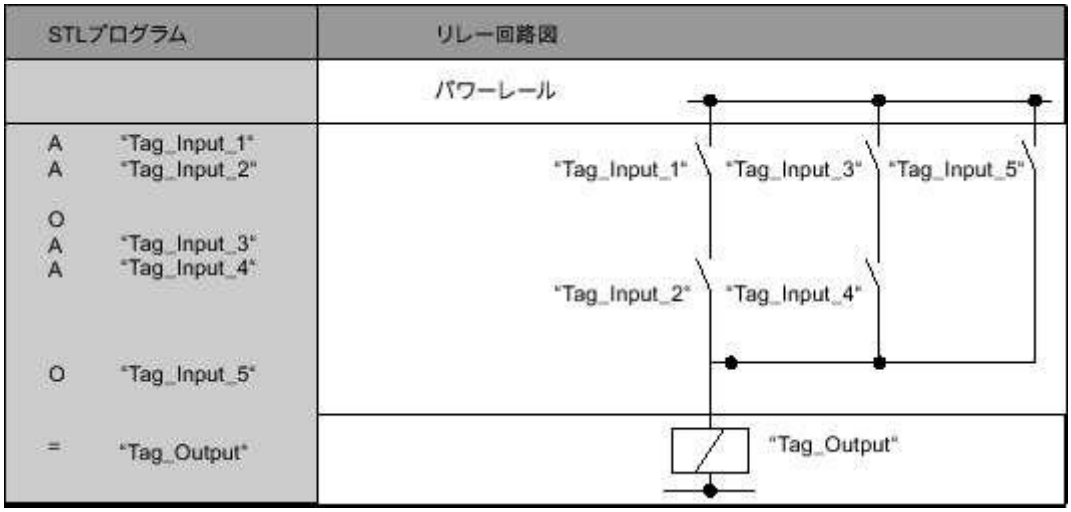
「AND ファンクションの論理和演算」命令には、以下の構文を使用します。

O

この命令の構文には、角括弧は不要です。

例

次の例で、命令がどのように動作するかを示します。



A(: 分岐付きの論理積演算

説明

「分岐付きの論理積演算」命令を使用して、論理積演算の前に括弧で括られた式の命令を実行します。

命令実行時には、CPU は現在の論理演算結果(RLO)をバイナリ結果 BR とともに保存します。 保存された RLO は、求められたネストされた式の論理演算結果と論理積演算されます。 ネスト化された式の後に追加の信号問い合わせが続く場合は、これも論理積が実行されます。

括弧内の式の中のさらに括弧に括られた式内に命令をプログラムすることで、括弧で括られた式をさらにネストすることが可能です。 これによって、ネストレベルは 7 レベルに制約されます。

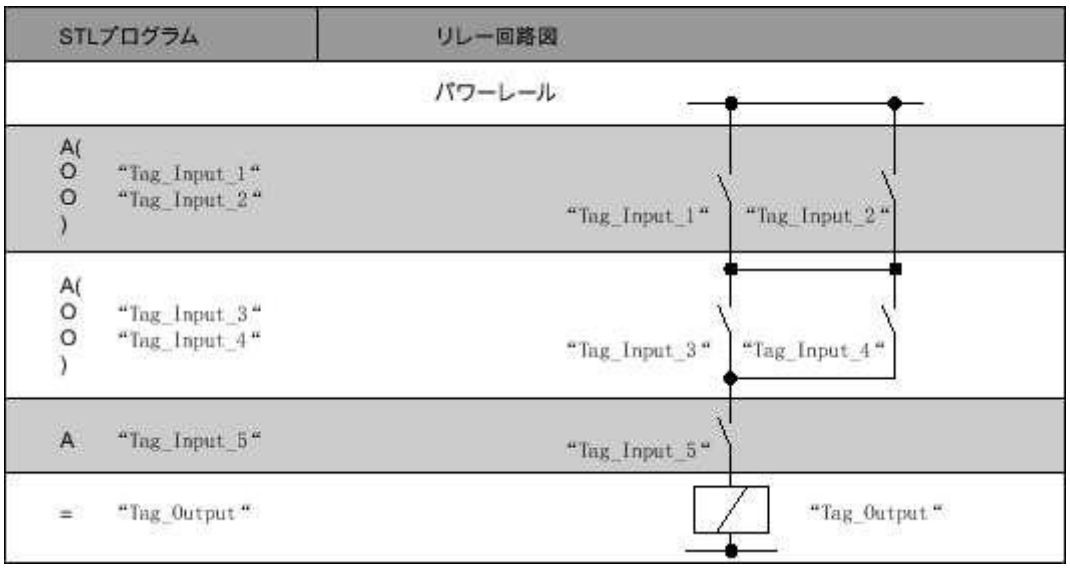
構文

「分岐付きの論理積演算」命令には、以下の構文を使用します。

```
A(
```

例

次の例で、命令がどのように動作するかを示します。



AN(: 分岐付きの否定論理積演算

説明

「分岐付きの否定論理積演算」命令を使用して、論理積演算の前に括弧で括られた式の命令を実行し、求められた括弧で括られた式の結果を否定します。

命令実行時には、CPU は現在の論理演算結果(RLO)をバイナリ結果 BR とともに保存します。 括弧で括られた式内の命令が実行された後、求められた括弧で括られた式の RLO が否定され、保存されている RLO と論理積演算されます。 括弧で括られた式の後に追加の信号問い合わせが続く場合は、これも論理積演算されます。

括弧内の式の中のさらに括弧に括られた式内に命令をプログラムすることで、括弧で括られた式をさらにネストすることが可能です。 これによって、ネストレベルは 7 レベルに制約されます。

構文

「分岐付きの否定論理積演算」命令には、以下の構文を使用します。

```
AN (
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
	// ネスト化された式の開始
AN(	// 括弧で括られた式の RLO を論理積演算し、求められた結果を否定する
O "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理和演算
O "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理和演算
)	// ネスト化された式の終了
AN "Tag_Input_3"	// オペランドのシグナル状態「0」を照会し、現在の RLO で論理積演算
= "Tag_Output"	// RLO のシグナル状態をオペランドに割り当て。

O(: 分岐付きの論理和演算

説明

「分岐付きの論理和演算」命令を使用して、論理和演算の前に括弧で括られた式の命令を実行します。

命令実行時には、CPU は現在の論理演算結果(RLO)をバイナリ結果 BR とともに保存します。 括弧で括られた式内の命令が実行された後、保存されている RLO と求められた括弧で括られた式の RLO が論理和演算されます。 括弧で括られた式の後に追加の信号問い合わせが続く場合は、これも論理和演算されます。

括弧内の式の中のさらに括弧に括られた式内に命令をプログラムすることで、括弧で括られた式をさらにネストすることが可能です。これによって、ネストレベルは 7 レベルに制約されます。

構文

「分岐付きの論理和演算」命令には、以下の構文を使用します。

```
O (
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
O(	// ネスト化された式の開始
	// // 括弧で括られた式の RLO を論理和演算
A "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
A "Tag_Input_3"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
)	// ネスト化された式の終了
= "Tag_Output"	// RLO のシグナル状態をオペランドに割り当て。

ON(: 否定論理和演算

説明

「分岐付きの否定論理和演算」命令を使用して、論理和演算の前に括弧で括られた式の命令を実行し、求められた括弧で括られた式の結果を否定します。

命令実行時には、CPU は現在の論理演算結果(RLO)をバイナリ結果 BR とともに保存します。 括弧で括られた式内の命令が実行された後、求められた括弧で括られた式の RLO が否定され、保存されている RLO と論理和演算されます。 括弧で括られた式の後に追加の信号問い合わせが続く場合は、これも論理和演算されます。

括弧内の式の中のさらに括弧に括られた式内に命令をプログラムすることで、括弧で括られた式をさらにネストすることが可能です。これによって、ネストレベルは 7 レベルに制約されます。

構文

「否定論理和演算」命令には、以下の構文を使用します。

```
ON (
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
	// ネスト化された式の開始
ON(	// 括弧で括られた式の RLO を論理和演算し、求められた結果を否定する
A "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
A "Tag_Input_3"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
)	// ネスト化された式の終了
= "Tag_Output"	// RLO のシグナル状態をオペランドに割り当て。

X(: 分岐付きの排他的論理和演算

説明

「分岐付きの排他的論理和演算」命令を使用して、排他的論理和演算の前に括弧で括られた式の命令を実行します。

命令実行時には、CPU は現在の論理演算結果(RLO)をバイナリ結果 BR とともに保存します。 括弧で括られた式内の命令が実行された後、保存されている RLO と求められた括弧で括られた式の RLO が排他的論理和演算されます。 括弧で括られた式の後に追加の信号問い合わせが続く場合は、これも排他的論理和演算されます。

括弧内の式の中のさらに括弧に括られた式内に命令をプログラムすることで、括弧で括られた式をさらにネストすることが可能です。これによって、ネストレベルは 7 レベルに制約されます。

構文

「分岐付きの排他的論理和演算」命令には、以下の構文を使用します。

```
X (
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
X(	// ネスト化された式の開始
	// // 括弧で括られた式の RLO を排他的論理和演算
A "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
A "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
)	// ネスト化された式の終了
X "Tag_Input_3"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
= "Tag_Output"	// RLO のシグナル状態をオペランドに割り当て。

XN(: 分岐付きの否定排他的論理和演算

説明

「分岐付きの否定排他的論理和演算」命令を使用して、排他的論理和演算の前に括弧で括られた式の命令を実行し、求められた括弧で括られた式の結果を否定します。

命令実行時には、CPU は現在の論理演算結果(RLO)をバイナリ結果 BR とともに保存します。 括弧で括られた式内の命令が実行された後、求められた括弧で括られた式の RLO が否定され、保存されている RLO と排他的論理和演算されます。 括弧で括られた式の後に追加の信号問い合わせが続く場合は、これも排他的論理和演算されます。

括弧内の式の中のさらに括弧に括られた式内に命令をプログラムすることで、括弧で括られた式をさらにネストすることが可能です。これによって、ネストレベルは 7 レベルに制約されます。

構文

「分岐付きの否定排他的論理和演算」命令には、以下の構文を使用します。

```
XN (
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
	// ネスト化された式の開始
XN(	// 括弧で括られた式の RLO を排他的論理和演算し、求められた結果を否定する
A "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
A "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
)	// ネスト化された式の終了
XN "Tag_Input_3"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
= "Tag_Output"	// RLO のシグナル状態をオペランドに割り当て。

): 分岐を閉じる

説明

「分岐クローズ」命令を使用して、括弧で括られた式の終わりを指定します。 この命令の処理後に、ネスト化された式の結果全体が、ネスト化された式を開くために使用した命令に従ってリンクされます。

ネスト化された式を開くには、次の命令を使用します。

- A(: 分岐付きの論理積演算
- AN(: 分岐付きの否定論理積演算
- O(: 分岐付きの論理和演算
- ON(: 否定論理和演算
- X(: 分岐付きの排他的論理和演算
- XN(: 分岐付きの否定排他的論理和演算

「分岐クローズ」命令の後に、CPU はバイナリ結果 BR を括弧で括られた式の前のシグナル状態に設定します。

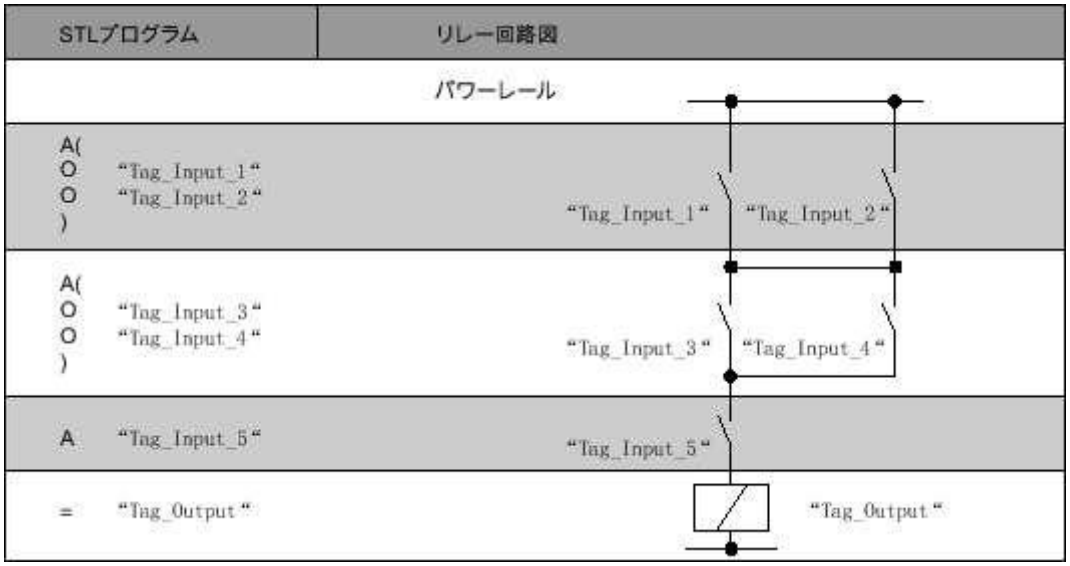
構文

「分岐を閉じる」命令には、以下の構文を使用します。

```
)
```

例

次の例で、命令がどのように動作するかを示します。



=: 割り当て

説明

「割り当て」命令を使用して、CPU に保存された論理演算の結果(RLO)のシグナル状態を指定したオペランドに割り当てます。 RLO のシグナル状態が「1」の場合、オペランドはセットされます。 信号が「0」の場合、オペランドが「0」にリセットされます。

構文

「割り当て」命令には、以下の構文を使用します。

```
= <operand>
```

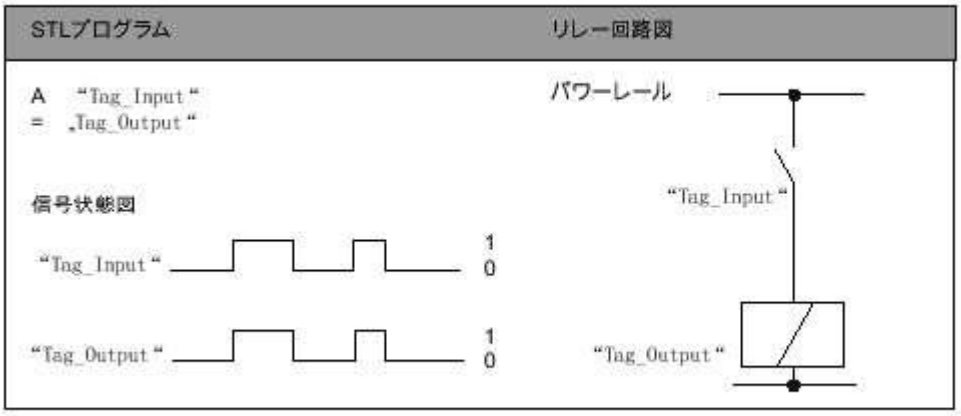
パラメータ

次の表に、「割り当て」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	現在の RLO が割り当てられるオペランド。

例

次の例で、命令がどのように動作するかを示します。



R: リセット

説明

「リセット」命令を使用して、指定したオペランドのシグナル状態を「0」にリセットします。

この命令は、論理演算の現在の結果(RLO)が「1」の場合にのみ実行されます。 命令の実行後、指定したオペランドが「0」にリセットされます。 現在の RLO が「0」の場合、指定されたオペランドのシグナル状態は変更されません。

構文

「リセット」命令には、以下の構文を使用します。

```
R <operand>
```

パラメータ

次の表に、「リセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	RLO = 「1」の場合にリセットされるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
A "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
R "Tag_Output"	// RLO が「1」の場合、オペランドを「0」にリセット

S: セット

説明

「セット」命令を使用して、指定したオペランドのシグナル状態を「1」にセットします。

この命令は、論理演算の現在の結果(RLO)が「1」の場合にのみ実行されます。 命令の実行後、指定されたオペランドは「1」にセットされます。 現在の RLO が「0」の場合、指定されたオペランドのシグナル状態は変更されません。

構文

「セット」命令には、以下の構文を使用します。

```
S <operand>
```

パラメータ

次の表に、「セット」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	RLO = 「1」の場合セットされるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
A "Tag_Input_2"	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
S "Tag_Output"	// RLO が「1」の場合、オペランドを「1」にセット

NOT: RLO の反転

説明

「RLO の反転」命令を使って、論理演算の結果(RLO)のシグナル状態を反転します。 「RLO の反転」命令は、論理演算内を含む任意の場所で使用できます。

構文

「RLO の反転」命令には、以下の構文を使用します。

```
NOT
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランド「Tag_Input_1」のシグナル状態「1」を照会し、現在の RLO で論理積演算。
A "Tag_Input_2"	// オペランド「Tag_Input_2」のシグナル状態「1」を照会し、現在の RLO で論理積演算。
A "Tag_Input_3"	// オペランド「Tag_Input_3」のシグナル状態「1」を照会し、現在の RLO で論理積演算。
NOT	// RLO のシグナル状態を反転
= "Tag_Output"	// RLO のシグナル状態をオペランド「Tag_Output」に割り当て。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Input_1	1	1	1	0	0	0	1	0
Tag_Input_2	0	1	1	0	1	1	0	1
Tag_Input_3	0	0	1	0	0	1	1	1
Tag_Output	1	1	0	1	1	1	1	1

SET: RLO を「1」にセット

説明

「RLO を 1 にセット」命令を使用して、現在の演算結果(RLO)をシグナル状態「1」にセットします。

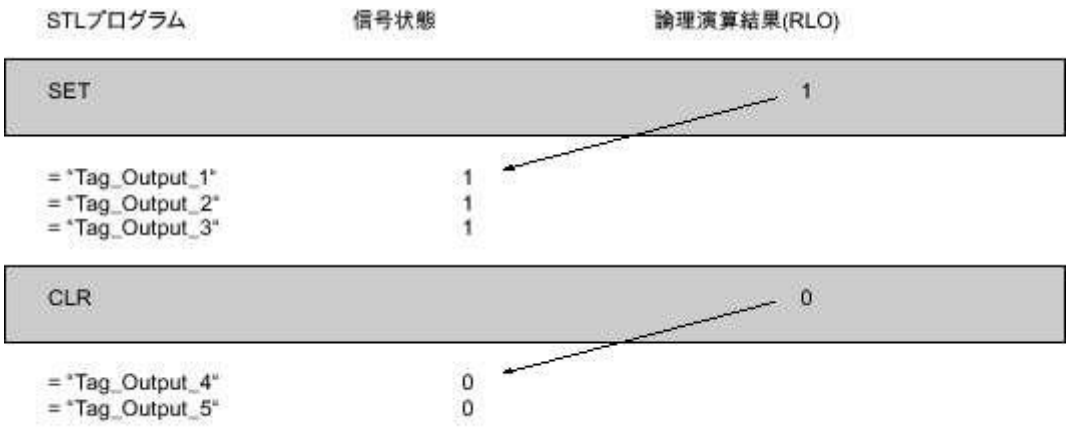
構文

「RLO を「1」にセット」命令には、以下の構文を使用します。

```
SET
```

例

次の例で、命令がどのように動作するかを示します。



CLR: RLO を「0」にリセット

説明

「RLO を 0 にリセット」命令を使用して、現在の演算結果(RLO)をシグナル状態「0」にセットします。

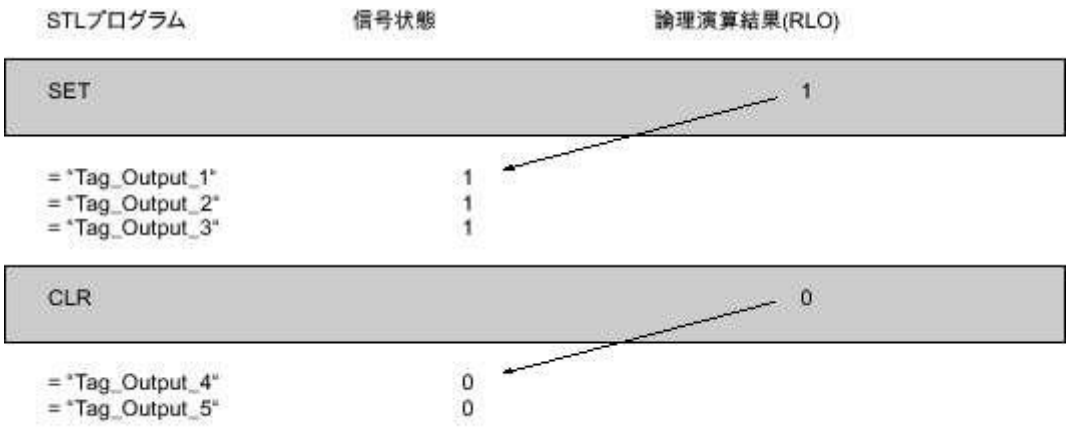
構文

「RLO を「0」にリセット」命令には、以下の構文を使用します。

```
CLR
```

例

次の例で、命令がどのように動作するかを示します。



SAVE: BR ビットに RLO を保存

説明

「BR ビットに RLO を保存」命令を使用し、論理演算の結果(RLO)をバイナリ結果(BR)に保存します。実行時、この命令は現在の論理演算の結果のシグナル状態をステータスビット BR に転送します。 この命令は条件に依存せずに実行され、他のいずれのステータスビットにも影響しません。

注記
「BR ビットに RLO を保存」命令が実行された後、BR ビットは同じまたは下位レベルのブロックにある以下の命令によって再び変更することができます。

「BR ビットに RLO を保存」命令を使用して、ブロックの実行ステータスをチェックします。 たとえばブロックの最後に「BR ビットに RLO を保存」命令を使用すると、BR ビットがブロックの現在の論理演算の結果のシグナル状態に設定されます。

BR ビットは、ブロック呼び出しに EN/ENO メカニズムを実装することに役立ちます。このトピックの詳細については、「関連項目」を参照してください。

構文

「BR ビットに RLO を保存」命令には、以下の構文を使用します。

```
SAVE
```

例

次の例で、命令がどのように動作するかを示します。

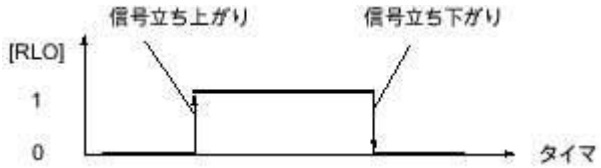
STL	説明
A "Tag_Input"	// オペランド「Tag_Input」のシグナル状態「1」で照会。
SAVE	// RLO のシグナル状態を BR ビットに転送。
BEC	// RLO が「1」の場合、ブロック終了。

FN: 立ち下がりエッジの RLO スキャン

説明

「立ち下がりエッジの RLO スキャン」命令を使用して、論理演算の結果(RLO)のシグナル状態での「1」から「0」への切り替えを照会します。 この命令は、命令のオペランドに保存された前の照会のシグナル状態と、論理演算の結果(RLO)の現在のシグナル状態を比較します。 命令が RLO の「1」から「0」への切り替えを検出すると、信号立ち下がりエッジが存在します。

次の図に、信号立ち下がりエッジおよび信号立ち上がりエッジの場合の RLO の切り替えを示します。



命令が実行されるたびに信号立ち下がりエッジが照会されます。 命令が信号立ち下がりエッジを検出すると、エッジ評価後に RLO がシグナル状態「1」にセットされます。 信号立ち下がりエッジが検出されない場合、論理演算の結果のシグナル状態は「0」です。

注記
エッジ評価のオペランドは、ビットメモリまたはデータブロックに存在する必要があります。

構文

「立ち下がりエッジの RLO スキャン」命令には、以下の構文を使用します。

```
FN <operand>
```

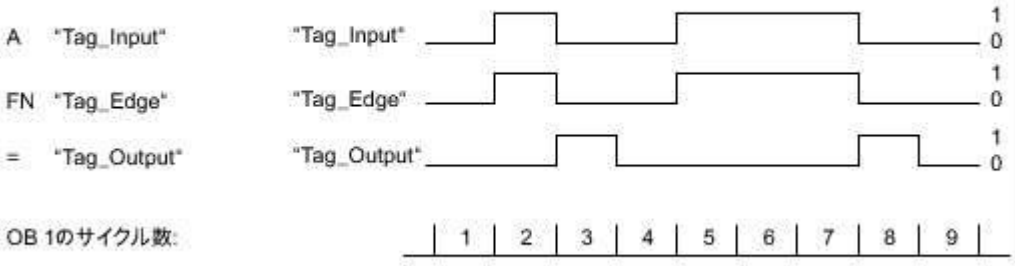
パラメータ

次の表に、「立ち下がりエッジの RLO スキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	前回の問い合わせの RLO が保存されているエッジメモリビット

例

次の例で、命令がどのように動作するかを示します。

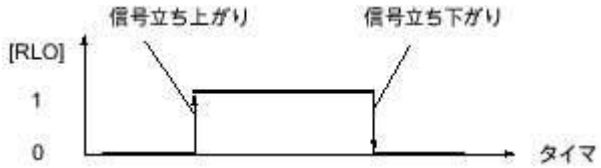
STLプログラム	信号状態図
A "Tag_Input" FN "Tag_Edge" = "Tag_Output" OB 1のサイクル数:	

FP: 立ち上がりエッジの RLO スキャン

説明

「立ち上がりエッジの RLO スキャン」命令を使用して、論理演算の結果(RLO)のシグナル状態での「0」から「1」への切り替えを照会します。 この命令は、命令のオペランドに保存された前の照会のシグナル状態と、論理演算の結果(RLO)の現在のシグナル状態を比較します。 命令が RLO の「0」から「1」への切り替えを検出すると、信号立ち上がりエッジが存在します。

次の図に、信号立ち下がりエッジおよび信号立ち上がりエッジの場合の RLO の切り替えを示します。



命令が実行されるたびに、信号立ち上がりエッジが照会されます。 命令が信号立ち上がりエッジを検出すると、エッジ評価後に RLO がシグナル状態「1」にセットされます。 信号立ち上がりエッジが検出されない場合、論理演算の結果のシグナル状態は「0」です。

注記
エッジ評価のオペランドは、ビットメモリまたはデータブロックに存在する必要があります。

構文

「立ち上がりエッジの RLO スキャン」命令には、以下の構文を使用します。

```
FP <operand>
```

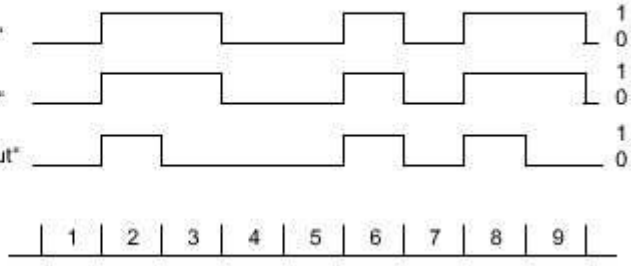
パラメータ

次の表に、「立ち上がりエッジの RLO スキャン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BOOL	I、Q、M、D、L	前回の問い合わせの RLO が保存されているエッジメモリビット

例

次の例で、命令がどのように動作するかを示します。

STLプログラム	信号状態図
A "Tag_Input" FP "Tag_Edge" = "Tag_Output" OB 1のサイクル数:	

タイマの動作

この章には下記に関する情報が記載されています：

- FR: タイマの有効化 (S7-1500)
- L: タイマ値のロード (S7-1500)
- LC: BCD コードタイマ値のロード (S7-1500)
- R: タイマのリセット (S7-1500)
- SP: パルスタイマの起動 (S7-1500)
- SE: 拡張パルスタイマの起動 (S7-1500)
- SD: オンディレータイマの起動 (S7-1500)
- SS: 拡張オンディレータイマの起動 (S7-1500)
- SF: オフディレータイマの起動 (S7-1500)

FR: タイマの有効化

説明

命令「タイマの有効化」を使用して、タイマを再開できます。 この命令は立ち上がり エッジによって実行され、タイマの開始のため内部エッジトリガフラグをリセットします。 命令の実行中タイマを論理演算の結果「1」によって開始するため、タイマは開始命令前に信号の立ち上がり エッジが存在しなくても再起動します。 開始命令前に論理演算の結果が「0」の場合、内部エッジトリガフラグをリセットしてもタイマに影響しません。

命令の前に論理演算の結果が「0」に切り替わると、「タイマの有効化」命令はタイマの再起動後に影響しません。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。関連項目「L: タイマ値のロード」。

構文

「タイマの有効化」命令には、以下の構文を使用します。

```
FR <Timer>
```

パラメータ

次の表に、「タイマの有効化」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	有効になったタイマ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_EnableInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
FR "MyTimer"	// 信号エッジが立ち上がりの場合、タイマの有効化。
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)

```
SP "MyTimer"          // オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、パルスタイマを開始。
A "Tag_Reset"         // オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"           // RLO = 「1」の場合、タイマをリセット
A "MyTimer"           // タイマのステータスを「1」で照会。
= "Tag_TimerStatus"   // タイマのステータスをオペランドに割り当て。
L "MyTimer"           // タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"    // 現在の時間値をオペランド「Tag_TimerValue」に転送。
```

L: タイマ値のロード

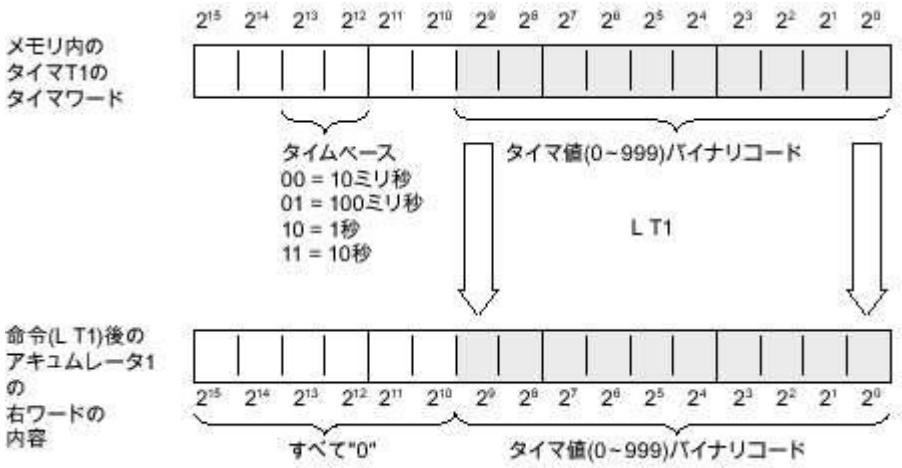
説明

「タイマ値のロード」命令は指定されたタイマの時間値をアキュムレータ 1 にロードします。二進化されたタイマ値はタイムベースなしにロードされます。 タイムベースの代わりにアキュムレータ 1 にゼロが書き込まれます。

ロード後、アキュムレータ 1 の値はデータタイプ INT の正の数に対応します。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

次の図に、時間値がアキュムレータにロードされる方法の例を示します。



構文

「タイマ値のロード」命令には、以下の構文を使用します。

```
L <Timer>
```

パラメータ

次の表に、「タイマ値のロード」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	現在の時間値がロードされるタイマ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_EnableInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
FR "MyTimer"	// エッジが立ち上がりの場合、タイマの有効化。
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SP "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、パルスとしてタイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセット
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
L "MyTimer"	// タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。

LC: BCD コードタイマ値のロード

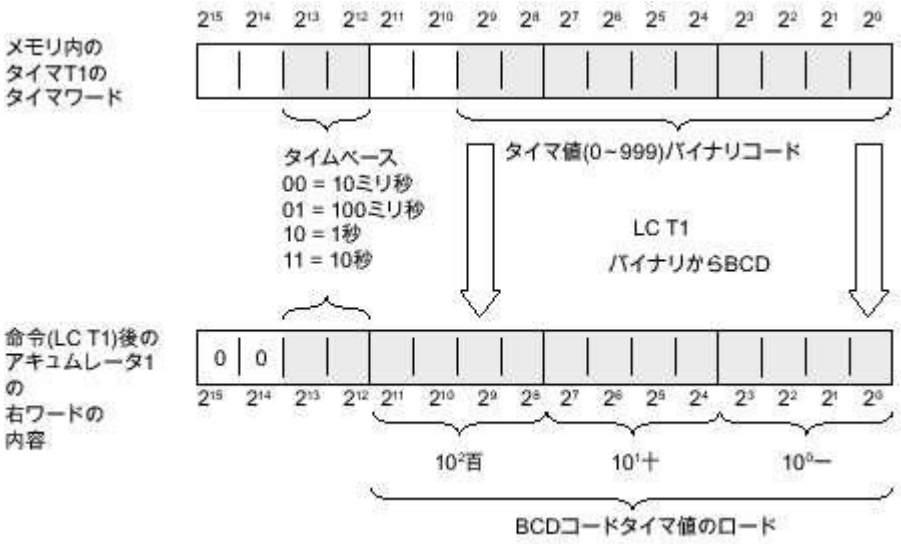
説明

「BCD コードタイマ値のロード」命令は指定されたタイマの二進化されたタイマ値を BCD フォーマットでアキュムレータ 1 にロードします。タイムベースはロード中にアキュムレータ 1 に転送されます。

ロード後、アキュムレータ 1 の値は S5TIME 形式の持続時間に対応します。 アキュムレータ 1 の左ワードはゼロで埋められます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

次の図に、時間値がアキュムレータにロードされる方法を示します。



構文

「BCD コードタイマ値のロード」命令には、以下の構文を使用します。

```
LC <Timer>
```

パラメータ

次の表に、「BCD コードタイマ値のロード」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	現在の BCD コード化されたタイマ値がロードされるタイマ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_EnableInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
FR "MyTimer"	// エッジが立ち上がりの場合、タイマの有効化。
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SP "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、パルスとしてタイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセット
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
LC "MyTimer"	// タイマの現在の BCD コード化されたタイマ値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。

R: タイマのリセット

説明

「タイマのリセット」命令を使用して、指定したタイマを「0」にリセットすることができます。 この命令は、論理演算の現在の結果(RLO)が「1」の場合に実行されます。 命令の前に RLO が「1」である限り、指定されたタイマは値「0」を返します。 この命令は、時間値およびプログラムされた持続時間のタイムベースも「0」にリセットします。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

注記
「タイマのリセット」命令は、内部エッジメモリビットをリセットしません。 内部エッジメモリビットをリセットするには、「タイマの有効化」命令をプログラムするか、シグナル状態「0」で時間を開始する命令を実行します。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「タイマのリセット」命令には、以下の構文を使用します。

```
R <Timer>
```

パラメータ

次の表に、「タイマのリセット」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	リセットされるタイマ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセット
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。

```
L "MyTimer"                 // タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"          // 現在の時間値をオペランド「Tag_TimerValue」に転送。
```

SP: パルスタイマの起動

説明

「パルスタイマの起動」命令を使用して、指定したタイマを信号立ち上がりエッジで起動することができます。 信号の立ち上がりエッジが検出されると、命令が実行されてタイマが起動します。 命令前の論理演算の結果(RLO)が「1」のままである限り、タイマはアキュムレータ 1 で指定された持続時間によって動作停止します。 期限が切れる前に RLO が「0」に切り替わると、タイマが停止します。 持続時間が実行中である限り、「1」のタイマステータスが照会されると照会結果「1」が返されます。

タイマは「パルスタイマの起動」命令の前の信号立ち上がりエッジ、または「タイマの有効化」命令で再開することができます。 この再起動はタイマがリセットされない場合に行うことができます。

アキュムレータ 1 の持続時間は、時間値とタイムベースで構成されています。 指定したタイマが「パルスタイマの起動」命令によって起動する場合、時間値がタイムベースに応じてカウントダウンされます。 タイマは、カウンタ値がゼロにカウントダウンされるまで動作します。

S5TIME 形式の有効な時間は、命令の実行中にタイマが 開始していなくてもアキュムレータ 1 で示す必要があります。 アキュムレータ 1 に有効な BCD 値が含まれず、直前の論理演算の RLO が「0」を返す場合、ステータスビット OV は「1」にセットされます。 同期エラー処理は開始されません。 アキュムレータ 1 が有効な BCD 値を含む場合、ステータスビット OV が「0」にリセットされます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「パルスタイマの起動」命令には、以下の構文を使用します。

```
SP <Timer>
```

パラメータ

次の表に、「パルスタイマの起動」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	パルスとして起動するタイマ。

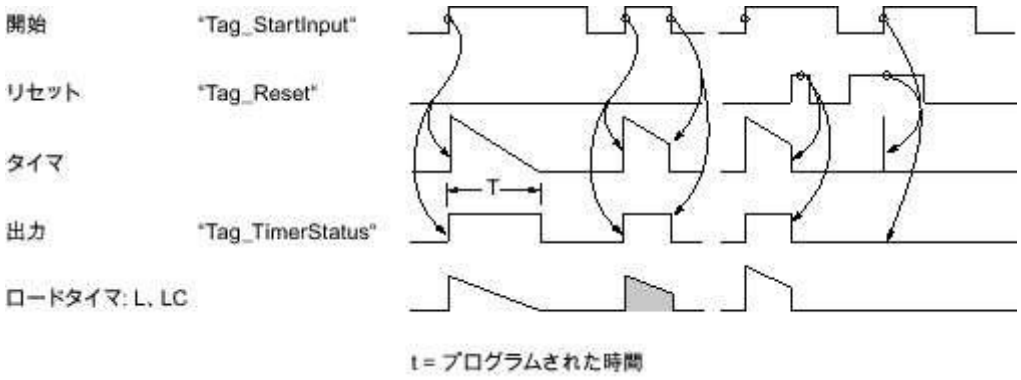
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SP "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、パルスタイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセットし時間計測を停止
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
L "MyTimer"	// タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。
LC "MyTimer"	// 現在の時間値を BCD フォーマットでアキュムレータ 1 にロード。
T "Tag_TimerValue_BCD"	// 時間値を BCD フォーマットでオペランド「Tag_TimerValue_BCD」に転送

次の図に、タイマ図の例を示します。



SE: 拡張パルスタイマの起動

説明

「拡張パルスタイマの起動」命令を使用して、指定したタイマを信号立ち上がり エッジで起動することができます。 信号の立ち上がり エッジが検出されると、命令が実行されてタイマが起動します。 命令前の論理演算の結果が「0」に切り替わる限り、タイマはアキュムレータ 1 で指定された持続時間によって動作停止します。 持続時間が実行中である限り、「1」のタイマステータスが照会されると照会結果「1」が返されます。

信号の立ち上がり エッジごとに、命令はタイマがまだ動作停止していなくてもタイマをプログラムされた持続時間によって再起動します。

アキュムレータ 1 の持続時間は、時間値とタイムベースで構成されています。 指定したタイマが「拡張パルスタイマの起動」命令によって起動する場合、時間値がタイムベースに応じてカウントダウンされます。 タイマは、カウンタ値がゼロにカウントダウンされるまで動作します。

S5TIME 形式の有効な時間は、命令の実行中にタイマが 開始していなくてもアキュムレータ 1 で示す必要があります。アキュムレータ 1 に有効な BCD 値が含まれず、直前の論理演算の RLO が「0」を返す場合、ステータスビット OV は「1」にセットされます。同期エラー処理は開始されません。アキュムレータ 1 が有効な BCD 値を含む場合、ステータスビット OV が「0」にリセットされます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。このデクリメントは、ユーザープログラムに対して非同期的に行われます。このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。関連項目「L: タイマ値のロード」。

構文

「拡張パルスタイマの起動」命令には、以下の構文を使用します。

```
SE <Timer>
```

パラメータ

次の表に、「拡張パルスタイマの起動」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	拡張パルスとして起動するタイマ

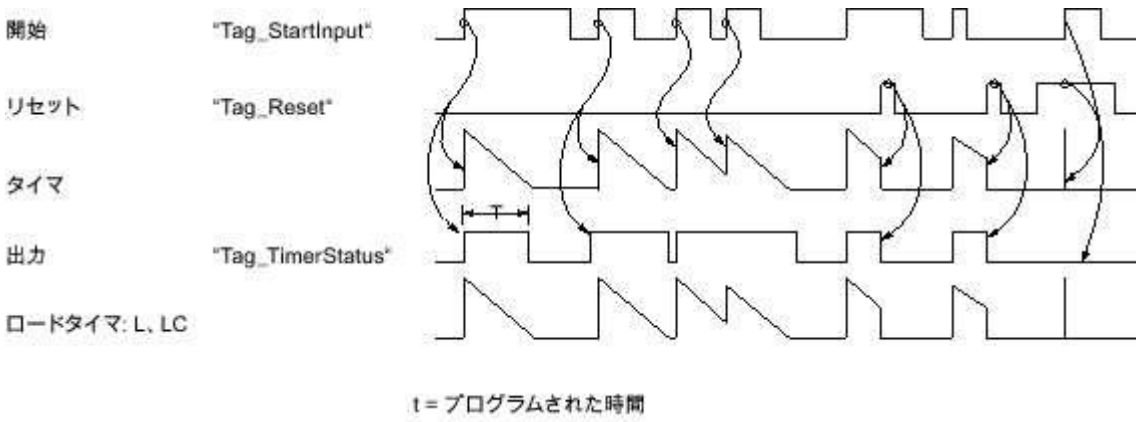
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SE "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、拡張パルスとしてタイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセットし時間計測を停止
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
L "MyTimer"	// タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。
LC "MyTimer"	// 現在の時間値を BCD フォーマットでアキュムレータ 1 にロード。
T "Tag_TimerValue_BCD"	// 時間値を BCD フォーマットでオペランド「Tag_TimerValue_BCD」に転送

次の図に、タイマ図の例を示します。



SD: オンディレータイマの起動

説明

「オンディレータイマの起動」命令を使用して、指定したタイマを信号立ち上がりエッジで起動することができます。 信号の立ち上がりエッジが検出されると、命令が実行されてタイマが起動します。 命令前の論理演算の結果(RLO)が「1」のままである限り、タイマはアキュムレータ 1 で指定された持続時間によって動作停止します。 持続時間の終了前に論理演算の結果が「0」に切り替わると、タイマは停止します。

時間が正常に経過し、命令の前の現在の RLO が「1」の場合、タイマステータス「1」で照会すると照会結果「1」が返されます。 時間が期限切れになっていない場合、または RLO が「0」に切り替わる場合、タイマステータス「1」の照会は照会結果「0」を返します。

アキュムレータ 1 の持続時間は、時間値とタイムベースで構成されています。 指定したタイマが「オンディレータイマの起動」命令によって起動する場合、時間値がタイムベースに応じてカウントダウンされます。 タイマは、カウンタ値がゼロにカウントダウンされるまで動作します。

S5TIME 形式の有効な時間は、命令の実行中にタイマが 開始していなくてもアキュムレータ 1 で示す必要があります。アキュムレータ 1 に有効な BCD 値が含まれず、直前の論理演算の RLO が「0」を返す場合、ステータスビット OV は「1」にセットされます。同期エラー処理は開始されません。アキュムレータ 1 が有効な BCD 値を含む場合、ステータスビット OV が「0」にリセットされます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。このデクリメントは、ユーザープログラムに対して非同期的に行われます。このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。関連項目「L: タイマ値のロード」。

構文

「オンディレータイマの起動」命令には、以下の構文を使用します。

```
SD <Timer>
```

パラメータ

次の表に、「オンディレータイマの起動」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	オンディレーとして起動するタイマ

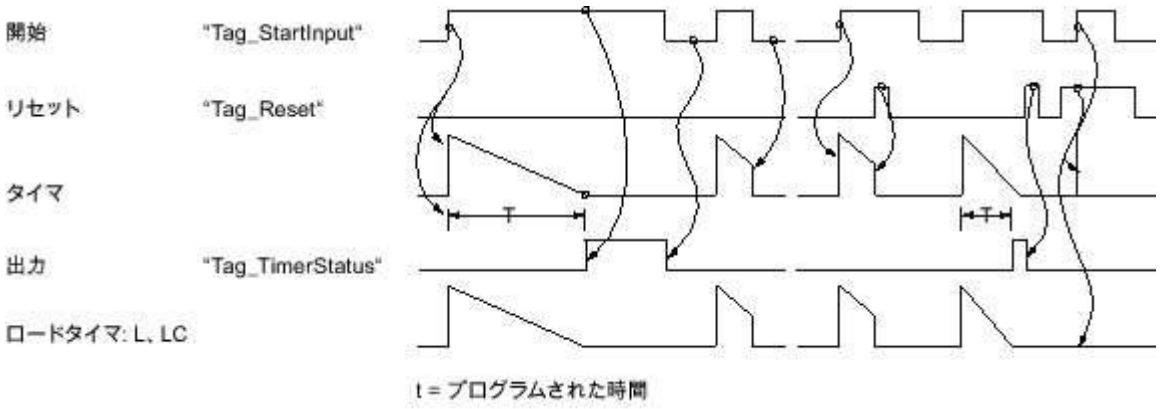
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SD "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、オンディレーとしてタイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセットし時間計測を停止
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
L "MyTimer"	// タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。
LC "MyTimer"	// 現在の時間値を BCD フォーマットでアキュムレータ 1 にロード。
T "Tag_TimerValue_BCD"	// 時間値を BCD フォーマットでオペランド「Tag_TimerValue_BCD」に転送

次の図に、タイマ図の例を示します。



SS: 拡張オンディレータイマの起動

説明

「保持型オンディレータイマの起動」命令を使用して、指定したタイマを信号立ち上がり工ッジで起動することができます。 信号の立ち上がり工ッジが検出されると、命令が実行されてタイマが起動します。 命令前の論理演算の結果(RLO)が「0」に切り替わっても、タイマはアキュムレータ 1 で指定された持続時間で動作します。 時間が期限切れになると、タイムステータス「1」の照会は照会結果「1」を返します。 これによって、照会結果は命令前の現在の論理演算の結果のシグナル状態では影響されません。

信号の立ち上がり工ッジごとに、命令はタイマがまだ動作停止していなくてもタイマをプログラムされた持続時間によって再起動します。

アキュムレータ 1 の持続時間は、時間値とタイムベースで構成されています。 指定したタイマが「保持型オンディレータイマの起動」命令によって起動する場合、時間値がタイムベースに応じてカウントダウンされます。 タイマは、カウンタ値がゼロにカウントダウンされるまで動作します。

S5TIME 形式の有効な時間は、命令の実行中にタイマが 開始していなくてもアキュムレータ 1 で示す必要があります。アキュムレータ 1 に有効な BCD 値が含まれず、直前の論理演算の RLO が「0」を返す場合、ステータスビット OV は「1」にセットされます。同期工ラー処理は開始されません。アキュムレータ 1 が有効な BCD 値を含む場合、ステータスビット OV が「0」にリセットされます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。このデクリメントは、ユーザープログラムに対して非同期的に行われます。このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。関連項目「L: タイマ値のロード」。

構文

「保持型オンディレータイマの起動」命令には、以下の構文を使用します。

```
SS <Timer>
```

パラメータ

次の表に、「保持型オンディレータイマの起動」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	保持型オンディレーとして起動するタイマ

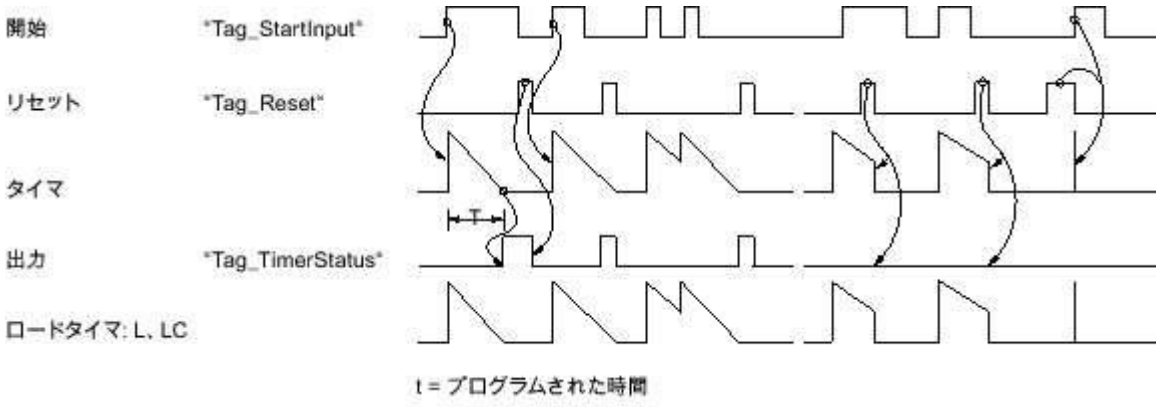
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SS "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がり工ッジがあり、かつオペランド「Tag_StartInput」のシグナル状態が「1」の場合、保持型オンディレータイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセットし時間計測を停止
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
L "MyTimer"	// タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。
LC "MyTimer"	// 現在の時間値を BCD フォーマットでアキュムレータ 1 にロード。
T "Tag_TimerValue_BCD"	// 時間値を BCD フォーマットでオペランド「Tag_TimerValue_BCD」に転送

次の図に、タイマ図の例を示します。



SF: オフディレータイマの起動

説明

「オフディレータイマの起動」命令を使用して、指定したタイマを信号立ち上がりエッジで起動することができます。 信号の立ち下がりエッジが検出されると、命令が実行されてタイマが起動します。 その後、タイマはアキュムレータ 1 で指定された持続時間によって動作停止します。持続時間の終了前に論理演算の結果(RLO)が「1」に切り替わると、タイマはリセットされます。 このタイマは、命令の処理中に信号立ち下がりエッジが検出された場合にのみ再起動されます。

命令実行時の論理演算の結果のシグナル状態が「1」の場合またはプログラムされた時間が実行中の場合、「1」のタイマステータスの照会は照会結果「1」を返します。 時間が経過しない場合または RLO が「0」の場合、タイマステータス「1」の照会は照会結果「0」を返します。

アキュムレータ 1 の持続時間は、時間値とタイムベースで構成されています。 指定したタイマが「オフディレータイマの起動」命令によって起動する場合、時間値がタイムベースに応じてカウントダウンされます。 タイマは、カウンタ値がゼロにカウントダウンされるまで動作します。

S5TIME 形式の有効な時間は、命令の実行中にタイマが 開始していなくてもアキュムレータ 1 で示す必要があります。アキュムレータ 1 に有効な BCD 値が含まれず、直前の論理演算の RLO が「0」を返す場合、ステータスビット OV は「1」にセットされます。同期エラー処理は開始されません。アキュムレータ 1 が有効な BCD 値を含む場合、ステータスビット OV が「0」にリセットされます。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。このデクリメントは、ユーザープログラムに対して非同期的に行われます。このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。関連項目「L: タイマ値のロード」。

構文

「オフディレータイマの起動」命令には、以下の構文を使用します。

```
SF <Timer>
```

パラメータ

次の表に、「オフディレータイマの起動」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<timer>	Input	TIMER	T	オフディレーとして起動するタイマ

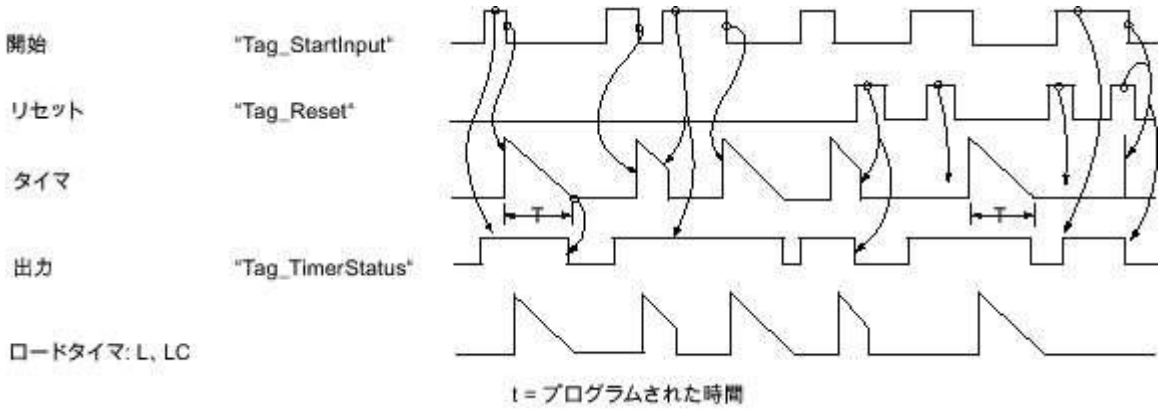
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_StartInput"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
L S5T#10s	// 持続時間のロード(10 秒)
SF "MyTimer"	// オペランド「Tag_StartInput」またはオペランド「Tag_EnableInput」に信号立ち上がりエッジがある場合、およびオペランド「Tag_StartInput」のシグナル状態が「1」の場合、オフディレーとしてタイマを開始。
A "Tag_Reset"	// オペランドを「1」で照会し、現在の RLO で AND 論理積演算。
R "MyTimer"	// RLO = 「1」の場合、タイマをリセットし時間計測を停止
A "MyTimer"	// タイマのステータスを「1」で照会。
= "Tag_TimerStatus"	// タイマのステータスをオペランドに割り当て。
L "MyTimer"	// タイマの現在の時間値をアキュムレータ 1 にロード。
T "Tag_TimerValue"	// 現在の時間値をオペランド「Tag_TimerValue」に転送。
LC "MyTimer"	// 現在の時間値を BCD フォーマットでアキュムレータ 1 にロード。
T "Tag_TimerValue_BCD"	// 時間値を BCD フォーマットでオペランド「Tag_TimerValue_BCD」に転送

次の図に、タイマ図の例を示します。



カウンタ演算

この章には下記に関する情報が記載されています：

- FR: カウンタの有効化 (S7-1500)
- L: カウンタのロード (S7-1500)
- LC: BCD カウンタ値のロード (S7-1500)
- R: カウンタのリセット (S7-1500)
- S: カウンタセット (S7-1500)
- CU: カウントアップ (S7-1500)
- CD: カウントダウン (S7-1500)

FR: カウンタの有効化

説明

「カウンタの有効化」命令を使用して、信号立ち上がりエッジがない場合でも「カウンタセット」、「カウントアップ」、および「カウントダウン」命令を実行します。 処理中、この命令は内部エッジフラグをシグナル状態「0」にリセットします。 対応するカウント命令前の照会は、引き続きシグナル状態「1」であることが必要です。

「カウンタの有効化」命令は、命令前に論理演算の結果(RLO)が「0」から「1」に切り替わると実行されます。

カウンタは実行対象のカウント命令のために有効にする必要はありません。

構文

「カウンタの有効化」命令には、以下の構文を使用します。

```
FR <operand>
```

パラメータ

次の表に、「カウンタの有効化」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	有効になるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// シグナル状態を「1」で照会し、現在の RLO で論理積演算。
FR "MyCounter"	// カウンタの有効化
A "Tag_Input_2"	// シグナル状態を「1」で照会し、現在の RLO で論理積演算。
CD "MyCounter"	// カウントダウン
A "MyCounter"	// カウンタステータスを照会
= "Tag_Output"	// 結果をオペランドに割り当てる。

この例でカウンタ(「MyCounter」)はオペランド「Tag_Input_1」のシグナル状態が「0」から「1」に切り替わると有効になる。 オペランド「Tag_Input_2」のシグナル状態が「1」の場合、カウンタ値はカウンタが有効になると 1 デクリメントされる。 アドレス「Tag_Input_2」のシグナル状態が「0」の場合、カウンタ値は変更なしのままです。 カウンタ値はカウンタの有効化に関わらず、アドレス「Tag_Input_2」のシグナル状態が「0」から「1」に切り替わった場合も 1 デクリメントされます。

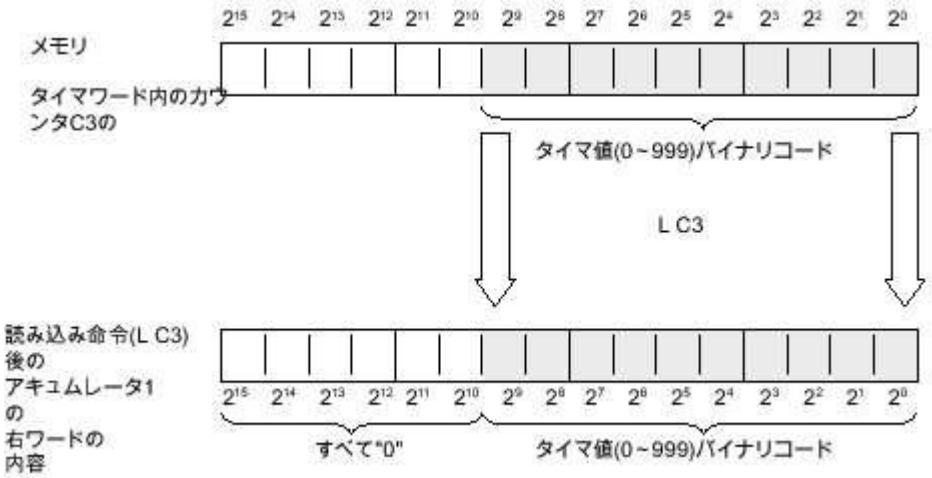
カウンタステータスの照会は、カウンタ読み取りがゼロよりも大きい場合結果「1」になります。 カウンタ読み取りがゼロに等しい場合、照会の結果は「0」です。

L: カウンタのロード

説明

「カウンタのロード」命令は、指定したバイナリコーディングされたカウンタの現在値をアキュムレータ 1 の右ワードに転送します。ロードされたカウンタ値は、以降の処理では 16 ビット整数として使用できます。

以下の図で、命令がどのように動作するかを示します。



構文

「カウンタのロード」命令には、以下の構文を使用します。

```
L <operand>
```

パラメータ

次の表に、「カウンタのロード」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	現在値がロードされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

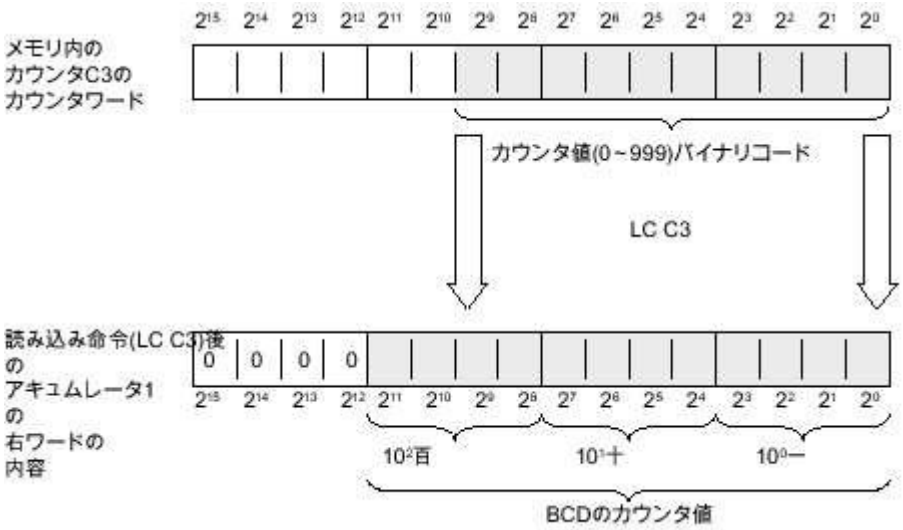
STL	説明
L "MyCounter"	// バイナリコーディングされたカウンタ現在値をアキュムレータ 1 にロード
L "Tag_Value"	// カウンタ値をアキュムレータ 2 に移動します。 // オペランド「Tag_Value」の値をアキュムレータ 1 にロード。
>=I	// カウンタ値がアドレス「Tag_Value」の値以上であるかを照会
= "Tag_Output"	// 結果をオペランドに割り当てる。

LC: BCD カウンタ値のロード

説明

「カウンタのロード」命令は、指定したカウンタの現在値を BCD としてアキュムレータ 1 の右側ワードに転送します。ロードされたカウンタ値は、以降の処理では BCD 形式で使用できます。

以下の図で、「BCD カウンタ値のロード」命令がどのように動作するかを示します。



構文

「BCD カウンタ値のロード」命令には、以下の構文を使用します。

```
LC <operand>
```

パラメータ

次の表に、「BCD カウンタ値のロード」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	現在値がロードされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LC "MyCounter"	// 現在の BCD コーディングされたカウンタ値をアキュムレータ 1 にロード。
T "Tag_Output"	// カウンタ現在値をアドレス「Tag_Output」に転送し保存。

R: カウンタのリセット

説明

「カウンタのリセット」命令を使用して、指定したカウンタのカウンタ現在値を「0」にリセットします。

この命令は、命令前の論理演算の結果(RLO)のシグナル状態が「1」の場合にのみ実行されます。「カウンタのリセット」命令前の RLO が「1」である限り、カウンタステータス「1」の照会は結果「0」を返します。

カウンタがリセットされるとき、「カウンタセット」、「カウントアップ」および「カウントダウン」命令の内部エッジメモリビットはリセットされません。

構文

「カウンタのリセット」命令には、以下の構文を使用します。

```
R <operand>
```

パラメータ

次の表に、「カウンタのリセット」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	カウンタ値が「0」にリセットされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input"	// シグナル状態を「1」で照会し、現在の RLO で論理積演算。
R "MyCounter"	// カウンタのリセット

S: カウンタセット

説明

「カウンタセット」命令を使用し、指定されたカウンタを照会の時点でアキュムレータ 1 に含まれる値にセットします。

「カウンタセット」命令は、命令前に論理演算の結果が「0」から「1」に切り替わると実行されます。信号立ち上がりエッジは、内部エッジメモリビットを「0」にリセットする「カウンタの有効化」命令でも発生します。

カウンタをセットするには、アキュムレータ 1 に正しいカウンタ値が含まれている必要があります。有効なカウンタ値は 3 桁の BCD フォーマット値です。 正のカウンタ値のみが許可され、カウント命令は負のカウンタ値を処理できません。 アキュムレータ 1 に有効な BCD 値が存在する場合、ステータスビット OV のシグナル状態は「0」になります。 有効な BCD 値が存在しない場合、ステータスビット OV のシグナル状態は「1」になります。 同期エラー処理は開始されません。

構文

「カウンタセット」命令には、以下の構文を使用します。

```
S <operand>
```

パラメータ

次の表に、「カウンタセット」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	セットされるカウンタ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input"	// シグナル状態を「1」で照会し、現在の RLO で論理積演算。
L "Tag_CountValue"	// カウンタ値をアキュムレータ 1 にロード。
S "MyCounter"	// カウンタをアキュムレータ 1 の値にセット

CU: カウントアップ

説明

「カウントアップ」命令を使用して、命令が「0」から「1」に切り替わる前に論理演算の結果(RLO)のカウンタ現在値を 1 インクリメントします。 カウンタ値は、上限値「999」に達するまで立ち上がりシグナルエッジが発生するたびに引き続きインクリメントされます。 限界値に達すると、命令前の RLO はカウンタ値に影響を与えなくなります。

「カウントアップ」命令の実行には、常に立ち上がりシグナルエッジが必要です。 信号立ち上がりエッジは、内部エッジメモリビットを「0」にリセットする「カウンタの有効化」命令でも発生します。

構文

「カウントアップ」命令には、以下の構文を使用します。

```
CU <operand>
```

パラメータ

次の表に、「カウントアップ」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	立ち上がりエッジで 1 インクリメントされるカウンタ値のカウンタ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input"	// シグナル状態を「1」で照会し、現在の RLO で論理積演算。
CU "MyCounter"	// 立ち上がりエッジでカウンタ値を 1 インクリメント。

CD: カウントダウン

説明

「カウントダウン」命令を使用して、命令が「0」から「1」に切り替わる前に論理演算の結果(RLO)のカウンタ現在値を 1 デクリメントします。 カウンタ値は、下限値「0」に達するまで立ち上がりシグナルエッジが発生するたびに、引き続きデクリメントされます。 限界値に達すると、命令前の RLO はカウンタ値に影響を与えなくなります。 カウンタ値が負の場合、カウントは発生しません。

「カウントダウン」命令の実行には、常に立ち上がりシグナルエッジが必要です。 信号立ち上がりエッジは、内部エッジメモリビットを「0」にリセットする「カウンタの有効化」命令でも発生します。

構文

「カウントダウン」命令には、以下の構文を使用します。

```
CD <operand>
```

パラメータ

次の表に、「カウントダウン」命令のパラメータを示します。

オペランド	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	COUNTER	C	立ち上がりエッジで 1 デクリメントされるカウンタ値のカウンタ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input"	// シグナル状態を「1」で照会し、現在の RLO で論理積演算。
CD "MyCounter"	// 立ち上がりエッジでカウンタ値を 1 デクリメント。

比較演算

この章には下記に関する情報が記載されています：

- ? I: 16 ビット整数比較 (S7-1500)
- ? D: 32 ビット整数比較 (S7-1500)
- ? R: 浮動小数点数比較 (S7-1500)

? I: 16 ビット整数比較

説明

「16 ビット整数比較命令」を使用して、アキュムレータ 2 の右ワードの内容をアキュムレータ 1 の右ワードの内容と比較します。アキュムレータ 1 および 2 の内容は、16 ビット整数として解釈されます。 この命令は、指定された比較ファンクションに従って比較を実行します。

次の表に、使用可能な比較ファンクションとその意味を示します。

比較ファンクション	意味
==I	16 ビット整数比較「等しい」
<>I	16 ビット整数比較「等しくない」
>I	16 ビット整数比較「超過」
<I	16 ビット整数比較「less than」
>=I	16 ビット整数比較「以上」
<=I	16 ビット整数比較「less than or equal to」

比較結果は、論理演算の結果(RLO)、およびステータスビット CC 1 と CC 0 に影響を与えます。 比較条件が満たされている場合、RLO がシグナル状態「1」にセットされます。 比較条件が満たされていない場合、RLO はシグナル状態「0」にセットされます。

次の表に、各比較結果が RLO にどのように影響するかを示します。

比較ファンクション	ACCU 2 > ACCU 1 の場合の RLO	ACCU 2 = ACCU 1 の場合の RLO	ACCU 2 < ACCU 1 の場合の RLO
==I	0	1	0
<>I	1	0	1
>I	1	0	0
<I	0	0	1
>=I	1	1	0
<=I	0	1	1

ステータスビット CC 1 および CC 0 の設定は、比較に関わる 2 つの値の関係によって異なります。

アキュムレータ 1 および 2 の内容は、比較ファンクションの実行によって変更されません。 比較ファンクションの処理後に、論理演算の結果を割り当てファンクションおよびジャンプファンクションで評価することができます。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// オペランド「Tag_Input_1」の値をアキュムレータ 1 にロード

```
                                  // アキュムレータ 1 の内容(Tag_Input_1)をアキュムレータ 2 へシフト。
L "Tag_Input_2"
                                  // オペランド「Tag_Input_2」の値をアキュムレータ 1 にロード
>I                                // オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかを比較。
= "Tag_Output"                    // 比較結果をオペランド「Tag_Output」に割り当て。
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	**値**		
Tag_Input_1	10	10	5
Tag_Input_2	5	10	10
Tag_Output	1	0	0

? D: 32 ビット整数比較

説明

「32 ビット整数比較」命令を使用して、アキュムレータ 2(ACCU 2)の内容をアキュムレータ 1(ACCU 1)の内容と比較します。 アキュムレータ 1 および 2 の内容は、32 ビット整数として解釈されます。この命令は、指定された比較ファンクションに従って比較を実行します。

次の表に、使用可能な比較ファンクションとその意味を示します。

比較ファンクション	意味
==D	32 ビット整数比較「等しい」
<>D	32 ビット整数比較「等しくない」
>D	32 ビット整数比較「超過」
<D	32 ビット整数比較「less than」
>=D	32 ビット整数比較「以上」
<=D	32 ビット整数比較「less than or equal to」

比較結果は、論理演算の結果(RLO)、およびステータスビット CC 1 と CC 0 に影響を与えます。 比較条件が満たされている場合、RLO がシグナル状態「1」にセットされます。 比較条件が満たされていない場合、RLO がシグナル状態「0」にリセットされます。

次の表に、各比較結果が RLO にどのように影響するかを示します。

比較ファンクション	ACCU 2 > ACCU 1 の場合の RLO	ACCU 2 = ACCU 1 の場合の RLO	ACCU 2 < ACCU 1 の場合の RLO
==D	0	1	0
<>D	1	0	1
>D	1	0	0
<D	0	0	1
>=D	1	1	0
<=D	0	1	1

ステータスビット CC 1 および CC 0 の設定は、比較に関わる 2 つの値の関係によって異なります。

アキュムレータ 1 および 2 の内容は、比較ファンクションの実行によって変更されません。 比較ファンクションの処理後に、論理演算の結果を割り当てファンクションおよびジャンプファンクションで評価することができます。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// オペランド「Tag_Input_1」の値をアキュムレータ 1 にロード。

```
                                  // アキュムレータ 1 の内容(Tag_Input_1)をアキュムレータ 2 へシフト。
L "Tag_Input_2"
                                  // オペランド「Tag_Input_2」の値をアキュムレータ 1 にロード
>D                                // オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかを比較。
= "Tag_Output"                    // 比較結果をオペランド「Tag_Output」に割り当て。
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	**値**		
Tag_Input_1	150 000	150 000	100 000
Tag_Input_2	100 000	150 000	150 000
Tag_Output	1	0	0

? R: 浮動小数点数比較

説明

「浮動小数点数比較」命令を使用して、アキュムレータ 2(ACCU 2)の内容をアキュムレータ 1(ACCU 1)の内容と比較します。 アキュムレータ 1 および 2 の内容は、浮動小数点数として解釈されます。 比較を行うには、アキュムレータ 1 および 2 に有効な浮動小数点数が入っている必要があります。 この命令は、指定された比較ファンクションに従って比較を実行します。

次の表に、使用可能な比較ファンクションとその意味を示します。

比較ファンクション	意味
==R	浮動小数点数比較「等しい」
<>R	浮動小数点数比較「等しくない」
>R	浮動小数点比較「超過」
<R	浮動小数点数比較「less than」
>=R	浮動小数点数比較「以上」
<=R	浮動小数点数比較「less than or equal to」

比較結果は、論理演算の結果(RLO)、およびステータスビット CC 1 と CC 0 に影響を与えます。 比較条件が満たされている場合、RLO がシグナル状態「1」にセットされます。 比較条件が満たされていない場合、RLO がシグナル状態「0」にリセットされます。

次の表に、各比較結果が RLO にどのように影響するかを示します。

比較ファンクション	ACCU 2 > ACCU 1 の場合の RLO	ACCU 2 = ACCU 1 の場合の RLO	ACCU 2 < ACCU 1 の場合の RLO
==R	0	1	0
<>R	1	0	1
>R	1	0	0
<R	0	0	1
>=R	1	1	0
<=R	0	1	1

ステータスビット CC 1 および CC 0 の設定は、比較に関わる 2 つの値の関係によって異なります。

注記
無効な浮動小数点数の比較中、CPU は論理演算の結果を「0」に、ステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

アキュムレータ 1 および 2 の内容は、比較ファンクションの実行によって変更されません。 比較ファンクションの処理後に、論理演算の結果を割り当てファンクションおよびジャンプファンクションで評価することができます。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// オペランド「Tag_Input_1」の値をアキュムレータ 1 にロード。
L "Tag_Input_2"	// アキュムレータ 1 の内容(Tag_Input_1)をアキュムレータ 2 へシフト。 // オペランド「Tag_Input_2」の値をアキュムレータ 1 にロード
>R	// オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかを比較。
= "Tag_Output"	// 比較結果をオペランド「Tag_Output」に割り当て。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値		
Tag_Input_1	10 515	10 515	5 232
Tag_Input_2	5 232	10 515	10 515
Tag_Output	1	0	0

四則演算

この章には下記に関する情報が記載されています：

- 整数 (S7-1500)
- 浮動小数点数 (S7-1500)
- 詳細浮動小数点数 (S7-1500)

整数

この章には下記に関する情報が記載されています：

+I: 整数加算(16 ビット)

説明

「整数加算(16 ビット)」命令を使用して、アキュムレータ 1 および 2 の右ワード内の値を加算します。この命令は、この値を 16 ビット整数として解釈します。

この命令は、アキュムレータ 1 の右ワードに和を保存します。アキュムレータ 1 の左ワードは、未変更のままです。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、和が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「整数加算(16 ビット)」命令に使用されます。

```
+I
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 加算の初期値のロード。
L "Tag_Value_2"	// 加算の 2 番目の値のロード。
+I	// 値の加算
T "Tag_Result"	// 和をオペランド「Tag_Result」に転送。

-I: 整数減算(16 ビット)

説明

「整数減算(16 ビット)」命令を使用して、アキュムレータ 1 の右ワードの値をアキュムレータ 2 の右ワードの値から減算します。この命令は、アキュムレータ 1 および 2 の値を 16 ビット整数として解釈します。

この命令は、アキュムレータ 1 の右ワードに差を保存します。アキュムレータ 1 の左ワードは、未変更のままです。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、差分が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「整数減算(16 ビット)」命令に使用されます。

```
-I
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 減算の最初の値をアキュムレータ 1 にロード。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // 加算の 2 番目の値をアキュムレータ 1 にロード。
-I	// アキュムレータ 1 の値をアキュムレータ 2 の値から減算。
T "Tag_Result"	// 差をオペランド「Tag_Result」に転送。

*I: 整数乗算(16 ビット)

説明

「整数乗算(16 ビット)」を使用して、アキュムレータ 1 および 2 の右ワード内の値を乗算します。この命令は、この値を 16 ビット整数として解釈します。

この命令は、アキュムレータ 1 に積を 32 ビット整数(DINT)として保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、積が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「整数乗算(16 ビット)」命令に使用されます。

```
*I
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 乗算の最初の値のロード。
L "Tag_Value_2"	// 乗算の 2 番目の値のロード。
*I	// 値を乗算
T "Tag_Result"	// 積をオペランド「Tag_Result」に転送。

/I: 整数除算(16 ビット)

説明

「整数除算(16 ビット)」命令を使用して、アキュムレータ 2 の右ワードの値をアキュムレータ 1 の右ワードの値で除算します。この命令は、アキュムレータの値を 16 ビット整数として解釈します。

この命令では、次の 2 つの結果が提供されます: 商と除算の余り。 商は、除算の整数結果です。 この命令は、整数をアキュムレータ 1 の右ワードに保存します。また、除算の余りをアキュムレータ 1 の左ワードに保存します。被除数が負の場合、除算の余りも負となります。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、商が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

0 による除算の場合、命令は値を商として報告し、どの場合でも値 0 を余りとして報告します。 この場合、ステータスビット CC 0、CC 1、OV、および OS がシグナル状態「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「整数除算(16 ビット)」命令に使用されます。

```
/I
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 被除数をアキュムレータ 1 にロード。
L "Tag_Value_2"	// 被除数をアキュムレータ 2 に移動。 // 除数をアキュムレータ 1 にロード。
/I	// アキュムレータ 2 の値をアキュムレータ 1 の値で除算。
T "Tag_Result"	// 結果をオペランド「Tag_Result」に転送。

処理前

次の表に、「整数除算(16 ビット)」命令の実行前のアキュムレータ 1 および 2 の右ワードの内容を示します。

アキュムレータ	値
アキュムレータ 2	13
アキュムレータ 1	4

処理後

次の表に、「整数除算(16 ビット)」命令の実行後のアキュムレータ 1 の内容を示します。

アキュムレータ 1	値
ビット 0～15 (右ワード)	3
ビット 16～31 (左ワード)	1

+D: 倍長整数加算(32 ビット)

説明

「倍長整数加算(32 ビット)」命令を使用して、アキュムレータ 1 および 2 の右ワード内の値を加算します。この命令は、この値を 32 ビット整数として解釈します。 この命令は、和をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、和が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「倍長整数加算(32 ビット)」命令に使用されます。

```
+D
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 加算の初期値のロード。
L "Tag_Value_2"	// 加算の 2 番目の値のロード。
+D	// 値の加算
T "Tag_Result"	// 和をオペランド「Tag_Result」に転送。

-D: 倍長整数減算(32 ビット)

説明

「倍長整数減算(32 ビット)」命令を使用して、アキュムレータ 1 の値をアキュムレータ 2 の値から減算します。この命令は、アキュムレータ 1 および 2 の値を 32 ビット整数として解釈します。 この命令は、減算の結果をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、差分が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「倍長整数減算(32 ビット)」命令に使用されます。

```
-D
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 減算の最初の値をアキュムレータ 1 にロード。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // 減算の 2 番目の値をアキュムレータ 1 にロード。
-D	// アキュムレータ 1 の値をアキュムレータ 2 の値から減算。
T "Tag_Result"	// 差をオペランド「Tag_Result」に転送。

*D: 倍長整数乗算(32 ビット)

説明

「倍長整数乗算(32 ビット)」命令を使用して、アキュムレータ 1 および 2 の右ワード内の値を乗算します。この命令は、この値を 32 ビット整数として解釈します。 この命令は、積をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、積が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「倍長整数乗算(32 ビット)」命令に使用されます。

```
*D
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 乗算の最初の値のロード。
L "Tag_Value_2"	// 乗算の 2 番目の値のロード。
*D	// 値を乗算
T "Tag_Result"	// 積をオペランド「Tag_Result」に転送。

/D: 倍長整数除算(32 ビット)

説明

「倍長整数除算(32 ビット)」命令を使用して、アキュムレータ 2 の値をアキュムレータ 1 の値で除算します。この命令は、アキュムレータ 1 および 2 の値を 32 ビット整数として解釈します。 この命令は、除算の結果(商)をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、商が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

0 による除算の場合、命令は値 0 を商として報告します。この場合、ステータスビット CC 0、CC 1、OV、および OS がシグナル状態「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

以下の構文が「倍長整数除算(32 ビット)」命令に使用されます。

```
/D
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 被除数をアキュムレータ 1 にロード。
L "Tag_Value_2"	// 被除数をアキュムレータ 2 に移動。 // 除数をアキュムレータ 1 にロード。
/D	// アキュムレータ 2 の値をアキュムレータ 1 の値で除算。
T "Tag_Result"	// 商をオペランド「Tag_Result」に転送。

処理前

次の表に、「倍長整数除算(32 ビット)」命令の実行処理前のアキュムレータ 1 および 2 の内容を示します。

アキュムレータ	値
アキュムレータ 2	13
アキュムレータ 1	4

処理後

次の表に、「倍長整数除算(32 ビット)」命令の実行後のアキュムレータ 1 の内容を示します。

アキュムレータ	値
アキュムレータ 1	3

+: 定数加算

説明

「定数加算」命令を使用して指定された定数の値をアキュムレータ 1 の内容に加算します。定数値を 16 ビットまたは 32 ビット整数として指定できます。 この命令は結果をアキュムレータ 1 に保存します。16 ビット整数を加算する場合、命令はアキュムレータ 1 の右ワードにのみ影響します。

16 ビット整数の加算を DINT 演算ファンクションとして実行するには、定数の前に「L#」を入力します。 「定数加算」命令を使用すると、負数の使用が許可されます。

この命令はステータスビットには影響しません。

構文

「定数加算」命令には、以下の構文を使用します。

```
+ <constant>
```

パラメータ

次の表に、「定数加算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	INT, DINT	加算される定数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をロード。
+ 25	// アキュムレータ 1 の値を 25 インクリメント。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランド「Tag_Result_1」に転送。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
L "Tag_Value_3"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード
+ -100	// アキュムレータ 1 の値を 100 デクリメント。
<I	// アキュムレータ 2 の値がアキュムレータ 1 の値未満かどうかを比較
JC NEXT	// 照会結果が「1」の場合、ジャンプラベル NEXT にジャンプ。 // 問い合わせの結果が「0」の場合、次の命令を実行。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランド「Tag_Result_2」に転送。

```
L "Tag_Value_4"          // オペランド「Tag_Value_4」の値をロード。
                         // アキュムレータ 1 の内容を 20 インクリメント。
+ L#20
                         // 加算は DINT 計算として実行される
T "Tag_Result_3"         // アキュムレータ 1 の内容をオペランド「Tag_Result_3」に転送。
                         // ジャンプラベル「NEXT」
NEXT: L "Tag_Value_5"
                         // オペランド「Tag_Value_5」の値をロード。
+D                       // アキュムレータ 1 および 2 の値を加算。
T "Tag_Result_4"         // 和をオペランド「Tag_Result_4」に転送。
```

INC: インクリメント

説明

「インクリメント」命令を使用して、アキュムレータ 1 の値を命令のパラメータとして指定された値によってインクリメントします。 指定された値は値の範囲 0～255 になります。この値はアキュムレータ 1 の右バイトのみでインクリメントされます。左のバイトへの転送は発生しません。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータ 2 の内容は変わりません。

パラメータ値が値の許容範囲外にある場合は、エラーメッセージを生成することなく誤った結果となります。

構文

「インクリメント」命令には、以下の構文を使用します。

```
INC <constant>
```

パラメータ

次の表に、「インクリメント」命令のパラメータを示します。

パラメータ	書式	説明
<constant>	8 ビット整数	値がインクリメントされる値

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// オペランドの値をアキュムレータ 1 にロード。
INC 5	// アキュムレータ 1 の値を 5 インクリメント。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランドに転送。

DEC: デクリメント

説明

「デクリメント」命令を使用して、アキュムレータ 1 の値を命令のパラメータとして指定された値によってデクリメントします。 指定された値は値の範囲 0～255 になります。この値はアキュムレータ 1 の右バイトのみでデクリメントされます。左のバイトへの転送は発生しません。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータ 2 の内容は変わりません。

パラメータ値が値の許容範囲外にある場合は、エラーメッセージを生成することなく誤った結果となります。

構文

「デクリメント」命令には、以下の構文を使用します。

```
DEC <constant>
```

パラメータ

次の表に、「デクリメント」命令のパラメータを示します。

パラメータ	書式	説明
<constant>	8 ビット整数	値がデクリメントされる値

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// オペランドの値をアキュムレータ 1 にロード。
DEC 5	// アキュムレータ 1 の値を 5 デクリメント。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランドに転送。

MOD: 除算の剰余を返す

説明

「除算の剰余を返す」命令を使用して、アキュムレータ 2 の値をアキュムレータ 1 の値で除算し、除算の余りを保存します。

この命令は、アキュムレータ 1 および 2 の内容を 32 ビット整数として解釈します。 この命令は、除算の余りをアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、除算の剰余が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

0 による除算の場合、命令は値 0 を結果として返します。 この場合、ステータスビット CC 0、CC 1、OV、および OS がシグナル状態「1」にセットされます。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

「除算の剰余を返す」命令には、以下の構文を使用します。

```
MOD
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 被除数をアキュムレータ 1 にロード。
L "Tag_Value_2"	// 被除数をアキュムレータ 2 に移動。 // 除数をアキュムレータ 1 にロード。
MOD	// アキュムレータ 2 の値をアキュムレータ 1 の値で除算。
T "Tag_Result"	// 除算の余りをオペランド「Tag_Result」に転送。

処理前

次の表に、「除算の剰余を返す」命令の実行前のアキュムレータ 1 および 2 の内容を示します。

アキュムレータ	値
アキュムレータ 2	13
アキュムレータ 1	4

処理後

次の表に、「除算の剰余を返す」命令の実行後のアキュムレータ 1 の内容を示します。

アキュムレータ	値
アキュムレータ 1	1

浮動小数点数

この章には下記に関する情報が記載されています：

- +R: 浮動小数点数加算 (S7-1500)
- -R: 浮動小数点数減算 (S7-1500)
- *R: 浮動小数点数乗算 (S7-1500)
- /R: 浮動小数点数除算 (S7-1500)
- ABS: 絶対値の形成 (S7-1500)

+R: 浮動小数点数加算

説明

「浮動小数点数加算」命令を使用して、アキュムレータ 1 および 2 に含まれている値を加算します。この命令は、この値を浮動小数点数として解釈します。 この命令は、和をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、和が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

無効な浮動小数点数または無限記号(+/- ∞)が入力された場合、この命令はアキュムレータ 1 に無効な値を書き込み、スタータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

「浮動小数点数加算」命令には、以下の構文を使用します。

```
+R
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 加算の初期値のロード。
L "Tag_Value_2"	// 加算の 2 番目の値のロード。
+R	// 値の加算
T "Tag_Result"	// 和をオペランド「Tag_Result」に転送。

-R: 浮動小数点数減算

説明

「浮動小数点数減算」命令を使用して、アキュムレータ 1 の値をアキュムレータ 2 の値から減算します。この命令は、アキュムレータ 1 および 2 の値を浮動小数点数として解釈します。 この命令は、差をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、差分が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

無効な浮動小数点数または無限記号(+/- ∞)が入力された場合、この命令はアキュムレータ 1 に無効な値を書き込み、スタータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

「浮動小数点数減算」命令には、以下の構文を使用します。

```
-R
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 減算の最初の値をアキュムレータ 1 にロード。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // 減算の 2 番目の値をアキュムレータ 1 にロード。
-R	// アキュムレータ 1 の値をアキュムレータ 2 の値から減算。
T "Tag_Result"	// 差をオペランド「Tag_Result」に転送。

*R: 浮動小数点数乗算

説明

「浮動小数点数乗算」命令を使用して、アキュムレータ 1 および 2 に含まれている値を乗算します。この命令は、この値を浮動小数点数として解釈します。 この命令は、積をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、積が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

無効な浮動小数点数または無限記号(+/- ∞)が入力された場合、この命令はアキュムレータ 1 に無効な値を書き込み、スタータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

「浮動小数点数乗算」命令には、以下の構文を使用します。

```
*R
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 乗算の最初の値のロード。
L "Tag_Value_2"	// 乗算の 2 番目の値のロード。
*R	// 値を乗算
T "Tag_Result"	// 積をオペランド「Tag_Result」に転送。

/R: 浮動小数点数除算

説明

「浮動小数点数除算」命令を使用して、アキュムレータ 2 の値をアキュムレータ 1 の値で除算します。この命令は、アキュムレータ 1 および 2 の値を浮動小数点数として解釈します。 この命令は、除算の結果(商)をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、商が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

0 による除算の場合、命令は値 0 を商として報告します。 この場合、ステータスビット CC 0、CC 1、OV、および OS がシグナル状態「1」にセットされます。

無効な浮動小数点数または無限記号(+/- ∞)が入力された場合、この命令はアキュムレータ 1 に無効な値を書き込み、スタータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令実行後、アキュムレータ 2 の内容は変更されません。

構文

「浮動小数点数除算」命令には、以下の構文を使用します。

```
/R
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 被除数をアキュムレータ 1 にロード。
L "Tag_Value_2"	// 被除数をアキュムレータ 2 に移動。 // 除数をアキュムレータ 1 にロード。
/R	// アキュムレータ 2 の値をアキュムレータ 1 の値で除算。
T "Tag_Result"	// 商をオペランド「Tag_Result」に転送。

ABS: 絶対値の形成

説明

「絶対値の形成」命令を使用して、アキュムレータ 1 の値の絶対値を形成します。この命令は、アキュムレータ 1 の内容を浮動小数点数として解釈し、仮数の符号を「0」にセットします。 この結果は、アキュムレータ 1 に保存されます。

「絶対値の形成」命令はステータスビットに影響しません。

構文

「絶対値の形成」命令には、以下の構文を使用します。

```
ABS
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// 浮動小数点数をアキュムレータ 1 にロード
ABS	// 絶対値の形成
T "Tag_Result"	// 結果をオペランド「Tag_Result」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	-1.5E+02
Tag_Result	1.5E+02

詳細浮動小数点数

この章には下記に関する情報が記載されています：

SQR: 2 乗値の形成

説明

「2 乗値の形成」命令を使用して、アキュムレータ 1 に入っている浮動小数点数の 2 乗値を計算します。結果はアキュムレータ 1 に保存されます。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

浮動小数点数が無効の場合、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「2 乗値の形成」命令には、以下の構文を使用します。

```
SQR
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード

SQR	// 2 乗値の形成
	// 結果をアキュムレータ 1 に保存。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。

L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。 // 積をアキュムレータ 1 に保存。

SQRT: 平方根の形成

説明

「平方根の形成」命令を使用して、アキュムレータ 1 に入っている浮動小数点数の平方根を形成します。結果はアキュムレータ 1 に保存されます。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

アキュムレータ 1 の値が 0 未満または無効な浮動小数点数である場合、命令はアキュムレータ 1 に無効な値を書き込み、ステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

アキュムレータ 1 の値が「0」の場合、命令は値「0」を返します。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「平方根の形成」命令には、以下の構文を使用します。

```
SQRT
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
+R	// アキュムレータ 1 および 2 の値を加算。 // 和をアキュムレータ 1 に保存。
SQRT	// 平方根を取得
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

EXP: 指数値の形成

説明

「指数値の形成」を使用して、基数 e (e = 2.718282)から指数および浮動小数点数をアキュムレータ 1 に形成します。この命令は結果をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

浮動小数点数が無効の場合、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「指数値の形成」命令には、以下の構文を使用します。

```
EXP
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
EXP	// 指数値の形成 // 結果をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

LN: 自然対数の形成

説明

「自然対数の形成」命令を使用して、アキュムレータ 1 内の e (e = 2.718282)を底とする自然対数の値を計算します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

アキュムレータ 1 の値が 0 以下の場合、命令はアキュムレータ 1 に無効な値を書き込み、ステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。 浮動小数点数が無効の場合も、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「自然対数の形成」命令には、以下の構文を使用します。

```
LN
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
LN	// 自然対数の形成。 // 結果をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

SIN: 正弦値の形成

説明

「正弦値の形成」命令を使用して、アキュムレータ 1 内の角度の正弦値を計算します。角度は弧度法で指定され、アキュムレータ 1 に浮動小数点数として入っている必要があります。命令は、結果をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。 浮動小数点数が無効の場合、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「正弦値の形成」命令には、以下の構文を使用します。

```
SIN
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
SIN	// 正弦値の形成 // 結果をアキュムレータ 1 に保存。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。 // 積をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

ASIN: 逆正弦値の形成

説明

「逆正弦値の形成」命令を使用し、この値に対応するアキュムレータ 1 内の値から角度のサイズを計算します。 アキュムレータ 1 の値は、「-1」から「1」までの値の範囲の浮動小数点数として存在している必要があります。

命令の結果は弧度法で返され、アキュムレータ 1 に保存されます。アキュムレータ 1 の値に応じて、結果の値の範囲は-π/2 ~ +π/2 (π= 3.14159)となります。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

アキュムレータ 1 の値が許容値の範囲を超える場合、命令はアキュムレータ 1 に無効な値を書き込み、ステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。 浮動小数点数が無効の場合も、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「逆正弦値の形成」命令には、以下の構文を使用します。

```
ASIN
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
ASIN	// 逆正弦値の形成 // 結果をアキュムレータ 1 に保存。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。 // 積をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

COS: 余弦値の形成

説明

「余弦値の形成」命令を使用して、アキュムレータ 1 内の角度の余弦値を計算します。角度は弧度法で指定され、アキュムレータ 1 に浮動小数点数として入っている必要があります。命令は、結果をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。 浮動小数点数が無効の場合、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「余弦値の形成」命令には、以下の構文を使用します。

```
COS
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
COS	// 余弦値の形成
	// 結果をアキュムレータ 1 に保存。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
	// オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。
	// 積をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

ACOS: 逆余弦値の形成

説明

「逆余弦値の形成」命令を使用し、この値に対応するアキュムレータ 1 内の値から角度のサイズを計算します。 アキュムレータ 1 の値は、「-1」から「1」までの値の範囲の浮動小数点数として存在している必要があります。

命令の結果は弧度法で返され、アキュムレータ 1 に保存されます。アキュムレータ 1 の値に応じて、結果の値の範囲は「0」～「π」(π= 3.14159)となります。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

アキュムレータ 1 の値が許容値の範囲を超える場合、命令はアキュムレータ 1 に無効な値を書き込み、ステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。 浮動小数点数が無効の場合も、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「逆余弦値の形成」命令には、以下の構文を使用します。

```
ACOS
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
ACOS	// 逆余弦値の形成 // 結果をアキュムレータ 1 に保存。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。 // 積をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

TAN: 正接値の形成

説明

「正接値の形成」命令を使用して、アキュムレータ 1 内の角度の正接値を計算します。角度は弧度法で指定され、アキュムレータ 1 に浮動小数点数として入っている必要があります。命令は、結果をアキュムレータ 1 に保存します。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。 浮動小数点数が無効の場合、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「正接値の形成」命令には、以下の構文を使用します。

```
TAN
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
TAN	// 正接値の形成 // 結果をアキュムレータ 1 に保存。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。 // 積をアキュムレータ 1 に保存。
T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

ATAN: 逆正接値の形成

説明

「逆正接値の形成」命令を使用し、この値に対応するアキュムレータ 1 内の値から角度のサイズを計算します。 アキュムレータ 1 の値は、浮動小数点数として存在している必要があります。

命令の結果は弧度法で返され、アキュムレータ 1 に保存されます。アキュムレータ 1 の値に応じて、結果の値の範囲は-π/2 ~ +π/2 (π= 3.14159)となります。

命令が実行されると、ステータスビット CC 0 および CC 1 によって、結果が負であるか、0 であるか、正であるかが示されます。 数字が許可された数字の範囲内にない場合、ステータスビット OV および OS が「1」にセットされます。

アキュムレータ 1 の値が許容値の範囲を超える場合､命令はアキュムレータ 1 に無効な値を書き込み、ステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。 浮動小数点数が無効の場合も、命令はステータスビット CC 0、CC 1、OV、および OS を「1」にセットします。

命令は、アキュムレータ 1 の内容のみに影響します。アキュムレータ 2 の内容は、変わりません。

構文

「逆正接値の形成」命令には、以下の構文を使用します。

```
ATAN
```

例

次の例で、命令がどのように動作するかを示します。

```
STL                         説明
L "Tag_Value_1"             // オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
                            // 逆正接値の形成
ATAN
                            // 結果をアキュムレータ 1 に保存。
                            // アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_2"
                            // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
                            // アキュムレータ 1 および 2 の値を乗算。
*R
                            // 積をアキュムレータ 1 に保存。
T "Tag_Result"              // 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。
```

ロードと転送

この章には下記に関する情報が記載されています：

- ロード (S7-1500)
- 転送 (S7-1500)

ロード

この章には下記に関する情報が記載されています：

L: ロード

説明

「ロード」命令は指定されたオペランドの内容をアキュムレータ 1 にロードします。ロード対象のオペランドは、以下のメモリ領域のいずれかでバイト、ワードまたはダブルワードとしてアドレス指定できます。

- 入力および出力(I、Q)のプロセスイメージ
- ビットメモリ(M)
- 一時ローカルデータ(L)
- データブロック(DB、DI)
- ポインタ
- I/O (PI)
- タイマ(T)
- カウンタ(C)

アキュムレータ 1 のメモリ領域はバイト単位で整理され、32 ビット幅です。

命令はロード対象のオペランドの内容をバイトフォーマットによって右揃えでアキュムレータ 1 に書き込みます。アキュムレータ 1 の残りのバイトは「0」で埋められます。

命令はロード対象のオペランドの内容をワードフォーマットによってアキュムレータ 1 の右ワードに書き込みます。したがって上位アドレス指定されたバイトがアキュムレータ 1 の右バイト(ビット 0 ~ 7)に転送されます。下位アドレス指定されたバイトはその左に書き込まれます。 アキュムレータ 1 の左ワードの残りのバイトは「0」で埋められます。

命令はロード対象のオペランドの内容をダブルワードフォーマットによってアキュムレータ 1 の 32 ビットに書き込みます。したがって上位アドレス指定されたバイトがアキュムレータ 1 の右バイト(ビット 0 ~ 7)に転送されます。下位アドレス指定されたバイトはビット 24 ~ 31 に書き込まれます。

次の表に、たとえばアキュムレータ 1 の内容がバイト、ワードおよびダブルワードフォーマットによるオペランドのロードによって変更されるようすを示します。

<table>
<tr><th>命令</th><th colspan="8">アキュムレータ 1</th></tr>
<tr><td></td><td>31 . . .</td><td>. . . 24</td><td>23 . . .</td><td>. . . 16</td><td>15 . . .</td><td>. . . 8</td><td>7 . . .</td><td>. . . 0</td></tr>
<tr><td>バイトのロード:
L MB10</td><td>0000</td><td>0000</td><td>0000</td><td>0000</td><td>0000</td><td>0000</td><td colspan="2"><MB10></td></tr>
<tr><td>ワードのロード:
L MW10</td><td>0000</td><td>0000</td><td>0000</td><td>0000</td><td colspan="2"><MB10></td><td colspan="2"><MB11></td></tr>
<tr><td>ダブルワードのロード:
L MD10</td><td colspan="2"><MB10></td><td colspan="2"><MB11></td><td colspan="2"><MB12></td><td colspan="2"><MB13></td></tr>
</table>

指定されたオペランドのロード中、アキュムレータ 1 の前の内容はアキュムレータ 2 に移動されます。したがって「ロード」命令はアキュムレータ 1 の完全な内容をアキュムレータ 2 に転送します。アキュムレータ 2 の前の内容はこのプロセスで失われます。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「ロード」命令には、以下の構文を使用します。

```
L <operand>
```

パラメータ

次の表に、「ロード」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、タイマ、REAL、DATE、TOD、CHAR	I、Q、PI、M、L、DB、DI、T、C、ポインタ、パラメータ	内容がロードされるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。

*R	// オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
	// アキュムレータ 1 および 2 の値を乗算。
	// 積をアキュムレータ 1 に保存。

L "Tag_Value_3"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。

+R	// オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード
	// アキュムレータ 1 および 2 の値を加算。
	// 和をアキュムレータ 1 に保存。

T "Tag_Result"	// 結果(アキュムレータ 1 の内容)をオペランド「Tag_Result」に転送。

L STW: アキュムレータ 1 にステータスワードをロード

説明

「アキュムレータ 1 にステータスワードをロード」命令を使用してステータスワードをアキュムレータ 1 にロードします。実行されると、命令は個々のステータスビットをアキュムレータ 1 の右ワードに書き込みます。アキュムレータ 1 の残りのビットは「0」で書き込まれます。

次の表に、「アキュムレータ 1 にステータスワードをロード」命令の実行後のアキュムレータの内容を示します。

	アキュムレータ 1									
ビット番号	31-9	8	7	6	5	4	3	2	1	0
内容	0	BR	CC 1	CC 0	OV	OS	0	0	[RLO]	0

命令は、ステータスビットのシグナル状態に関係なく常に実行されます。

構文

「アキュムレータ 1 にステータスワードをロード」命令には、以下の構文を使用します。

```
L STW
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L STW	// アキュムレータ 1 にステータスワードをロード。
T "Tag_STW"	// アキュムレータ 1 の内容(ステータスビット)をオペランド「Tag_STW」に転送。

LAR1: AR1 にアキュムレータ 1 の内容をロード

説明

「AR1 にアキュムレータ 1 の内容をロード」命令を使用して、アキュムレータ 1 の内容をアドレスレジスタ 1 (AR1)にロードします。アキュムレータ 1 の内容は、エリアポインタ (POINTER)のフォーマットと一致している必要があります。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR1 にアキュムレータ 1 の内容をロード」命令には、以下の構文を使用します。

```
LAR1
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L P#10.0	//ポインタ(P#10.0)をアキュムレータ 1 にロード。
LAR1	// アドレスレジスタ 1 にアキュムレータ 1 の内容をロード。
L MW [AR1,P#4.0]	// MW14 をアキュムレータ 1 にロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value"	// オペランド「Tag_Value」の内容をアキュムレータ 1 にロード。
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。
A I [AR1,P#2.1]	// ビット I12.1 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
= "Tag_Output"	// 条件が満たされている場合(RLO = 「1」)、オペランド「Tag_Output」を「1」にセット。

LAR1 <D>: AR1 にダブルワードまたはエリアポインタをロード

説明

「AR1 にダブルワードまたはエリアポインタをロード」命令を使用して、アドレスレジスタ 1 (AR1)にダブルワードまたは領域内部またはポインタに交差する領域の内容をロードします。 ダブルワードの内容は、エリアポインタのフォーマットに一致している必要があります。

ポインタおよびダブルワードは、以下のいずれか 1 つのメモリ領域をアドレス指定できます。

- ビットメモリ(M)
- 一時ローカルデータ(L)
- データブロック(DB、DI)

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR1 にダブルワードまたはエリアポインタをロード」命令には、以下の構文を使用します。

```
LAR1 <D>
```

パラメータ

「AR1 にダブルワードまたはエリアポインタをロード」命令には、以下の構文を使用します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<D>	Input	DWORD, POINTER	D、M、L	内容がロードされるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR1 P#10.0	// アドレスレジスタ 1 にエリア内部ポインタ P#10.0 の内容をロード。
LMW [AR1,P#4.0]	// MW14 をアキュムレータ 1 にロード。 // アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value"	// オペランド「Tag_Value」の内容をアキュムレータ 1 にロード。
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。

```
A I [AR1,P#2.1]          // ビット I12.1 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
= "Tag_Output"           // 条件が満たされている場合(RLO = 「1」)、オペランド「Tag_Output」を「1」にセット。
LAR1 MD20                // アドレスレジスタ 1 に MD20 (MD20 = P#30.0)の内容をロード
A I [AR1,P#2.1]          // ビット I32.1 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
```

LAR1 AR2: AR1 に AR2 の内容をロード

説明

「AR1 に AR2 の内容をロード」命令を使用して、アドレスレジスタ 2 (AR2)の内容をアドレスレジスタ 1 (AR1)にロードします。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR1 に AR2 の内容をロード」命令には、以下の構文を使用します。

```
LAR1 AR2
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR2 P#20.0	// アドレスレジスタ 2 にエリア内部ポインタ P#20.0 の内容をロード。
LAR1 AR2	// アドレスレジスタ 2 の内容をアドレスレジスタ 1 にコピー。
L MW [AR1,P#4.0]	// MW24 をアキュムレータ 1 にロード。
T "Tag_2"	// アキュムレータ 1 を「Tag_2」に転送。

LAR2: AR2 にアキュムレータ 1 の内容をロード

説明

「AR2 にアキュムレータ 1 の内容をロード」命令を使用して、アキュムレータ 1 の内容をアドレスレジスタ 2 (AR2)にロードします。アキュムレータ 1 の内容は、エリアポインタ (POINTER)のフォーマットと一致している必要があります。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR2 にアキュムレータ 1 の内容をロード」命令には、以下の構文を使用します。

```
LAR2
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L P#10.0	//ポインタ(P#10.0)をアキュムレータ 1 にロード。
LAR2	// アドレスレジスタ 2 にアキュムレータ 1 の内容をロード。
L MW [AR2,P#4.0]	// MW14 をアキュムレータ 1 にロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value"	// オペランド「Tag_Value」の内容をアキュムレータ 1 にロード。
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。
A I [AR2,P#2.1]	// ビット I12.1 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
= "Tag_Output"	// 条件が満たされている場合(RLO = 「1」)、オペランド「Tag_Output」を「1」にセット。

LAR2 <D>: AR2 にダブルワードまたはエリアポインタをロード

説明

「AR2 にダブルワードまたはエリアポインタをロード」命令を使用して、アドレスレジスタ 2 (AR2)にダブルワードまたは領域内部またはクロスエリアのポインタの内容をロードします。 ダブルワードの内容は、エリアポインタ(POINTER)のフォーマットに一致している必要があります。

ポインタおよびダブルワードは、以下のいずれか 1 つのメモリ領域をアドレス指定できます。

- ビットメモリ(M)
- 一時ローカルデータ(L)
- データブロック(DB、DI)

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR2 にダブルワードまたはエリアポインタをロード」命令には、以下の構文を使用します。

```
LAR2 <D>
```

パラメータ

「AR2 にダブルワードまたはエリアポインタをロード」命令には、以下の構文を使用します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<D>	Input	DWORD, POINTER	D、M、L	内容がロードされるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR2 P#10.0	// アドレスレジスタ 2 にエリア内部ポインタ P#10.0 の内容をロード。
L MW [AR2,P#4.0]	// MW14 をアキュムレータ 1 にロード。 // アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value"	// オペランド「Tag_Value」の内容をアキュムレータ 1 にロード。
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。

```
A I [AR2,P#2.1]          // ビット I12.1 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
= "Tag_Output"           // 条件が満たされている場合(RLO = 「1」)、オペランド「Tag_Output」を「1」にセット。
LAR2 MD20                // アドレスレジスタ 2 に MD20 (MD20 = P#30.0)の内容をロード
A I [AR2,P#2.1]          // ビット I32.1 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
```

転送

この章には下記に関する情報が記載されています：

T: 転送

説明

「転送」命令を使用して、アキュムレータ 1 の内容をバイト、ワードまたはダブルワードフォーマットで指定されたオペランドに転送します。 アキュムレータ 1 の内容はこの命令によって変更されません。 指定されたオペランドは、以下のメモリ領域の 1 つでアドレス指定する必要があります。

- 入力および出力(I、Q)のプロセスイメージ
- ビットメモリ(M)
- 一時ローカルデータ(L)
- データブロック(DB、DI)
- I/O (PQ)

転送対象のバイト数は、指定されたオペランドのフォーマットによって異なります。 オペランドがバイトフォーマットで存在する場合、命令はアキュムレータ 1 のビット 0 ~ 7 の内容を転送します。

オペランドがワードフォーマットで存在する場合、命令はアキュムレータ 1 で右揃えされるワードの内容を転送します。アキュムレータ 1 の右バイトはオペランドの上位アドレス指定されたバイトに転送されます。 アキュムレータ 1 のビット 8 ~ 15 は、オペランドの下位バイトに転送されます。

指定したオペランドがダブルワードフォーマットで使用可能な場合、命令はアキュムレータ 1 の内容全体を転送します。アキュムレータ 1 の右バイトは、オペランドの最上位アドレス指定したバイトに転送されます。 アキュムレータ 1 のビット 24 ~ 31 は、オペランドの最下位バイトに転送されます。

次の表に、アキュムレータ 1 の内容がバイト、ワードおよびダブルワードフォーマットで転送される例を示します。

命令	アキュムレータ 1			
ビット	31 24	23 16	15 8	7 0
アキュムレータ 1 の内容	バイト(n)	バイト(n+1)	バイト(n+2)	バイト(n+3)
バイトの転送				バイト(n)
ワードの転送			バイト(n)	バイト(n+1)
ダブルワードの転送	バイト(n)	バイト(n+1)	バイト(n+2)	バイト(n+3)

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「転送」命令には、以下の構文を使用します。

```
T <operand>
```

パラメータ

次の表に、「転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Output	BYTE、WORD、DWORD、SINT、INT、DINT、USINT、UINT、UDINT、タイマ、REAL、DATE、TOD、CHAR	I、Q、PQ、M、L、DB、DI	アキュムレータ 1 の内容が転送されるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
	// アキュムレータ 1 および 2 の値を乗算。
*R	// 積をアキュムレータ 1 に保存。
T "Tag_Result_1"	// 積(アキュムレータ 1 の内容)をオペランド「Tag_Result_1」に転送。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_3"	// オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード
	// アキュムレータ 1 および 2 の値を加算。
+R	// 和をアキュムレータ 1 に保存。
T "Tag_Result_2"	// 合計(アキュムレータ 1 の内容)をオペランド「Tag_Result_2」に転送

T STW: アキュムレータ 1 をステータスワードに転送

説明

「アキュムレータ 1 をステータスワードに転送」命令を使用して、アキュムレータ 1 のビット 0～8 をステータスワードに転送します。

次の表に、命令の処理によってステータスビットを上書きするアキュムレータ 1 のビットを示します。

	アキュムレータ 1								
ビット番号	8	7	6	5	4	3	2	1	0
ステータスビット	BR	CC 1	CC 0	OV	OS	0	0	[RLO]	0

命令は、ステータスビットのシグナル状態に関係なく常に実行されます。

構文

「アキュムレータ 1 をステータスワードに転送」命令には、以下の構文を使用します。

```
T  STW
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_STW"	// オペランド「Tag_STW」の値をアキュムレータ 1 にロード。
T STW	// アキュムレータ 1 のビット 0～8 をステータスワードに転送

CAR: AR1 と AR2 を切り替える

説明

「AR1 と AR2 を切り替える」命令を使用して、アドレスレジスタ 1 (AR1)と 2 (AR2)の内容を切り替えます。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「"AR1 と AR2 を切り替える」命令には、以下の構文を使用します。

```
CAR
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR1 P#10.0	// アドレスレジスタ 1 にエリア内部ポインタ P#10.0 の内容をロード。
LAR2 P#20.0	// アドレスレジスタ 2 にエリア内部ポインタ P#20.0 の内容をロード。
CAR	// アドレスレジスタの内容を切り替え。
L MD [AR1,P#2.0]	// MD22 をアキュムレータ 1 にロード。
T MD [AR2,P#2.0]	// MD12 をアキュムレータ 2 に転送。

TAR1: AR1 をアキュムレータ 1 に転送

説明

「AR1 をアキュムレータ 1 に転送」命令を使用して、アドレスレジスタ 1 (AR1)の内容をアキュムレータ 1 に転送します。アキュムレータ 1 の内容はアキュムレータ 2 に移動します。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「AR1 をアキュムレータ 1 に転送」命令には、以下の構文を使用します。

```
TAR1
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR1 P#15.0	// アドレスレジスタ 1 にエリア内部ポインタ P#15.0 の内容をロード。
TAR1	// アドレスレジスタ 1 (P#15.0)の内容をアキュムレータ 1 に転送。
T "Tag_Pointer"	// アキュムレータ 1 の内容をオペランド「Tag_Pointer」に転送。

TAR1 <D>: AR1 をダブルワードに転送

説明

「AR1 をダブルワードに転送」命令を使用して、アドレスレジスタ 1 (AR1)の内容をダブルワードに転送します。 ダブルワードは、以下のメモリ領域の 1 つでアドレス指定する必要があります。

- ビットメモリ(M)
- 一時ローカルデータ(L)
- データブロック(DB、DI)

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR1 をダブルワードに転送」命令には、以下の構文を使用します。

```
TAR1 <D>
```

パラメータ

次の表に、「AR1 をダブルワードに転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<D>	Output	DWORD	D、M、L	アキュムレータ 1 の内容が転送されるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
TAR1 %DBD20	// アドレスレジスタ 1 の内容をダブルワード DBD20 に転送。
TAR1 %DID30	// アドレスレジスタ 1 の内容をインスタンスダブルワード DID30 に転送。
TAR1 %LD18	// アドレスレジスタ 1 の内容をローカルデータダブルワード LD18 に転送。
TAR1 %MD24	// アドレスレジスタ 1 の内容をメモリダブルワード MD24 に転送。

TAR1 AR2: AR1 を AR2 に転送

説明

「AR1 を AR2 に転送」命令を使用して、アドレスレジスタ 1 (AR1)の内容をアドレスレジスタ 2 (AR2)にコピーします。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「"AR1 を AR2 に転送」命令には、以下の構文を使用します。

```
TAR1 AR2
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR1 P#10.4	// アドレスレジスタ 1 にエリア内部ポインタ P#10.4 の内容をロード。
TAR1 AR2	// アドレスレジスタ 1 の内容をアドレスレジスタ 2 に転送。
A I [AR2,P#3.2]	// ビット I13.6 のシグナル状態が「1」かどうかをチェックし、現在の RLO で論理積演算。
= "Tag1"	// オペランド「Tag_1」を「1」にセット

TAR2: AR2 をアキュムレータ 1 に転送

説明

「AR2 をアキュムレータ 1 に転送」命令を使用して、アドレスレジスタ 2 (AR2)の内容をアキュムレータ 1 に転送します。アキュムレータ 1 の内容はアキュムレータ 2 に移動します。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「AR2 をアキュムレータ 1 に転送」命令には、以下の構文を使用します。

```
TAR2
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
LAR2 P#20.0	// アドレスレジスタ 2 にエリア内部ポインタ P#20.0 の内容をロード。
TAR2	// アドレスレジスタ 2 の内容(P#20.0)をアキュムレータ 1 に転送。
T "Tag_Pointer"	// アキュムレータ 1 の内容をオペランド「Tag_Pointer」に転送。

TAR2 <D>: AR2 をダブルワードに転送

説明

「AR2 をダブルワードに転送」命令を使用して、アドレスレジスタ 2 (AR2)の内容をダブルワードに転送します。 ダブルワードは、以下のメモリ領域の 1 つでアドレス指定する必要があります。

- ビットメモリ(M)
- 一時ローカルデータ(L)
- データブロック(DB、DI)

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

「AR2 をダブルワードに転送」命令には、以下の構文を使用します。

```
TAR2 <D>
```

パラメータ

次の表に、「AR2 をダブルワードに転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<D>	Output	DWORD	D、M、L	アキュムレータ 2 の内容が転送されるオペランド。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
TAR2 %DBD20	// アドレスレジスタ 2 の内容をダブルワード DBD20 に転送。
TAR2 %DID30	// アドレスレジスタ 2 の内容をインスタンスダブルワード DID30 に転送。
TAR2 %LD18	// アドレスレジスタ 2 の内容をローカルデータダブルワード LD18 に転送。
TAR2 %MD24	// アドレスレジスタ 2 の内容をメモリダブルワード MD24 に転送。

変換操作

この章には下記に関する情報が記載されています：

BTI: BCD を整数(16 ビット)に変換

説明

「BCD を整数(16 ビット)に変換」命令を使用して、アキュムレータ 1 の右ワードの値を 16 ビット整数に変換します。 この命令は、変換するワードを 3 桁の 2 進化 10 進数(BCD)として解釈します。

アキュムレータ 1 のビット 1～11 は、変換する値を示しています。 「-999」～「+999」の範囲の値が有効です。

結果値の符号は、アキュムレータ 1 のビット 15 から読み出されます。ビットのシグナル状態が「0」の場合、符号は正です。 ビットのシグナル状態が「1」の場合、符号は負です。 ビット 12～14 は、変換に使用されません。

BCD コードに BCD エラーが含まれている場合、CPU はステータスワードの OV および OS ビットをセットします。

変換の結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードの内容は、命令に影響されません。

構文

「BCD を整数(16 ビット)に変換」命令には、以下の構文を使用します。

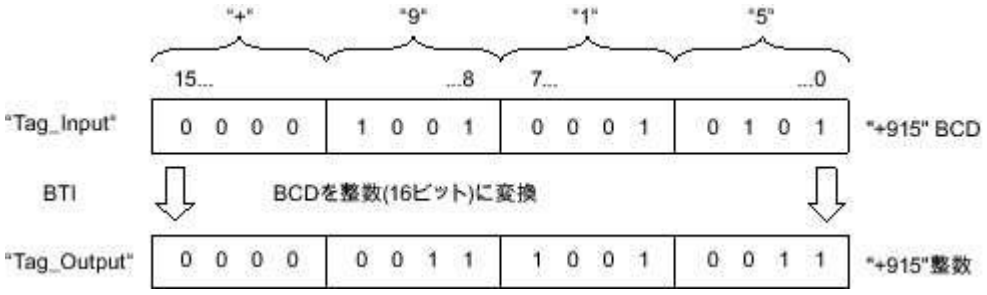
```
BTI
```

例

次の例で、命令がどのように機能するかを示します。

STL	説明
L "Tag_Input"	// BCD をアキュムレータ 1 にロード。
BTI	// BCD を 16 ビット整数に変換。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の図に、命令が特定のオペランド値を使用してどのように機能するかを示します。



ITB: 整数(16 ビット)を BCD に変換

説明

「整数(16 ビット)を BCD に変換」命令を使用して、アキュムレータ 1 の右ワードの値を 3 桁の 2 進化 10 進数(BCD)に変換します。 この命令は、変換する値を 16 ビット整数として解釈します。

アキュムレータ 1 のビット 0～11 は、変換する値を示しています。 「-999」～「+999」の範囲の値が有効です。 変換される数値がこの範囲にない場合、ビット OV および OS がシグナル状態「1」にセットされます。 この場合、変換は実行されません。

結果値の符号は、アキュムレータ 1 のビット 12～15 から読み出されます。ビットのシグナル状態が「0」の場合、符号は正です。 4 つのすべてのビットのシグナル状態が「1」の場合、符号は負です。

変換の結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードの内容は、命令に影響されません。

構文

「整数(16 ビット)を BCD に変換」命令には、以下の構文を使用します。

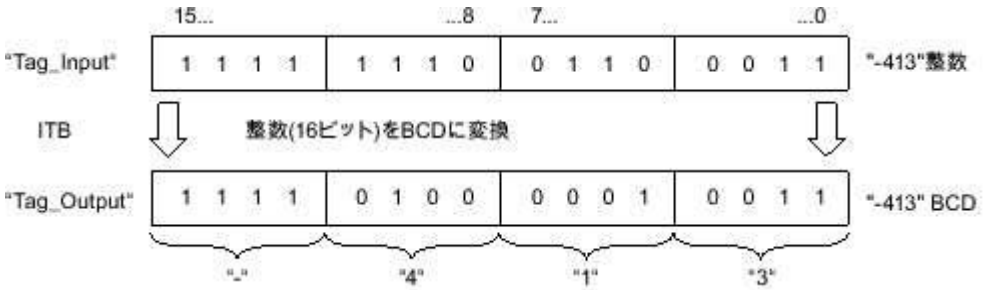
```
ITB
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 整数(16 ビット)をアキュムレータ 1 にロード。
ITB	// 整数を BCD に変換。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



BTD: BCD を整数(32 ビット)に変換

説明

「BCD を整数(32 ビット)に変換」命令を使用して、アキュムレータ 1 の値を 32 ビット整数に変換することができます。 この命令は、変換する値を 7 桁の 2 進化 10 進数(BCD)として解釈します。

アキュムレータ 1 のビット 0～27 は、変換する値を示しています。 「-9999999」～「+9999999」の範囲の値が有効です。

結果値の符号は、アキュムレータ 1 のビット 31 から読み出されます。ビットのシグナル状態が「0」の場合、符号は正です。 ビットのシグナル状態が「1」の場合、符号は負です。 ビット 28～30 は、変換に使用されません。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

BCD コードに BCD エラーが含まれている場合、CPU はステータスワードの OV および OS ビットをセットします。

構文

「BCD を整数(32 ビット)に変換」命令には、以下の構文を使用します。

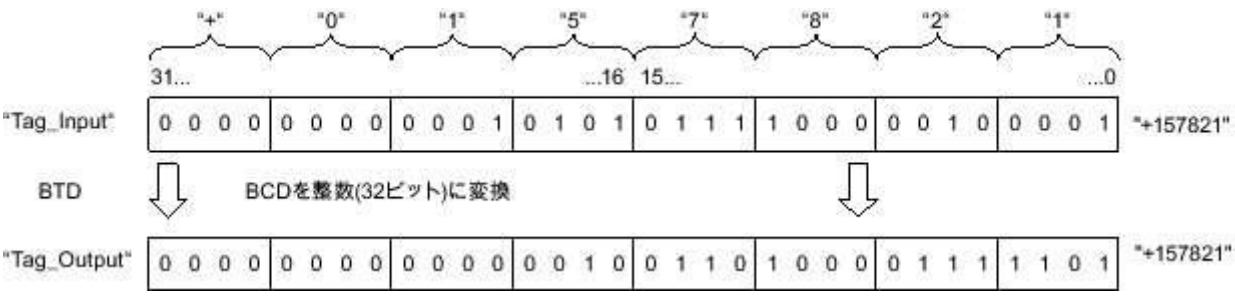
```
BTD
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// BCD をアキュムレータ 1 にロード。
BTD	// BCD が 32 ビット整数に変換される。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



ITD: 整数(16 ビット)を整数(32 ビット)に変換

説明

「整数(16 ビット)を整数(32 ビット)に変換」命令を使用して、アキュムレータ 1 の右ワードの値を 32 ビット整数に変換します。 この命令は、変換する値を 16 ビット整数として解釈します。

ビット 0 ~ 15 は、変換に使用されません。 結果値のビット 16 ~ 31 には、ビット 15 のシグナル状態が入力されます。

結果値は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

構文

「整数(16 ビット)を整数(32 ビット)に変換」命令には、以下の構文を使用します。

```
ITD
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 整数(16 ビット)をアキュムレータ 1 にロード。
ITD	// 整数(16 ビット)を整数(32 ビット)に変換。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令の処理前および処理後のアキュムレータ 1 の内容を示します。

ステータス	アキュムレータ 1							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
処理前	XXXX	XXXX	XXXX	XXXX	1111	1111	1111	0110
処理後	1111	1111	1111	1111	1111	1111	1111	0110
	X: ビットのシグナル状態は、変換とは無関係です。							

DTB: 整数(32 ビット)を BCD に変換

説明

「整数(32 ビット)を BCD に変換」命令を使用して、アキュムレータ 1 の値を 7 桁の 2 進化 10 進数(BCD)に変換します。 この命令は、変換する値を 32 ビット整数として解釈します。

アキュムレータ 1 のビット 0～27 は、変換する値を示しています。 「-9999999」～「+9999999」の範囲の値が有効です。 変換される数値がこの範囲にない場合、ビット OV および OS がシグナル状態「1」にセットされます。 この場合、変換は実行されません。

結果値の符号は、アキュムレータ 1 のビット 28～31 から読み出されます。ビットのシグナル状態が「0」の場合、符号は正です。 4 つのすべてのビットのシグナル状態が「1」の場合、符号は負です。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

構文

「整数(32 ビット)を BCD に変換」命令には、以下の構文を使用します。

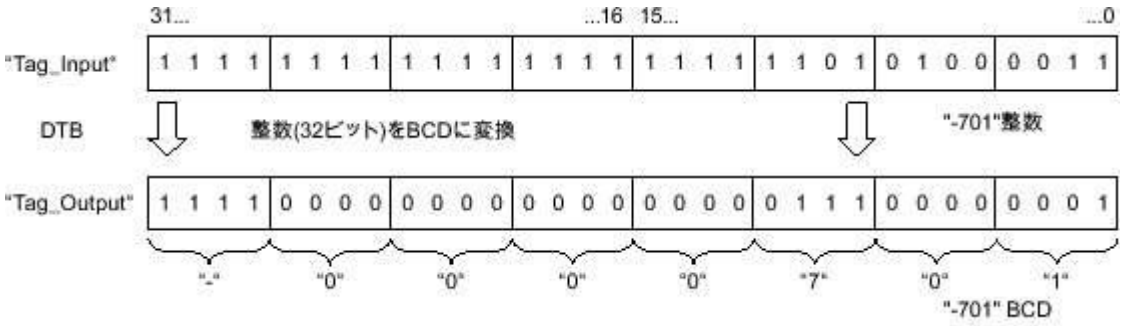
```
DTB
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 整数(32 ビット)をアキュムレータ 1 にロード
DTB	// 整数(32 ビット)を BCD に変換
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



DTR: 整数(32 ビット)を浮動小数点数に変換

説明

「整数(32 ビット)を浮動小数点数に変換」を使用して、アキュムレータ 1 の値を浮動小数点数に変換します。 この命令は、変換する値を 32 ビット整数として解釈します。

32 ビット整数のほうが浮動小数点数よりも確度が高いため、変換の結果は次の表現可能な数値で四捨五入されます。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

構文

「整数(32 ビット)を浮動小数点数に変換」命令には、以下の構文を使用します。

```
DTR
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 整数(32 ビット)をアキュムレータ 1 にロード
DTR	// 整数(32 ビット)を浮動小数点数に変換。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。

INVI: 1 の補数整数(16 ビット)の作成

説明

「1 の補数整数(16 ビット)の作成」命令を使用して、アキュムレータ 1 の右ワードの値をビット対ビットで反転します。

処理中に、命令はアキュムレータ 1 の右ワードの個々のビットのシグナル状態を反転させます。1 は 0 に置き換えられ、0 は 1 に置き換えられます。

結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードの内容は、命令に影響されません。

構文

「1 の補数整数(16 ビット)の作成」命令には、以下の構文を使用します。

```
INVI
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 値をアキュムレータ 1 にロード
INVI	// 1 の補数を作成
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	0011	1010	1110
Tag_Output	1001	1100	0101	0001

INVD: 1 の補数倍精度整数(32 ビット)の作成

説明

「1 の補数整数(32 ビット)の作成」命令を使用して、アキュムレータ 1 の値をビット対ビットで反転します。

処理中に、命令はアキュムレータ 1 の個々のビットのシグナル状態を反転させます。1 は 0 に置き換えられ、0 は 1 に置き換えられます。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

構文

「1 の補数整数(32 ビット)の作成」命令には、以下の構文を使用します。

```
INVD
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 値をアキュムレータ 1 にロード
INVD	// 1 の補数を作成
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0110	1111	1000	1100	0110	0011	1010	1110
Tag_Output	1001	0000	0111	0011	1001	1100	0101	0001

NEGI: 負の整数(16 ビット)

説明

「負の整数(16 ビット)」命令を使用して、アキュムレータ 1 の右ワードの値の符号を 2 の補数を形成することによって変更します。 命令はアキュムレータ 1 の値を 16 ビット整数として解釈します。

命令の処理は、「-1」での乗算と等しくなります。 変換の結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードの内容は、命令に影響されません。

「負の整数(16 ビット)」命令はステータスビット CC 0、CC 1、OV、および OS に影響します。 次の表に、命令が結果によってステータスビットに影響するようすを示します。

結果:	CC 1	CC 0	OV	OS
+1 ~ +32 767	1	0	0	-
0	0	0	0	-
-1 ~ -32 767	0	1	0	-
(-) 32 768	0	1	1	1

構文

以下の構文が「負の整数(16 ビット)」命令に使用されます。

```
NEGI
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 値をアキュムレータ 1 にロード
NEGI	// 負の整数(16 ビット)
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1101	0011	1000
Tag_Output	1010	0010	1100	1000

NEGD: 負の整数(32 ビット)

説明

「負の整数(32 ビット)」命令を使用して、アキュムレータ 1 の値の符号を 2 の補数を形成することによって変更します。 命令はアキュムレータ 1 の値を 32 ビット整数として解釈します。

命令の処理は、「-1」での乗算と等しくなります。 命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

「負の整数(32 ビット)」命令はステータスビット CC 0、CC 1、OV、および OS に影響します。 次の表に、命令が結果によってステータスビットに影響するようすを示します。

結果	CC 1	CC 0	OV	OS
+1 ~ +2147483647	1	0	0	-
0	0	0	0	-
-1 ~ -2147483647	0	1	0	-
(-) 2147483648	0	1	1	1

構文

以下の構文が「負の整数(32 ビット)」命令に使用されます。

```
NEGD
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 値をアキュムレータ 1 にロード
NEGD	// 負の整数(32 ビット)。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

ステータス	値							
	31 . . .	. . . .	. . . .	. . . 16	15 . . .	. . . .	. . . .	. . . 0
Tag_Input	0101	1111	0110	0100	0101	1101	0011	1000
Tag_Output	1010	0000	1001	1011	1010	0010	1100	1000

NEGR: 負の浮動小数点数

説明

「負の浮動小数点数」命令は、仮数の符号を反転します。この命令は、「-1」による乗算と同じです。

この命令は、アキュムレータ 1 の値を浮動小数点数として解釈します。 仮数の符号は、浮動小数点数のビット 31 に入っています。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

この命令はステータスビットには影響しません。

構文

「負の浮動小数点数」命令には、以下の構文を使用します。

```
NEGR
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 浮動小数点数をアキュムレータ 1 にロード
NEGR	// 負の浮動小数点数。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Input	1.5E+02
Tag_Output	-1.5E+02

CAW: アキュムレータ 1 の右ワードのバイトを切り替える

説明

「アキュムレータ 1 の右ワードのバイトを切り替える」命令を使用して、アキュムレータ 1 の右ワードの右側の 2 つのバイトの順番を切り替えます。

次の表に、命令の処理前および処理後のアキュムレータ 1 の内容を示します。

ステータス	アキュムレータ 1 のバイト			
処理前	A	B	C	D
処理後	A	B	D	C

命令の結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードのバイトは命令に影響されず、変わりません。

構文

「アキュムレータ 1 の右ワードのバイトを切り替える」命令には、以下の構文を使用します。

```
CAW
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 値をアキュムレータにロード。
CAW	// アキュムレータ 1 の右ワードのバイトを切り替える。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
Tag_Input	0000	1111	0000	1111
Tag_Output	0000	1111	1111	0000

CAD: アキュムレータ 1 のすべてのバイトを切り替える

説明

「アキュムレータ 1 のすべてのバイトを切り替える」命令を使用して、アキュムレータ 1 のバイトの順番を入れ替えます。

次の表に、命令の処理前および処理後のアキュムレータ 1 の内容を示します。

ステータス	アキュムレータ 1 のバイト			
処理前	A	B	C	D
処理後	D	C	B	A

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

構文

「アキュムレータ 1 のすべてのバイトを切り替える」命令には、以下の構文を使用します。

```
CAD
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// オペランドの値をアキュムレータ 1 にロード。
CAD	// アキュムレータ 1 のバイトを入れ替え。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Input	1111	0000	0000	1111	0000	0000	1111	1111
Tag_Output	1111	1111	0000	0000	0000	1111	1111	0000

RND: 数値の四捨五入

説明

「数値の四捨五入」命令を使用して、アキュムレータ 1 の値を 32 ビット整数に変換します。 この命令は、変換する値を浮動小数点数として解釈し、次の整数に四捨五入します。

浮動小数点数がちょうど偶数と奇数の中間にある場合は、結果として偶数が選択されます。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

次のいずれかの条件に該当する場合、変換は実行されず、ステータスビット OV および OS が設定されます。

- アキュムレータ 1 の値が有効な浮動小数点数でない。
- 結果が、データタイプ DINT の整数に許可されている範囲を超えている。

構文

「数値の四捨五入」命令には、以下の構文を使用します。

```
RND
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 浮動小数点数をアキュムレータ 1 にロード
RND	// 浮動小数点数を整数に四捨五入。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Input	101.5	-101.5
Tag_Output	102	-102

TRUNC: 数値の切り捨て

説明

「数値を切り捨てる」命令を使用して、アキュムレータ 1 の値を 32 ビット整数に変換します。 この命令は、変換する値を浮動小数点数として解釈し、結果として整数を返します。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

次のいずれかの条件に該当する場合、変換は実行されず、ステータスビット OV および OS が設定されます。

- アキュムレータ 1 の値が有効な浮動小数点数でない。
- 結果が、データタイプ DINT の整数に許可されている範囲を超えている。

構文

「数値を切り捨てる」命令には、以下の構文を使用します。

```
TRUNC
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 浮動小数点数をアキュムレータ 1 にロード
TRUNC	// 浮動小数点数をデータタイプ DINT の整数に変換。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Input	101.5	-101.5
Tag_Output	101	-101

RND+: 浮動小数点数から次に大きい整数を生成

説明

「浮動小数点数から次に大きい整数を生成」命令を使用して、アキュムレータ 1 の値を 32 ビット整数に変換します。 この命令は、変換する値を浮動小数点数として解釈し、次に大きい整数に四捨五入します。 命令の結果は、変換する浮動小数点数以上のデータタイプ DINT の数値です。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

次のいずれかの条件に該当する場合、変換は実行されず、ステータスビット OV および OS が設定されます。

- アキュムレータ 1 の値が有効な浮動小数点数でない。
- 結果が、データタイプ DINT の整数に許可されている範囲を超えている。

構文

「浮動小数点数から次に大きい整数を生成」命令には、次の構文を使用します。

```
RND+
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 浮動小数点数をアキュムレータ 1 にロード
RND+	// 浮動小数点数を次に大きい整数に四捨五入。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Input	100.5	-100.5
Tag_Output	101	-100

RND-: 浮動小数点数から次に小さい整数を生成

説明

「浮動小数点数から次に小さい整数を生成」命令を使用して、アキュムレータ 1 の値を 32 ビット整数に変換します。 この命令は、変換する値を浮動小数点数として解釈し、浮動小数点数の値を次に小さい整数に四捨五入します。 命令の結果は、変換する浮動小数点数以下のデータタイプ DINT の数値です。

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

次のいずれかの条件に該当する場合、変換は実行されず、ステータスビット OV および OS が設定されます。

- アキュムレータ 1 の値が有効な浮動小数点数でない。
- 結果が、データタイプ DINT の整数に許可されている範囲を超えている。

構文

「浮動小数点数から次に小さい整数を生成」命令には、次の構文を使用します。

```
RND-
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input"	// 浮動小数点数をアキュムレータ 1 にロード
RND-	// 浮動小数点数を次に小さい整数に四捨五入。
T "Tag_Output"	// 結果をオペランド「Tag_Output」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Input	100.5	-100.5
Tag_Output	100	-101

プログラム制御演算

この章には下記に関する情報が記載されています：

- ジャンプ (S7-1500)
- データブロック (S7-1500)
- プログラムブロック (S7-1500)

ジャンプ

この章には下記に関する情報が記載されています：

ジャンプラベル

説明

ジャンプラベルを使用して、ジャンプ後にプログラムの実行が再開されるプログラム内の場所を指定します。 ジャンプラベルの名前は、128 個までの文字、数字、アンダースコアで構成することができます。

ジャンプラベルとジャンプラベルを指定する命令は、同じブロックに置く必要があります。 ジャンプラベルの名前は、ブロック内で 1 回のみ割り当てることができます。 各ジャンプラベルから複数の位置にジャンプできます。 最大 256 のジャンプラベルを宣言できます。

ジャンプは順方向と逆方向のどちらも可能です。

構文

ジャンプラベルには、以下の構文を使用します。

```
<Jump label>:
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// 最初の比較値をロード。
L "Tag_Input_2"	// 2 番目の比較値をロード。
>I	// オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかをチェック。
JC MyLABEL	// スキャン結果が「1」の場合、ジャンプラベル「MyLABEL」にジャンプし、プログラムの処理を続行
	// 問い合わせの結果が「0」の場合、次の命令を実行。
L "Tag_Input_3"	// オペランド「Tag_Input_3」の内容をアキュムレータ 1 にロード。
T "Tag_Output"	// アキュムレータ 1 の内容をオペランド「Tag_Output」にロード。
MyLABEL: A "Tag_Input_4"	// ジャンプがあった場合、この箇所でプログラム実行を続行。
	// オペランド「Tag_Input_4」が「1」かどうかを照会

JU: 条件なしジャンプ

説明

「条件なしジャンプ」命令を使用して、線形プログラム実行を解釈し、指定されたジャンプラベルによってマーキングされた箇所で続行します。 この命令は条件に関係なく常に実行されます。

「条件なしジャンプ」命令はステータスビットには影響しません。

構文

「条件なしジャンプ」命令には、以下の構文を使用します。

```
JU <jump label>
```

パラメータ

次の表に、「条件なしジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// 最初の比較値をロード。
L "Tag_Input_2"	// 2 番目の比較値をロード。
>I	// オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかをチェック。
JC MyLABEL_1	// スキャン結果が「1」の場合、ジャンプラベル「MyLABEL_1」にジャンプし、プログラムの処理を続行
	// 照会の結果が「0」の場合、線形プログラムの実行を続行。
L "Tag_Input_3"	// オペランド「Tag_Input_3」の内容をアキュムレータ 1 にロード。
T "Tag_Output_1"	// アキュムレータ 1 の内容をオペランド「Tag_Output_1」にロード。
JU MyLABEL_2	// ジャンプラベル「MyLABEL_2」にジャンプし、プログラムの実行をそこから再開。
MyLABEL_1:	// ジャンプラベル
L "Tag_Input_4"	// オペランド「Tag_Input_4」の内容をアキュムレータ 1 にロード。
T "Tag_Output_2"	// アキュムレータ 1 の内容をオペランド「Tag_Output_2」にロード。

```
MyLABEL_2: A "Tag_Input_5"
```

```
// ジャンプラベル「MyLABEL_2」
// オペランド「Tag_Input_5」が「1」かどうかを照会
```

JC:RLO = 1 ならばジャンプ

説明

「RLO = 1 ならばジャンプ」命令を使用して、線形プログラム実行を論理演算の結果に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。

指定されたジャンプラベルへのジャンプは、現在の RLO が「1」の場合にのみ実行されます。 命令前の現在の RLO が「0」の場合、ジャンプは実行されずプログラムは次の命令で再開されます。

「RLO = 1 ならばジャンプ」命令は、条件が満たされている場合、満たされていない場合を問わず RLO を「1」にセットします。

構文

「RLO = 1 ならばジャンプ」命令には、以下の構文を使用します。

```
JC <jump label>
```

パラメータ

次の表に、「RLO = 1 ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// 最初の比較値をロード。
L "Tag_Input_2"	// 2 番目の比較値をロード。
>I	// オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかをチェック。
JC MyLABEL	// RLO = 「1」の場合、ジャンプラベル「MyLABEL」にジャンプし、そこでプログラム実行を続行。
	// RLO = 「0」の場合、次の命令を実行。
L "Tag_Input_3"	// オペランド「Tag_Input_3」の内容をアキュムレータ 1 にロード。
T "Tag_Output_1"	// アキュムレータ 1 の内容をオペランド「Tag_Output_1」にロード。
	// ジャンプがあった場合、この箇所でプログラム実行を続行。
MyLABEL: L "Tag_Input_4"	// オペランド「Tag_Input_4」の内容をアキュムレータ 1 にロード。

T "Tag_Output_2"	// アキュムレータ 1 の内容をオペランド「Tag_Output_2」に転送。

JCN: RLO = 0 ならばジャンプ

説明

「RLO = 0 ならばジャンプ」命令を使用して、線形プログラム実行を論理演算の結果(RLO)に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。

指定されたジャンプラベルへのジャンプは、現在の RLO が「0」の場合にのみ実行されます。 命令前の現在の RLO が「1」の場合、ジャンプは実行されずプログラムは次の命令で再開されます。

「RLO = 0 ならばジャンプ」命令は、条件が満たされている場合、満たされていない場合を問わず RLO を「1」にセットします。

構文

「RLO = 0 ならばジャンプ」命令には、以下の構文を使用します。

```
JCN <jump label>
```

パラメータ

次の表に、「RLO = 0 ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランド「Tag_Input_1」の値が「1」かどうかを照会し、論理積演算。
A "Tag_Input_2"	// オペランド「Tag_Input_2」の値が「1」かどうかを照会し、論理積演算。
JCN MyLABEL	// RLO = 「0」の場合、ジャンプラベル「MyLABEL」にジャンプし、そこでプログラム実行を続行。
	// RLO = 「1」の場合、次の命令を実行。
L "Tag_Input_3"	// オペランド「Tag_Input_3」の内容をアキュムレータ 1 にロード。
T "Tag_Output"	// アキュムレータ 1 の内容をオペランド「Tag_Output」にロード。
	// ジャンプがあった場合、この箇所でプログラム実行を続行。
MyLABEL: A "Tag_Input_4"	// オペランド「Tag_Input_4」の値が「1」かどうかを照会し、RLO で論理積演算

JCB:RLO = 1 ならばジャンプして RLO を保存

説明

「RLO = 1 ならばジャンプして RLO を保存」命令を使用して、線形プログラム実行を論理演算の結果(RLO)に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。 同時に、現在の RLO のシグナル状態がバイナリ結果(BR)にコピーされます。

指定されたジャンプラベルへのジャンプは、現在の RLO が「1」の場合にのみ実行されます。 この場合、パラメータがバイナリ結果「1」にセットされます。

命令前の現在の RLO が「0」の場合、ジャンプは実行されずプログラムは次の命令で再開されます。この場合、命令はバイナリ結果にシグナル状態「0」を割り当てます。

「RLO = 1 ならばジャンプして RLO を保存」命令は、条件が満たされている場合、満たされていない場合を問わず RLO を「1」にセットします。

構文

「RLO = 1 ならばジャンプして RLO を保存」命令には、以下の構文を使用します。

```
JCB <jump label>
```

パラメータ

次の表に、「RLO = 1 ならばジャンプして RLO を保存」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Input_1"	// 最初の比較値をロード。
L "Tag_Input_2"	// 2 番目の比較値をロード。
>I	// オペランド「Tag_Input_1」の値がオペランド「Tag_Input_2」の値よりも大きいかどうかをチェック。
	// 現在の RLO を BR にコピー。
JCB MyLABEL	// RLO = 「1」の場合、ジャンプラベル「MyLABEL」にジャンプし、そこでプログラム実行を続行。
	// RLO = 「0」の場合、次の命令を実行。
L "Tag_Input_3"	// オペランド「Tag_Input_3」の内容をアキュムレータ 1 にロード。
T "Tag_Output_1"	// アキュムレータ 1 の内容をオペランド「Tag_Output_1」にロード。

```
                        // ジャンプがあった場合、この箇所でプログラム実行を続行。
MyLABEL: L "Tag_Input_4"  // オペランド「Tag_Input_4」の内容をアキュムレータ 1 にロード。
T "Tag_Output_2"          // アキュムレータ 1 の内容をオペランド「Tag_Output_2」にロード。
```

JNB: RLO = 0 ならばジャンプして RLO を保存

説明

「RLO = 0 ならばジャンプして RLO を保存」命令を使用して、線形プログラム実行を論理演算の結果(RLO)に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。 同時に、現在の論理演算の結果のシグナル状態がバイナリ結果(BR)にコピーされます。

指定されたジャンプラベルへのジャンプは、現在の RLO が「0」の場合にのみ実行されます。 この場合、命令はバイナリ結果を「0」にセットします。

命令前の現在の RLO が「1」の場合、ジャンプは実行されずプログラムは次の命令で再開されます。この場合、命令はバイナリ結果にシグナル状態「1」を割り当てます。

「RLO = 0 ならばジャンプして RLO を保存」命令は、条件が満たされている場合、満たされていない場合を問わず RLO を「1」にセットします。

構文

「RLO = 0 ならばジャンプして RLO を保存」命令には、以下の構文を使用します。

```
JNB <jump label>
```

パラメータ

次の表に、「RLO = 0 ならばジャンプして RLO を保存」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランド「Tag_Input_1」の値が「1」かどうかを照会し、論理積演算。
A "Tag_Input_2"	// オペランド「Tag_Input_2」の値が「1」かどうかを照会し、論理積演算。
	// 現在の RLO を BR にコピー。
JNB MyLABEL	// RLO = 「0」の場合、ジャンプラベル「MyLABEL」にジャンプし、そこでプログラム実行を続行。
	// RLO = 「1」の場合、次の命令を実行。
L "Tag_Input_3"	// オペランド「Tag_Input_3」の内容をアキュムレータ 1 にロード。
T "Tag_Output"	// アキュムレータ 1 の内容をオペランド「Tag_Output」に転送。
MyLABEL: A "Tag_Input_4"	// ジャンプがあった場合、この箇所でプログラム実行を続行。

```
                    // オペランド「Tag_Input_4」の値が「1」かどうかを照会し、
                    RLO で論理積演算
```

JBI:BR = 1 ならばジャンプ

説明

「BR = 1 ならばジャンプ」命令を使用して、線形プログラム実行をバイナリ結果に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。

指定されたジャンプラベルへのジャンプは、ステータスビット BR が「1」の場合にのみ実行されます。 ステータスビット BR が「0」の場合、ジャンプは実行されずプログラムは次の命令で再開します。

構文

「BR = 1 ならばジャンプ」命令には、以下の構文を使用します。

```
JBI <jump label>
```

パラメータ

次の表に、「BR = 1 ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
SET	// ブロックの開始で RLO を「1」にセット。
SAVE	// RLO のシグナル状態を BR ビットに転送。
....	// 任意のプログラム
ANOV	// ブロックの終了で OV ビットを照会。
SAVE	// オーバーフローがない場合、BR ビットを「0」にセット。 // オーバーフローがない場合 BR を「1」にセット。
JBI END	// BR = 「1」の場合、ジャンプラベル END にジャンプ。 // BR = 「0」の場合、次の命令を実行。
R "Tag_Output_1"	// オペランド「Tag_Output_1」を「0」にリセット。
END: S "Tag_Output_2"	// オペランド「Tag_Output_2」を「1」にセット

JNBI:BR = 0 ならばジャンプ

説明

「BR = 0 ならばジャンプ」命令を使用して、線形プログラム実行をバイナリ結果(BR)に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。

指定されたジャンプラベルへのジャンプは、ステータスビット BR が「0」の場合にのみ実行されます。 命令の前のステータスビット BR が「1」の場合、ジャンプは実行されずプログラムは次の命令で再開します。

構文

「BR = 0 ならばジャンプ」命令には、以下の構文を使用します。

```
JNBI <jump label>
```

パラメータ

次の表に、「BR = 0 ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
SET	// ブロックの開始で RLO を「1」にセット。
SAVE	// RLO のシグナル状態を BR ビットに転送。
....	// 任意のプログラム
ANOV	// ブロックの終了で OV ビットを照会。
SAVE	// オーバーフローがない場合、BR ビットを「0」にセット。 // オーバーフローがない場合 BR を「1」にセット。
JNBI END	// BR = 「0」の場合、ジャンプラベル END にジャンプ。 // BR = 「1」の場合、次の命令を実行。
S "Tag_Output"	// オペランド「Tag_Output」を「1」にセット
END: BE	// ブロックの終了

JO:OV = 1 ならばジャンプ

説明

「OV = 1 ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット OV に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。

指定されたジャンプラベルへのジャンプは、ステータスビット OV が「1」の場合にのみ実行されます。 ステータスビット OV は、たとえば計算の結果が許容範囲を超えている場合や浮動小数点数の比較中に無効な値が返された場合にセットされます。

ステータスビット OV が「0」の場合、ジャンプは実行されずプログラムは次の命令で続行します。

構文

「OV = 1 ならばジャンプ」命令には、以下の構文を使用します。

```
JO <jump label>
```

パラメータ

次の表に、「OV = 1 ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 値をロード
L "Tag_Value_2"	// 値をロード
*I	// 値を乗算
JO OVER	// ステータスビット OV のシグナル状態が「1」の場合、ジャンプラベル「OVER」にジャンプし、そこでプログラム実行を続行。 // ステータスビット OV のシグナル状態が「0」の場合、次の命令を実行。
T "Tag_Result"	// 乗算結果をオペランド「Tag_Result」に転送。
JU NEXT	// ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
OVER: SET	// 条件(OV = 「1」)を満たす場合、この箇所でプログラム実行を続行。 // RLO をシグナル状態「1」にセット
R "Tag_Output"	// オペランド「Tag_Output」を「0」にリセット。

```
                        // ジャンプラベル「NEXT」
NEXT: A "MyTag_1"       // オペランド「MyTag_1」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_2"             // オペランド「MyTag_2」の値が「1」かどうかを照会し、論理積演算。
S "Tag_Output_2"        // RLO = 1 の場合、オペランド「Tag_Output_2」を「1」にセット。
```

JOS:OS = 1 のときにジャンプ

説明

「OS = 1 ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット OS に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所で処理を再開します。

指定されたジャンプラベルへのジャンプは、ステータスビット OS が「1」の場合にのみ実行されます。 ステータスビット OS は、番号範囲オーバーフローがステータスビット OV を「1」にセットすると常にセットされます。 ステータスビット OV とは異なり、その後の結果が数値の許容範囲内であったとしてもステータスビット OS はセットされた状態となります。

ステータスビット OS が「0」の場合、ジャンプは実行されずプログラムは次の命令で続行します。

構文

「OS = 1 ならばジャンプ」命令には、以下の構文を使用します。

```
JOS <jump label>
```

パラメータ

次の表に、「OS = 1 ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// 値をロード
L "Tag_Value_2"	// 値をロード
*I	// ロード値を乗算。 // 積をアキュムレータ 1 に保存。
L "Tag_Value_3"	// 値をロード
+I	// ロード値を積に加算。 // 積をアキュムレータ 1 に保存。
L "Tag_Value_4"	// 値をロード
-I	// 計算済み合計からロード値を減算。 // 積をアキュムレータ 1 に保存。
JOS OVER	// 先行する 3 つの命令のいずれか 1 つでオーバーフローが発生する場合、ステータスビット OS は「1」にセットされます。

```
                        // ステータスビット OS のシグナル状態が「1」の場合、プログラムはジャンプラベル「OVER」から再開します。
                        // ステータスビット OS のシグナル状態が「0」の場合、次の命令を実行。
T "Tag_Result"          // 計算全体の結果をオペランド「Tag_Result」に転送
JU NEXT                 // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
                        // 条件(OV = 「1」)を満たす場合、この箇所でプログラム実行を続行。
OVER: SET               // RLO をシグナル状態「1」にセット
R "Tag_Output"          // オペランド「Tag_Output」を「0」にリセット。
                        // ジャンプラベル「NEXT」
NEXT: A "MyTag_1"       // オペランド「MyTag_1」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_2"             // オペランド「MyTag_2」の値が「1」かどうかを照会し、論理積演算。
S "Tag_Output_2"        // RLO = 1 の場合、オペランド「Tag_Output_2」を「1」にセット。
```

JZ: 結果がゼロならばジャンプ

説明

「結果が 0 ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 0 および CC 1 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 0 と CC 1 の両方のシグナル状態が「0」の場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- オーバーフローなしで算術命令を処理した後に、アキュムレータ 1 の値が「0」であること。
- 「整数加算(16 ビット)」(+I)命令または「整数加算(32 bit)」(+D)命令の実行時に負のオーバーフローが発生していること。
- 算術命令(浮動小数点計算)がオーバーフローありで処理されているときに、この値が次第に許容範囲を下回って低下していること。
- アキュムレータ 2 の内容が、比較命令の実行後にアキュムレータ 1 の内容に等しいこと。
- アキュムレータ 1 の内容が、ワード論理演算の実行後にゼロであること。
- シフト命令の実行後、最後にシフトアウトされたビットの値が「0」であること。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果がゼロならばジャンプ」命令には、以下の構文を使用します。

```
JZ <jump label>
```

パラメータ

次の表に、「結果がゼロならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// オペランド「Tag_Value」の値をロード。
SRW 1	// アキュムレータ 1 の右ワードの内容を右へ場所を 1 つシフトする。
JZ ZERO	// 最後にシフトされたビットのシグナル状態が「0」の場合、ジャンプラベル「OVER」へジャンプし、そこからプログラム処理を続行

```
                        // 最後にシフトされたビットのシグナル状態が「1」の場合、次の命令を実行。
L "MyTag_1"             // オペランド「MyTag_1」の値をアキュムレータ 1 にロード
INC 1                   // アキュムレータ 1 の右側のバイトに値「1」を加算。
T "Tag_Result_1"        // 結果をオペランド「Tag_Result_1」に転送
JU NEXT                 // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
ZERO: L "MyTag_2"       // ジャンプラベル「ZERO」
                        // オペランド「MyTag_2」の値をアキュムレータ 1 にロード
INC 1                   // アキュムレータ 1 の右側のバイトに値「1」を加算。
T "Tag_Result_2"        // 結果をオペランド「Tag_Result_2」に転送。
NEXT: A "MyTag_3"       // ジャンプラベル「NEXT」
                        // オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"             // オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。
```

JN: 結果がゼロでないならばジャンプ

説明

「結果が 0 でないならばジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 0 および CC 1 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 0 および CC 1 のシグナル状態が異なる場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- オーバーフローなしで算術命令を処理した後に、アキュムレータ 1 の値がゼロに等しくないこと。
- 整数(+I、-I、*I、+D、-D、*D)によって算術命令を処理したときに、負のオーバーフローが発生していること。
- 整数(+I、-I、*I、/I、+D、-D、*D、/D、NEGI、NEGD)によって算術命令を実行したときに、正のオーバーフローが発生していること。
- 算術命令を浮動小数点数によって処理したときに、正または負のオーバーフローが発生していること。
- アキュムレータ 2 の内容が、比較命令の実行後にアキュムレータ 1 の内容に等しくないこと。
- アキュムレータ 1 の内容が、ワード論理演算の実行後にゼロでないこと。
- シフト命令の実行後、最後にシフトアウトされたビットの値が「1」であること。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果がゼロでないならばジャンプ」命令には、以下の構文を使用します。

```
JN <jump label>
```

パラメータ

次の表に、「結果がゼロでないならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Valu_1」の値をロード。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をロード。
XOW	// 排他的論理和演算を適用

```
                                  // アキュムレータ 1 の値がゼロに等しくない場合、「NOZERO」ジャンプラベルへのジャンプが実行されプログラム処理はそこから続行。
JN NOZERO
                                  // アキュムレータ 1 の値がゼロに等しい場合、次の命令が処理される。
AN "MyTag_1"                      // オペランド「MyTag_1」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_1"                  // RLO=「1」の場合、オペランド「Tag_Output_1」を「1」にセット。
JU NEXT                           // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
                                  // ジャンプラベル「NOZERO」
NOZERO: AN "MyTag_2"              // オペランド「MyTag_2」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_2"                  // RLO=「1」の場合、オペランド「Tag_Output_2」を「1」にセット。
                                  // ジャンプラベル「NEXT」
NEXT: A "MyTag_3"                 // オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"                       // オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。
```

JP: 結果がゼロよりも大きければジャンプ

説明

「結果がゼロよりも大きければジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 0 および CC 1 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 0 のシグナル状態が「0」、かつ CC 1 のシグナル状態が「1」の場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- オーバーフローなしで算術命令を処理した後に、アキュムレータ 1 の値がゼロよりも大きいこと。
- 整数(+I、-I、*I、+D、-D、*D)によって算術命令を処理したときに、負のオーバーフローが発生していること。
- 整数(*I、/I、*D、/D)によって算術命令を処理したときに、正のオーバーフローが発生していること。
- 算術命令を浮動小数点数によって処理したときに、正のオーバーフローが発生していること。
- アキュムレータ 2 の内容が、比較命令の実行後にアキュムレータ 1 の内容よりも大きいこと。
- アキュムレータ 1 の内容が、ワード論理演算の実行後にゼロでないこと。
- シフト命令の実行後、最後にシフトアウトされたビットの値が「1」であること。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果がゼロより大きいならばジャンプ」命令には、以下の構文を使用します。

```
JP <jump label>
```

パラメータ

次の表に、「結果がゼロよりも大きければジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をロード。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をロード。
-I	// オペランド「Tag_Value_2」の値をオペランド「Tag_Value_1」の値から減算。

```
JP POSITIVE                 // アキュムレータ「1」に正の結果値があると、ジャンプラベル「POSITIVE」にジャンプし、そこからプログラム処理を続行
                            // アキュムレータ 1 の結果が負の場合、次の命令を実行。
AN "MyTag_1"                // オペランド「MyTag_1」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_1"            // RLO=「1」の場合、オペランド「Tag_Output_1」を「1」にセット。
JU NEXT                     // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
                            // ジャンプラベル「POSITIVE」
POSITIVE: AN "MyTag_2"      // オペランド「MyTag_2」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_2"            // RLO=「1」の場合、オペランド「Tag_Output_2」を「1」にセット。
                            // ジャンプラベル「NEXT」
NEXT: A "MyTag_3"           // オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"                 // オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。
```

JM: 結果がゼロ以下ならばジャンプ

説明

「結果がゼロ未満ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 0 および CC 1 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 0 のシグナル状態が「1」、かつ CC 1 のシグナル状態が「0」の場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- オーバーフローなしで算術命令を処理した後に、アキュムレータ 1 の値がゼロ未満であること。
- 整数(*I、*D)によって算術命令を処理したときに、負のオーバーフローが発生していること。
- 整数(+I、-I、+D、-D、NEGI、NEGD)によって算術命令を処理したときに、正のオーバーフローが発生していること。
- 算術命令を浮動小数点数によって処理したときに、負のオーバーフローが発生していること。
- アキュムレータ 2 の内容が、比較命令の実行後にアキュムレータ 1 の内容未満であること。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果がゼロ以下ならばジャンプ」命令には、以下の構文を使用します。

```
JM <jump label>
```

パラメータ

次の表に、「結果がゼロ未満ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をロード。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をロード。
-I	// オペランド「Tag_Value_2」の値をオペランド「Tag_Value_1」の値から減算。
JM NEGATIVE	// アキュムレータ「1」に負の結果値があると、ジャンプラベル「NEGATIVE」にジャンプし、そこからプログラム処理を続行 // アキュムレータ 1 の結果が正の場合、次の命令を実行。

```
AN "MyTag_1"                // オペランド「MyTag_1」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_1"            // RLO=「1」の場合、オペランド「Tag_Output_1」を「1」にセット。
JU NEXT                     // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
                            // ジャンプラベル「NEGATIVE」
NEGATIVE: AN "MyTag_2"      // オペランド「MyTag_2」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_2"            // RLO=「1」の場合、オペランド「Tag_Output_2」を「1」にセット。
                            // ジャンプラベル「NEXT」
NEXT: A "MyTag_3"           // オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"                 // オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。
```

JPZ: 結果がゼロ以上ならばジャンプ

説明

「結果がゼロ以上ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 0 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 0 のシグナル状態が「0」の場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- オーバーフローなしで算術命令を処理した後に、アキュムレータ 1 の値がゼロ以上であること。
- 整数(+I、-I、*I、+D、-D、*D)によって算術命令を処理したときに、負のオーバーフローが発生していること。
- 整数(*I、/I、*D、/D)によって算術命令を処理したときに、正のオーバーフローが発生していること。
- 算術命令を浮動小数点数によって処理したときに、正のオーバーフローが発生していること。
- 算術命令(浮動小数点計算)がオーバーフローありで処理されているときに、この値が次第に許容範囲を下回って低下していること。
- アキュムレータ 2 の内容が、比較命令の実行後にアキュムレータ 1 の内容以上であること。
- ワード論理演算の命令が実行された。
- シフト命令が実行された。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果がゼロ以上ならばジャンプ」命令には、以下の構文を使用します。

```
JPZ <jump label>
```

パラメータ

次の表に、「結果がゼロ以上ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をロード。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をロード。
-I	// オペランド「Tag_Value_2」の値をオペランド「Tag_Value_1」の値から減算。

JPZ REGULAR	// アキュムレータ「1」に正の結果値または Null があると、ジャンプラベル「REGULAR」にジャンプし、そこからプログラム処理を続行
	// アキュムレータ 1 の結果が負の場合、次の命令を実行。
AN "MyTag_1"	// オペランド「MyTag_1」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_1"	// RLO=「1」の場合、オペランド「Tag_Output_1」を「1」にセット。
JU NEXT	// ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
	// ジャンプラベル「REGULAR」
REGULAR: AN "MyTag_2"	// オペランド「MyTag_2」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_2"	// RLO=「1」の場合、オペランド「Tag_Output_2」を「1」にセット。
	// ジャンプラベル「NEXT」
NEXT: A "MyTag_3"	// オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"	// オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。

JMZ: 結果がゼロ以下ならばジャンプ

説明

「結果がゼロ以下ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 1 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 1 のシグナル状態が「0」の場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- オーバーフローなしで算術命令を処理した後に、アキュムレータ 1 の値がゼロ以下であること。
- 整数(+I、*I、+D、*D)によって算術命令を処理したときに、負のオーバーフローが発生していること。
- 整数(+I、-I、+D、-D、NEGI、NEGD)によって算術命令を処理したときに、正のオーバーフローが発生していること。
- 算術命令を浮動小数点数によって処理したときに、負のオーバーフローが発生していること。
- 算術命令(浮動小数点計算)がオーバーフローありで処理されているときに、この値が次第に許容範囲を下回って低下していること。
- アキュムレータ 2 の内容が、比較命令の実行後にアキュムレータ 1 の内容以下であること。
- アキュムレータ 1 の値がワード論理の処理後ゼロに等しいこと。
- 書き込み命令の処理後、最後にシフトされたビットの値がゼロに等しいこと。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果がゼロ以下ならばジャンプ」命令には、以下の構文を使用します。

```
JMZ <jump label>
```

パラメータ

次の表に、「結果がゼロ以下ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をロード。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をロード。

```
-I                          // オペランド「Tag_Value_2」の値をオペランド「Tag_Value_1」の値から減算。
JMZ MyLABEL                 // アキュムレータ「1」に負の結果値または Null があると、ジャンプラベル「MyLABEL」にジャンプし、そこからプログラム処理を続行
                            // アキュムレータ 1 の結果が正の場合、次の命令を実行。
AN "MyTag_1"                // オペランド「MyTag_1」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_1"            // RLO=「1」の場合、オペランド「Tag_Output_1」を「1」にセット。
JU NEXT                     // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
                            // ジャンプラベル「MyLABEL」
MyLABEL: AN "MyTag_2"       // オペランド「MyTag_2」の値が「0」かどうかを照会し、論理積演算。
S "Tag_Output_2"            // RLO=「1」の場合、オペランド「Tag_Output_2」を「1」にセット
                            // ジャンプラベル「NEXT」
NEXT: A "MyTag_3"           // オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"                 // オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。
```

JUO: 結果が無効ならばジャンプ

説明

「結果が無効ならばジャンプ」命令を使用して、線形プログラム実行をステータスビット CC 0 および CC 1 に応じて割り込みし、指定されたジャンプラベルによってマーキングされた箇所から続行します。

指定されたジャンプラベルへのジャンプは、ステータスビット CC 0 と CC 1 の両方のシグナル状態が「1」の場合にのみ実行されます。 この状況は、次の条件の 1 つに該当すると発生します。

- 算術命令でゼロによって除算されること(/I、/D、MOD)。
- 算術命令を浮動小数点数によって処理したときに、オーバーフローが発生して結果値が無効な浮動小数点数になること。
- 比較命令を浮動小数点数によって処理するとき、無効な浮動小数点数が使用されたか、または結果値として発生していること。

それ以外のすべての場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。

構文

「結果が無効ならばジャンプ」命令には、以下の構文を使用します。

```
JUO <jump label>
```

パラメータ

次の表に、「結果が無効ならばジャンプ」命令のパラメータを示します。

パラメータ	説明
<Jump label>	ジャンプ先のシンボル名

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をロード。
L "Tag_Value_2"	// オペランド「Tag_Value_2」の値をロード。
/I	// オペランド「Tag_Value_1」の値をオペランド「Tag_Value_2」の値で除算。
JUO ERROR	// ゼロによる除算で、ジャンプラベル「ERROR」にジャンプしプログラム処理をそこから続行
	// それ以外の場合、次の命令を実行。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランド「Tag_Result」に転送。
A "MyTag_1"	// オペランド「MyTag_1」の値が「1」かどうかを照会し、論理積演算。

```
R "MyTag_1"              // RLO=「1」の場合、オペランド「MyTag_1」を「0」にリセット。
JU NEXT                  // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
                         // ジャンプラベル「ERROR」
ERROR: AN "MyTag_1"      // オペランド「MyTag_1」の値が「0」かどうかを照会し、論理積演算。
S "MyTag_1"              // RLO=「1」の場合、オペランド「MyTag_1」を「1」にセット。
                         // ジャンプラベル「NEXT」
NEXT: A "MyTag_3"        // オペランド「MyTag_3」の値が「1」かどうかを照会し、論理積演算。
A "MyTag_4"              // オペランド「MyTag_4」の値が「1」かどうかを照会し、論理積演算。
```

JL: ジャンプリストの定義

説明

「ジャンプリスト定義」命令を使用して、「条件なしジャンプ」(JU)命令の複数エントリから成るリストをプログラムします。 このリストは「ジャンプリスト定義」命令の直後に開始し、最大 255 エントリを格納することができます。 リストでのジャンプファンクションのナンバリングはゼロで開始します。 ジャンプリストはギャップなしでプログラムする必要があります。 ジャンプリストの終了は、「ジャンプリスト定義」命令に指定されたジャンプラベルによってマーキングされます。

リストのどのジャンプファンクションが実行されるかは、アキュムレータ 1 の右バイトの値によって異なります。たとえば、アキュムレータ 1 の値が「0」の場合、最初のジャンプファンクションが実行されます。 アキュムレータ 1 の値が「1」の場合、2 番目のジャンプファンクションが実行されます。 アキュムレータ 1 の値がリストエントリの数よりも大きい場合、「ジャンプリスト定義」命令はリストの終了を参照します。

「ディストリビュータジャンプ」命令は条件に関係なく実行され、ステータスビットに影響しません。

構文

「ジャンプリスト定義」命令は、以下の一般的なアウトラインに従ってプログラムします。

```
L <jump number>
JL <jump label end>
JU <jump label>
JU <jump label>
.... <Jump label>
<Jump label_end>:
```

オペランド「ジャンプ数」には、実行される「条件なしジャンプ」(JU)命令の数が含まれています。この番号はアキュムレータ 1 にロードする必要があります。ジャンプリストの終了はジャンプラベル「jumplabel_end」によってマーキングされます。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// ジャンプ数をアキュムレータ 1 にロード。
JL END	// ジャンプリストの開始
JU MyLABEL_1	// 値が「0」の場合、アキュムレータ 1 で実行しジャンプラベル MyLABEL_1"にジャンプ。
JU MyLABEL_2	// 値が「1」の場合、アキュムレータ 1 で実行しジャンプラベル「MyLABEL_2」にジャンプ
JU MyLABEL_3	// 値が「2」の場合、アキュムレータ 1 で実行し、ジャンプラベル「MyLABEL_3」にジャンプ。
END: L "MyTag_1"	// ジャンプリストの終了

```
                        // オペランド「MyTag_1」の内容をアキュムレータ 1 にロード。
L "MyTag_2"             // オペランド「MyTag_2」の内容をアキュムレータ 1 にロード。
+I                      // 値の加算
T "Tag_Output_2"        // アキュムレータ 1 の内容をオペランド「Tag_Output_2」にロード。
MyLABEL_1: ...          // ジャンプラベル「MyLABEL_1」
....                    // 任意のプログラム
JU NEXT                 // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
MyLABEL_2: ...          // ジャンプラベル「MyLABEL_2」
....                    // 任意のプログラム
JU NEXT                 // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
MyLABEL_3: ...          // ジャンプラベル「MyLABEL_3」
....                    // 任意のプログラム
JU NEXT                 // ジャンプラベル「NEXT」にジャンプし、プログラムの実行をそこから再開。
NEXT: ...               // ジャンプラベル「NEXT」
```

LOOP: ループ

説明

「ループ」命令を使用して、プログラム内にループをプログラムします。

命令はアキュムレータ 1 の右ワードを 0～65535 範囲で符号なし 16 ビット整数として解釈します。実行中に命令は最初にアキュムレータ 1 の内容を 1 デクリメントします。 デクリメント後にアキュムレータ 1 の値がゼロでない場合、指定したジャンプラベルでジャンプが実行されます。 値がゼロの場合、ジャンプは実行されずプログラムの処理は次の命令で再開されます。 したがって、アキュムレータ 1 内の値は実行するプログラムループの数を指定します。 この数はループカウンタに保存する必要があります。

「ループ」命令はステータスビットには影響しません。

警告
ループカウンタのプリセット ループカウンタがプリセットされていない場合、または値「0」でプリセットされている場合、ループカウンタはさらに負の数にデクリメントされます。これによって、CPU を STOP モードにする可能性のある無限ループに陥ります。

構文

「ループ」命令は、以下の一般的なアウトラインに従ってプログラムされます。

```
L <number>
<Jump label>:
T <loop counter>
.... 任意のプログラム
L <loop counter>
LOOP <jump label>
```

<number>オペランドにはループサイクルの数が含まれます。 <loop counter>オペランド にはこれから実行されるループサイクルの数が含まれます。 プログラムループの終了時にループカウンタの内容はアキュムレータ 1 に読み込まれ、命令によって 1 デクリメントされます。 デクリメント後にアキュムレータの値がゼロでない場合、プログラムループ開始にあるジャンプラベルへのジャンプが実行されます。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// 実行される時間ループの数をアキュムレータ 1 の右ワードにロード。
START: T "Tag_Counter"	// ジャンプリストの開始

```
                        // アキュムレータ 1 の内容をループカウンタに転送する。
L "MyTag_1"             // オペランド「MyTag_1」の値をロード。
L "MyTag_2"             // オペランド「MyTag_2」の値をロード。
*D                      // 値を乗算
T "MyTag_1"             // 乗算結果をオペランド「MyTag_1」に転送。
L "Tag_Counter"         // ループカウンタの値をアキュムレータ 1 にロード。
                        // アキュムレータ 1 の値を 1 デクリメント。
LOOP START              // 値がゼロでない場合はプログラムループ開始にジャンプ。
                        // 値がゼロの場合は、次の命令を実行。
L "MyTag_2"             // オペランド「MyTag_2」の値をロード。
L 100                   // 値 100 をロード。
>I                      // オペランド「MyTag_2」の値が 200 よりも大きいかどうかを
                        比較。
= "MyTag_3"             // 比較結果をオペランド「MyTag_3」に書き込む。
```

データブロック

この章には下記に関する情報が記載されています：

- OPN: DB レジスタのデータブロックを開く (S7-1500)
- OPNDI: DI レジスタ内のデータブロックを開く (S7-1500)
- CDB: データブロックレジスタのスワップ (S7-1500)
- L DBLG: グローバルデータブロック長をアキュムレータ 1 にロード (S7-1500)
- L DBNO: グローバルデータブロックの　暗号をアキュムレータ 1 に読み込む (S7-1500)
- L DILG: インスタンスデータブロック長をアキュムレータ 1 にロード (S7-1500)
- L DINO: インスタンスデータブロック数をアキュムレータ 1 にロード (S7-1500)

OPN: DB レジスタのデータブロックを開く

説明

「DB レジスタのデータブロックを開く」命令でグローバルデータブロック(DB)を開けます。 データブロックの番号が、DB レジスタに転送されます。 以降の DB コマンドは、レジスタの内容に応じて関連するブロックにアクセスします。

「DB レジスタのデータブロックを開く」命令は条件なしで処理され、論理演算の結果またはアキュムレータの内容のいずれにも影響しません。

構文

「DB レジスタのデータブロックを開く」命令には、以下の構文を使用します。

```
OPN <data block>
```

パラメータ

次の表に、「DB レジスタのデータブロックを開く」命令のパラメータを示します。

オペランド	宣言	データブロックタイプ	説明
<data block>	-	DB	開くデータブロック。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPN "GlobalDataBlock"	// グローバルデータブロックを開き、ブロック数を DB レジスタに転送。
L %DBW0	// 開いたグローバルデータブロックのデータワード DBW0 をアキュムレータ 1 にロード。
T "MyTag"	// アキュムレータ 1 の内容をオペランド「MyTag_1」に転送。

OPNDI: DI レジスタ内のデータブロックを開く

説明

「DI レジスタのデータブロックを開く」命令であらゆるデータブロック(DB)を開けます。 データブロックの番号が DI レジスタに転送されます。 以降の DI コマンドは、レジスタの内容に応じて関連するブロックにアクセスします。

ブロックインターフェイスからのローカル仮パラメータのシンボリックアドレス指定では、必ずブロック呼び出し時に指定した番号のデータブロックにアクセスします。

「DI レジスタのデータブロックを開く」命令は条件なしで処理され、論理演算の結果またはアキュムレータの内容のいずれにも影響しません。

構文

「DI レジスタのデータブロックを開く」命令には、以下の構文を使用します。

```
OPNDI <data block>
```

パラメータ

次の表に、「DI レジスタのデータブロックを開く」命令のパラメータを示します。

オペランド	宣言	データブロックタイプ	説明
<data block>	-	DI	開くデータブロック。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPNDI "DataBlock"	// データブロックを開き、データブロックの番号を DI レジスタに転送。
L %DIW0	// 開いたデータブロックのデータワード DIW0 をアキュムレータ 1 にロード
T "MyTag"	// アキュムレータ 1 の内容をオペランド「MyTag」に転送。

CDB: データブロックレジスタのスワップ

説明

「データブロックレジスタのスワップ」命令を使用して、データブロックレジスタの内容をスワップします。 この命令は条件に関係なく実行され、ステータスビットに影響しません。

構文

「データブロックレジスタのスワップ」命令には、以下の構文を使用します。

```
CDB <data block>
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPN "GlobalDataBlock"	// グローバルデータブロックを開き、データブロックの番号を DB レジスタに転送。
OPNDI "DataBlock"	// データブロックを開き、データブロックの番号を DI レジスタに転送。
	// データブロックレジスタのスワップ
CDB	// DB レジスタは「DataBlock」をポイントし、DI レジスタは「GlobalDataBlock」をポイント。
L %DIW0	// 「GlobalDataBlock」からアキュムレータ 1 に DW0 をロード。

L DBLG: グローバルデータブロック長をアキュムレータ 1 にロード

説明

「グローバルデータブロック長をアキュムレータ 1 にロード」命令を使用し、データブロックレジスタによって開かれたグローバルデータブロックの長さをアキュムレータ 1 にロードします。アキュムレータ 1 の以前の内容はアキュムレータ 2 に移動します。

この命令の処理前にデータブロックレジスタによってグローバルデータブロックが開かなかった場合、値「0」がアキュムレータ 1 に読み込まれます。データブロックの長さはデータのバイト数に等しくなります。

ステータスビットはこの命令に影響されません。

構文

「グローバルデータブロック長をアキュムレータ 1 にロード」命令には、以下の構文を使用します。

```
L DBLG
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPN "GlobalDataBlock"	// グローバルデータブロックを開き、データブロックの番号を DB レジスタに転送。
L DBLG	// 開いたデータブロックの長さをアキュムレータ 1 にロード。
L "MyTag_1"	// 比較値をロード
<D	// データブロックの長さがオペランド「MyTag_1」の値未満であるかどうかを比較。
= "MyTag_2"	// 結果をオペランド「MyTag_2」に書き込む。

L DBNO: グローバルデータブロックの　暗号をアキュムレータ 1 に読み込む

説明

「グローバルデータブロック番号をアキュムレータ 1 にロード」命令を使用し、データブロックレジスタによって開いたグローバルデータブロックの番号をアキュムレータ 1 にロードします。アキュムレータ 1 の以前の内容はアキュムレータ 2 に移動します。

この命令の処理前にデータブロックレジスタによってグローバルデータブロックが開かなかった場合、値「0」がアキュムレータ 1 に読み込まれます。

ステータスビットはこの命令に影響されません。

構文

「グローバルデータブロックの　暗号をアキュムレータ 1 に読み込む」命令には、以下の構文を使用します。

```
L DBNO
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPN "GlobalDataBlock"	// グローバルデータブロックを開き、データブロックの番号を DB レジスタに転送。
L DBNO	// 開いたデータブロックの番号をアキュムレータ 1 にロード。
L "MyTag_1"	// 比較値をロード
==I	// データブロックの番号がオペランド「MyTag_1」に等しいかどうかを比較。
= "MyTag_2"	// 結果をオペランド「MyTag_2」に書き込む。

L DILG: インスタンスデータブロック長をアキュムレータ 1 にロード

説明

「インスタンスデータブロック長をアキュムレータ 1 にロード」命令を使用し、データブロックレジスタによって開かれたインスタンスデータブロックの長さをアキュムレータ 1 にロードします。アキュムレータ 1 の以前の内容はアキュムレータ 2 に移動します。

この命令の処理前にデータブロックレジスタによってインスタンスデータブロックが開かなかった場合、値「0」がアキュムレータ 1 に読み込まれます。データブロックの長さはデータのバイト数に等しくなります。

ステータスビットはこの命令に影響されません。

構文

「インスタンスデータブロック長をアキュムレータ 1 にロード」命令には、以下の構文を使用します。

```
L DILG
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPNDI "InstanceDataBlock"	// データブロックを開き、データブロックの番号を DI レジスタに転送。
L DILG	// 開いたデータブロックの長さをアキュムレータ 1 にロード。
L "MyTag_1"	// 比較値をロード
<I	// データブロックの長さがオペランド「MyTag_1」の値未満であるかどうかを比較。
= "MyTag_2"	// 結果をオペランド「MyTag_2」に書き込む。

L DINO: インスタンスデータブロック数をアキュムレータ 1 にロード

説明

「インスタンスデータブロックの番号をアキュムレータ 1 にロード」命令を使用し、データブロックレジスタによって開いたインスタンスデータブロックの番号をアキュムレータ 1 にロードします。アキュムレータ 1 の以前の内容はアキュムレータ 2 に移動します。

この命令の処理前にデータブロックレジスタによってインスタンスデータブロックが開かなかった場合、値「0」がアキュムレータ 1 に読み込まれます。

ステータスビットはこの命令に影響されません。

構文

「インスタンスデータブロック数をアキュムレータ 1 にロード」命令には、以下の構文を使用します。

```
L DINO
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
OPNDI "InstanceDataBlock"	// データブロックを開き、データブロックの番号を DI レジスタに転送。
L DINO	// 開いたデータブロックの番号をアキュムレータ 1 にロード。
L "MyTag_1"	// 比較値をロード
==I	// データブロックの番号がオペランド「MyTag_1」に等しいかどうかを比較。
= "MyTag_2"	// 結果をオペランド「MyTag_2」に割り当てる。

プログラムブロック

この章には下記に関する情報が記載されています：

- BE: ブロック終了 (S7-1500)
- BEC: 条件付きブロック終了 (S7-1500)
- BEU: 条件なしブロック終了 (S7-1500)
- CALL: ブロック呼び出し (S7-1500)
- CC: 条件付きブロック呼び出し (S7-1500)
- UC: 条件なしブロックの呼び出し (S7-1500)

BE: ブロック終了

説明

「ブロック終了」命令を使用して、現在処理中のブロックを終了し、ブロックを呼び出しているプログラムの箇所に移動することができます。 プログラムの処理は、ブロック呼び出し後に直接置かれた命令によって再開されます。

「ブロック終了」命令は、条件に関わりなく、実行されます。 命令の処理がジャンプ命令によってスキップした場合、現在のプログラムの実行は終了せず、ブロック内のジャンプ宛先で再開されます。

構文

「ブロック終了」命令には、以下の構文を使用します。

```
BE
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
SIN	// 正弦値の形成
	// 結果をアキュムレータ 1 に保存。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
	// オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。
	// 積をアキュムレータ 1 に保存。
L "Tag_Value_3"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
	// オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード
*R	// アキュムレータ 1 および 2 の値を乗算。
	// 積をアキュムレータ 1 に保存。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランド「Tag_Result」に転送。
BE	// ブロックの終了

BEC: 条件付きブロック終了

説明

「条件付きブロック終了」命令を使用して、現在処理中のブロックを論理演算の結果(RLO)に応じて終了し、ブロックを呼び出しているプログラムの箇所に移動することができます。

RLO が「1」の場合、命令は実行されます。 現在処理中のブロックは終了し、プログラム処理はブロック呼び出しで再開されます。 プログラムの処理は、ブロック呼び出し後に直接置かれた命令によって再開されます。

命令の処理時に RLO が「0」の場合、命令は実行されません。 この場合、CPU は RLO を「1」にセットし、次の命令を実行します。

命令の処理がジャンプ命令によってスキップした場合、現在のプログラムの実行は終了せず、ブロック内のジャンプ宛先で再開されます。

構文

「条件付きブロック終了」命令には、以下の構文を使用します。

```
BEC
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。
A "Tag_Input"	// オペランド「Tag_Input」の値が「1」かどうかを照会し、RLO を AND 論理積演算。
BEC	// 条件(RLO = 「1」)を満たす場合、ブロック終了。 // 条件(RLO = 「0」)を満たさない場合、次の命令を実行。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランド「Tag_Result」に転送。

BEU: 条件なしブロック終了

説明

「条件なしブロック終了」命令を使用して、現在処理中のブロックを終了し、ブロックを呼び出しているプログラムの箇所に移動することができます。 プログラムの処理は、ブロック呼び出し後に直接置かれた命令によって再開されます。

「条件なしブロック終了」命令は、条件に関わりなく実行され、ブロック内で複数回プログラムできます。 命令の処理がジャンプ命令によってスキップした場合、現在のプログラムの実行は終了せず、ブロック内のジャンプ宛先で再開されます。

構文

「条件なしブロック終了」命令には、以下の構文を使用します。

```
BEU
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。
A "Tag_Input"	// オペランド「Tag_Input」の値が「1」かどうかを照会し、RLO を AND 論理積演算。
JC NEXT	// 条件(RLO = 「1」)を満たす場合、ジャンプラベル「NEXT」からプログラム実行を続行。 // 条件(RLO = 「0」)を満たさない場合、次の命令を実行。
BEU	// ブロックの終了
NEXT: T "Tag_Result"	// ジャンプラベル「NEXT」 // アキュムレータ 1 の内容をオペランド「Tag_Result」に転送。

CALL: ブロック呼び出し

説明

「ブロック呼び出し」命令は、プログラム内で以下のブロックタイプの呼び出しに使用します。

- ファンクション
- ファンクションブロック

「ブロック呼び出し」命令は条件なしで実行されます。 この命令の処理後に、プログラムは呼び出されたブロックで再開します。

データ付きの呼び出されたブロックをフィードできます。 このデータはブロックパラメータによって転送されます。 呼び出されたブロックのパラメータは、呼び出されたブロックの呼び出し命令に従って一覧に表示されます。 現在必要なパラメータをこれらのパラメータに割り当てることができます。「ブロック呼び出し」命令の実行中、データは呼び出されたブロックに転送されます。 データを転送することによって、ステータスワード、アドレスレジスタ、およびデータブロックレジスタの内容が変更されます。

パラメータ付きでフィードされないファンクションブロックのパラメータは、現在値を保持します。 すべてのパラメータは、ファンクションが呼び出されるときフィードされる必要があります。 呼び出されたブロックにパラメータがない場合、パラメータリストは表示されません。

呼び出されたブロックにインスタンスデータブロックが必要な場合、呼び出し時にこのブロックにコンマで区切って割り当てます。 指定されたデータブロックは、呼び出し前に作成する必要があります。

呼び出されたブロックが処理されると、CPU は呼び出し元ブロックに再び切り替わり、呼び出し命令後にこのブロックの処理を再開します。

注記
「条件付きブロック呼び出し」命令は、ブロックプロパティ「レジスタ経由のパラメータ渡し」が無効な場合にのみ使用可能です。

構文

「ブロック呼び出し」命令には、以下の構文を使用します。

```
CALL <FC>
```

ファンクションブロックを呼び出すとき、「ブロック呼び出し」命令には、以下の構文を使用します。

```
CALL <FB>, <FB_DB>
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
CALL "MyFunction"	
Input_1 := "Tag_Input_1",	// 呼び出しファンクション「MyFunction」
Input_2 := "Tag_Input_2"	// 実パラメータを割り当て
Output_1 := "Tag_Output_1"	
Output_2 := "Tag_Output_1"	
CALL "MyFunctionBlock", "MyFB_DB"	
Value_1 := "Tag_Value_1"	// 呼び出しファンクションブロック「MyFunction-Block」
Value_2 := "Tag_Value_2"	// 実パラメータを割り当て
Output := "Tag_Output"	
CALL "LIMIT"	
MN := "Tag_LowLimit"	// 「制限値の設定」命令の呼び出し
IN := "Tag_InputValue"	// 実パラメータを割り当て
MX := "Tag_HighLimit"	
OUT := "Tag_Output"	
CALL "CTU", "CTU_DB"	
CU := "Tag_StartCTU"	
R := "Tag_ResetCounter"	// 「カウントアップ」カウンタを呼び出し。
PV := "Tag_PresetValue"	// 実パラメータを割り当て
Q := "Tag_CounterStatus"	
CV := "Tag_CounterValue"	

CC: 条件付きブロック呼び出し

説明

「条件付きブロック呼び出し」命令を使用して、論理演算の結果(RLO)に応じて呼び出しファンクション(FC)およびファンクションブロック(FB)を呼び出します。

この命令は、命令実行前の論理演算の現在の結果(RLO)が「1」の場合にのみ実行されます。 この命令の処理後に、プログラムは呼び出されたブロックで再開します。 呼び出されたブロックが処理されると、CPU は呼び出し元ブロックに再び切り替わり、呼び出し命令後にこのブロックの処理を再開します。

ブロック変更がある場合、ステータスビット OS が「0」にリセットされます。 スタータスビット CC 0、CC 1、および OV は、呼び出しブロックによっては影響を受けます。

注記
「条件付きブロック呼び出し」命令は、ブロックプロパティ「レジスタ経由のパラメータ渡し」が有効な場合にのみ使用可能です。 ただし、これによりパフォーマンスが低下します。

アキュムレータおよびアドレスレジスタの内容は、「条件付きブロック呼び出し」命令では変化しません。

現在の RLO が「0」の場合、命令もブロック呼び出しも実行されず、RLO は「1」にセットされます。

構文

ファンクションを呼び出すとき、「条件付きブロック呼び出し」命令には、以下の構文を使用します。

```
CC <FC>
```

ファンクションブロックを呼び出すとき、「条件付きブロック呼び出し」命令には、以下の構文を使用します。

```
CC <FB>
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランド「Tag_Input_1」の値が「1」かどうかを照会し、現在の RLO を論理積演算。
CC "MyFunction"	// 条件(RLO = 「1」)を満たす場合、ファンクション My-Function を呼び出し。 // 条件(RLO = 「0」)を満たさない場合、次の命令を実行。
A "Tag_Input_2"	// オペランド「Tag_Input_2」の値が「1」かどうかを照会し、現在の RLO を論理積演算。

```
CC "MyFunctionBlock"
```

// 条件(RLO =「1」)を満たす場合、ファンクションブロック MyFunctionBlock を呼び出し。

// 条件(RLO = 「0」)を満たさない場合、次の命令を実行。

UC: 条件なしブロックの呼び出し

説明

「無条件ブロック呼び出し」命令を使用して、パラメータもインスタンスデータブロックもない呼び出しファンクション(FC)およびファンクションブロック(FB)を呼び出すことができます。

この命令は条件に関係なく実行されます。 この命令の処理後に、プログラムは呼び出されたブロックで再開します。 呼び出されたブロックが処理されると、CPU は呼び出し元ブロックに再び切り替わり、呼び出し命令後にこのブロックの処理を再開します。

ブロック変更がある場合、ステータスビット OS が「0」にリセットされます。 スタータスビット CC 0、CC 1、および OV は、呼び出しブロックによっては影響を受けます。

注記
「無条件ブロック呼び出し」命令は、ブロックプロパティ「レジスタ経由のパラメータ渡し」が有効な場合にのみ使用可能です。 ただし、これによりパフォーマンスが低下します。

アキュムレータおよびアドレスレジスタの内容は、「無条件ブロック呼び出し」命令では変化しません。

構文

ファンクションを呼び出すとき、「条件なしブロックの呼び出し」命令には、以下の構文を使用します。

```
UC <FC>
```

ファンクションブロックを呼び出すとき、「条件なしブロックの呼び出し」命令には、以下の構文を使用します。

```
UC <FB>
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
A "Tag_Input_1"	// オペランド「Tag_Input_1」の値が「1」かどうかを照会し、現在の RLO を論理積演算。
UC "MyFunction"	// 呼び出しファンクション「MyFunction」
A "Tag_Input_2"	// オペランド「Tag_Input_2」の値が「1」かどうかを照会し、現在の RLO を論理積演算。
UC "MyFunctionBlock"	// 呼び出しファンクションブロック「MyFunction-Block」

ワード論理演算

この章には下記に関する情報が記載されています：

- AW: 1 ワードごとに論理積演算 (S7-1500)
- OW: 1 ワードごとに論理和演算 (S7-1500)
- XOW: 1 ワードごとに排他的論理和演算 (S7-1500)
- AD: 1 ダブルワードごとに論理積演算 (S7-1500)
- OD: 1 ダブルワードごとに論理和演算 (S7-1500)
- XOD: 1 ダブルワードごとに排他的論理和演算 (S7-1500)

AW: 1 ワードごとに論理積演算

説明

「1 ワードごとに論理積演算」命令を使用して、アキュムレータ 1 の右ワード値とアキュムレータ 2 の右ワード値または指定された定数を 1 ビットずつ論理積演算します。 結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードは、未変更のままです。

この命令は、アキュムレータ 1 の左ビット 0 をアキュムレータ 2 のビット 0 または定数にリンクし、結果をアキュムレータ 1 のビット 0 に保存します。ビット 1 ~ 15 は同じ方法でリンクされます。

論理演算の両方のビットのシグナル状態が「1」ならば、結果ビットのシグナル状態が「1」になります。 論理演算の 2 つのビットの 1 つのシグナル状態が「0」の場合、対応する結果ビットはリセットされます。

次の表に、AND ワード論理演算によって、どのように結果が形成されるかを示します。

アキュムレータ 2 / 定数	0	0	1	1
アキュムレータ 1	0	1	0	1
結果	0	0	0	1

この命令は条件に関係なく実行され、論理演算の結果に影響しません。

この命令はステータスビット CC 0、CC 1 および OV に影響します。

構文

演算に定数を使用する場合、「1 ワードごとに論理積演算」命令には、以下の構文を使用します。

```
AW <constant>
```

演算にアキュムレータ 2 の右のワードの値を使用する場合、「1 ワードごとに論理積演算」命令には、以下の構文を使用します。

```
AW
```

パラメータ

次の表に、「1 ワードごとに論理積演算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	WORD	アキュムレータ 1 の右ワードの値と論理積演算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 の右ワードにロード。
AW W#16#F6B5	// アキュムレータ 1 の右ワードの値を定数(W#16#F6B5)と論理積演算。
	// アキュムレータ 1 の右ワードに結果を保存。
T "Tag_Result_1"	// 結果をオペランド「Tag_Result_1」に転送
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 の右ワードにロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_3"	// オペランド「Tag_Value_3」の値をアキュムレータ 1 の右ワードにロード。
AW	// アキュムレータ 2 の右ワードの値(「Tag_Value_2」)をアキュムレータ 1 の右ワードの値(「Tag_Value_3」)と論理積演算。
	// アキュムレータ 1 の右ワードに結果を保存。
T "Tag_Result_2"	// 結果をオペランド「Tag_Result_2」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
Tag_Value_1	0101	1001	0011	1011
定数(W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	0101	0000	0011	0001
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	0100	1000	0000	0010

OW: 1 ワードごとに論理和演算

説明

「1 ワードごとに論理和演算」命令を使用して、アキュムレータ 1 の右ワード値とアキュムレータ 2 の右ワード値または指定された定数を 1 ビットずつ論理和演算します。 結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードは、未変更のままです。

この命令は、アキュムレータ 1 の左ビット 0 をアキュムレータ 2 のビット 0 または定数にリンクし、結果をアキュムレータ 1 のビット 0 に保存します。ビット 1 ~ 15 は同じ方法でリンクされます。

論理演算の 2 つのビットのうち少なくとも 1 つのシグナル状態が「1」ならば、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が「0」ならば、結果ビットのシグナル状態が「0」にリセットされます。

次の表に、OR ワード論理和演算によって、どのように結果が形成されるかを示します。

アキュムレータ 2 / 定数	0	0	1	1
アキュムレータ 1	0	1	0	1
結果	0	1	1	1

この命令は条件に関係なく実行され、論理演算の結果に影響しません。

この命令はステータスビット CC 0、CC 1 および OV に影響します。

構文

演算に定数を使用する場合、「1 ワードごとに論理和演算」命令には、以下の構文を使用します。

```
OW <constant>
```

演算にアキュムレータ 2 の右のワードの値を使用する場合、「1 ワードごとに論理和演算」命令には、以下の構文を使用します。

```
OW
```

パラメータ

次の表に、「1 ワードごとに論理和演算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	WORD	アキュムレータ 1 の右ワードの値と論理和演算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 の右ワードにロード。
OW W#16#F6B5	// アキュムレータ 1 の右ワードの値を定数(W#16#F6B5)と論理和演算。
	// アキュムレータ 1 の右ワードに結果を保存。
T "Tag_Result_1"	// 結果をオペランド「Tag_Result_1」に転送。
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 の右ワードにロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_3"	// オペランド「Tag_Value_3」の値をアキュムレータ 1 の右ワードにロード。
OW	// アキュムレータ 2 の右ワードの値(「Tag_Value_2」)をアキュムレータ 1 の右ワードの値(「Tag_Value_3」)と論理和演算。
	// アキュムレータ 1 の右ワードに結果を保存。
T "Tag_Result_2"	// 結果をオペランド「Tag_Result_2」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
Tag_Value_1	0101	1001	0011	1011
定数(W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1111	1111	1011	1111
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1111	1110	1011	1011

XOW: 1 ワードごとに排他的論理和演算

説明

「1 ワードごとに排他的論理和演算」命令を使用して、アキュムレータ 1 の右ワード値とアキュムレータ 2 の右ワード値または指定された定数を 1 ビットずつ排他的論理和演算します。 結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードは、未変更のままです。

この命令は、アキュムレータ 1 の左ビット 0 をアキュムレータ 2 のビット 0 または定数にリンクし、結果をアキュムレータ 1 のビット 0 に保存します。ビット 1 ~ 15 は同じ方法でリンクされます。

リンクされる 2 つのビットのシグナル状態が異なる場合、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が同じならば、結果ビットのシグナル状態が「0」にリセットされます。

次の表に、EXCLUSIVE OR ワード論理和演算によって、どのように結果が形成されるかを示します。

アキュムレータ 2 / 定数	0	0	1	1
アキュムレータ 1	0	1	0	1
結果	0	1	1	0

この命令は条件に関係なく実行され、論理演算の結果に影響しません。

この命令はステータスビット CC 0、CC 1 および OV に影響します。

構文

演算に定数を使用する場合、「1 ワードごとに排他的論理和演算」命令には、以下の構文を使用します。

```
XOW <constant>
```

演算にアキュムレータ 2 の右のワードの値を使用する場合、「1 ワードごとに排他的論理和演算」命令には、以下の構文を使用します。

```
XOW
```

パラメータ

次の表に、「1 ワードごとに排他的論理和演算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	WORD	アキュムレータ 1 の右ワードの値と排他的論理和演算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 の右ワードにロード。
XOW W#16#F6B5	// アキュムレータ 1 の右ワードの値を定数(W#16#F6B5)と排他的論理和演算。
T "Tag_Result_1"	// アキュムレータ 1 の右ワードに結果を保存。 // 結果をオペランド「Tag_Result_1」に転送。
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 の右ワードにロード。
L "Tag_Value_3"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_3」の値をアキュムレータ 1 の右ワードにロード。
XOW	// アキュムレータ 2 の右ワードの値(「Tag_Value_2」)をアキュムレータ 1 の右ワードの値(「Tag_Value_3」)と排他的論理和演算。
T "Tag_Result_2"	// アキュムレータ 1 の右ワードに結果を保存。 // 結果をオペランド「Tag_Result_2」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
Tag_Value_1	0101	1001	0011	1011
定数(W#16#F6B5)	1111	0110	1011	0101
Tag_Result_1	1010	1111	1000	1110
Tag_Value_2	0110	1100	0010	1010
Tag_Value_3	1101	1010	1001	0011
Tag_Result_2	1011	0110	1011	1001

AD: 1 ダブルワードごとに論理積演算

説明

「1 ダブルワードごとに論理積演算」命令を使用して、アキュムレータ 1 の内容と、アキュムレータ 2 の内容または指定した定数値を 1 ビットずつ AND 論理積演算することができます。 この結果は、アキュムレータ 1 に保存されます。

この命令は、アキュムレータ 1 の左ビット 0 をアキュムレータ 2 のビット 0 または定数にリンクし、結果をアキュムレータ 1 のビット 0 に保存します。ビット 1 ~ 31 は同じ方法でリンクされます。

論理演算の両方のビットのシグナル状態が「1」ならば、結果ビットのシグナル状態が「1」になります。 論理演算の 2 つのビットの 1 つのシグナル状態が「0」の場合、対応する結果ビットはリセットされます。

次の表に、AND ワード論理演算によって、どのように結果が形成されるかを示します。

アキュムレータ 2 / 定数	0	0	1	1
アキュムレータ 1	0	1	0	1
結果	0	0	0	1

この命令は条件に関係なく実行され、論理演算の結果に影響しません。

この命令はステータスビット CC 0、CC 1 および OV に影響します。

構文

演算に定数を使用する場合、「1 ダブルワードごとに論理積演算」命令には、以下の構文を使用します。

```
AD <constant>
```

演算にアキュムレータ 2 の値を使用する場合、「1 ダブルワードごとに論理積演算」命令には、以下の構文を使用します。

```
AD
```

パラメータ

次の表に、「1 ダブルワードごとに論理積演算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	DWORD	アキュムレータ 1 の値と AND 論理積演算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
AD DW#16#39C657AC	// アキュムレータ 1 の値を定数(DW#16#39C657AC)と AND 論理積演算。
	// 結果をアキュムレータ 1 に保存。
T "Tag_Result_1"	// 結果をオペランド「Tag_Result_1」に転送。
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 にロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_3"	// オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード。
AD	// アキュムレータ 2 の値 Tag_Value_2 をアキュムレータ 1 の値(「Tag_Value_3)で AND 論理積演算
	// 結果をアキュムレータ 1 に保存。
T "Tag_Result_2"	// 結果をオペランド「Tag_Result_2」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
定数 (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0001	1001	0100	0100	0001	0101	0010	1000
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0010	0001	0100	0010	0101	0101	0010	1010

OD: 1 ダブルワードごとに論理和演算

説明

「1 ダブルワードごとに論理和演算」命令を使用して、アキュムレータ 1 の内容と、アキュムレータ 2 の内容または指定した定数値を 1 ビットずつ OR 論理和演算することができます。 この結果は、アキュムレータ 1 に保存されます。

この命令は、アキュムレータ 1 の左ビット 0 をアキュムレータ 2 のビット 0 または定数にリンクし、結果をアキュムレータ 1 のビット 0 に保存します。ビット 1 ~ 31 は同じ方法でリンクされます。

論理演算の 2 つのビットのうち少なくとも 1 つのシグナル状態が「1」ならば、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が「0」ならば、結果ビットのシグナル状態が「0」にリセットされます。

次の表に、OR ワード論理和演算によって、どのように結果が形成されるかを示します。

アキュムレータ 2 / 定数	0	0	1	1
アキュムレータ 1	0	1	0	1
結果	0	1	1	1

この命令は条件に関係なく実行され、論理演算の結果に影響しません。

この命令はステータスビット CC 0、CC 1 および OV に影響します。

構文

演算に定数を使用する場合、「1 ダブルワードごとに論理和演算」命令には、以下の構文を使用します。

```
OD <constant>
```

演算にアキュムレータ 2 の値を使用する場合、「1 ダブルワードごとに論理和演算」命令には、以下の構文を使用します。

```
OD
```

パラメータ

次の表に、「1 ダブルワードごとに論理和演算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	DWORD	アキュムレータ 1 の値と OR 論理和演算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
OD DW#16#39C657AC	// アキュムレータ 1 の値を定数(DW#16#39C657AC)と OR 論理和演算。
	// 結果をアキュムレータ 1 に保存。
T "Tag_Result_1"	// 結果をオペランド「Tag_Result_1」に転送。
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 にロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_3"	// オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード。
OD	// アキュムレータ 2 の値 Tag_Value_2 をアキュムレータ 1 の値(「Tag_Value_3)で OR 論理和演算
	// 結果をアキュムレータ 1 に保存
T "Tag_Result_2"	// 結果をオペランド「Tag_Result_2」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
定数 (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0111	1111	1110	0110	1101	1111	1011	1111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0111	1101	1100	0111	0101	1111	1110	1111

XOD: 1 ダブルワードごとに排他的論理和演算

説明

「1 ダブルワードごとに排他的論理和演算」命令を使用して、アキュムレータ 1 の内容とアキュムレータ 2 の内容を 1 ビットずつ排他的論理和演算することができます。 この結果は、アキュムレータ 1 に保存されます。

この命令は、アキュムレータ 1 の左ビット 0 をアキュムレータ 2 のビット 0 または定数にリンクし、結果をアキュムレータ 1 のビット 0 に保存します。ビット 1 ~ 31 は同じ方法でリンクされます。

リンクされる 2 つのビットのシグナル状態が異なる場合、結果ビットのシグナル状態は「1」になります。 論理演算の両方のビットのシグナル状態が同じならば、結果ビットのシグナル状態が「0」にリセットされます。

次の表に、EXCLUSIVE OR ワード排他的論理和演算によって、どのように結果が形成されるかを示します。

アキュムレータ 2 / 定数	0	0	1	1
アキュムレータ 1	0	1	0	1
結果	0	1	1	0

この命令は条件に関係なく実行され、論理演算の結果に影響しません。

この命令はステータスビット CC 0、CC 1 および OV に影響します。

構文

演算に定数を使用する場合、「1 ダブルワードごとに排他的論理和演算」命令には、以下の構文を使用します。

```
XOD <constant>
```

演算にアキュムレータ 2 の値を使用する場合、「1 ダブルワードごとに排他的論理和演算」命令には、以下の構文を使用します。

```
XOD
```

パラメータ

次の表に、「1 ダブルワードごとに排他的論理和演算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	DWORD	アキュムレータ 1 の値と XOR 排他的論理和演算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
XOD DW#16#39C657AC	// アキュムレータ 1 の値を定数(DW#16#39C657AC)と XOR 排他的論理和演算。
	// 結果をアキュムレータ 1 に保存。
T "Tag_Result_1"	// 結果をオペランド「Tag_Result_1」に転送。
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 にロード。
	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。
L "Tag_Value_3"	// オペランド「Tag_Value_3」の値をアキュムレータ 1 にロード。
XOD	// アキュムレータ 2 の値 Tag_Value_2 をアキュムレータ 1 の値(「Tag_Value_3)で XOR 排他的論理和演算
	// 結果をアキュムレータ 1 に保存。
T "Tag_Result_2"	// 結果をオペランド「Tag_Result_2」に転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
定数 (DW#16#39C657AC)	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_1	0110	0110	1010	0010	1100	1010	1001	0111
Tag_Value_2	0110	0101	0100	0011	0101	1101	0010	1011
Tag_Value_3	0011	1001	1100	0110	0101	0111	1010	1100
Tag_Result_2	0101	1100	1000	0101	0000	1010	1000	0111

シフトとローテーション

この章には下記に関する情報が記載されています：

- シフト (S7-1500)
- ローテーション (S7-1500)

シフト

この章には下記に関する情報が記載されています：

SSI: 符号付きで 1 ワードずつシフト

説明

「符号付で 1 ワードずつシフト」命令を使用して、アキュムレータ 1 の右ワード(ビット 0～15)を 1 ビットずつ右へシフトします。 シフト中、空きになる位置はビット 15 のシグナル状態(INT 数の符号ビット)で埋められます。 アキュムレータ 1 のビット 16～31 は変更なしのままです。

ビットがシフトされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。 指定する値は 0～15 の値の範囲の数値である必要があります。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定したシフトする桁数が 15 よりも大きい場合、アキュムレータ 1 の右ワードにあるすべてのビットはビット 15 のシグナル状態で埋められます。

指定したシフトする桁数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

シフトする桁数が「符号付きで 1 ワードずつシフト」命令のパラメータとして指定される場合、以下の構文を使用します。

```
SSI  <シフトする桁数>
```

シフトする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「符号付きで 1 ワードずつシフト」命令に以下の構文が使用されます。

```
SSI
```

パラメータ

次の表に、「符号付きで 1 ワードずつシフト」命令のパラメータを示します。

パラメータ	書式	説明
<シフトする桁数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	値がシフトされるビット位置の数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
SSI 6	// アキュムレータ 1 のビット 0～15 を右に 6 桁シフト。 // 空になったビット位置をビット 15 の状態で埋める。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 3	// シフトする桁数をアキュムレータ 1 にロード
L "Tag_Value_2"	// シフトする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
SSI	// アキュムレータ 1 のビット 0～15 を右に 3 桁シフト。 // 空になったビット位置をビット 15 の状態で埋める。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	1001	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1111	1110	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	1011	1010	0101

SSD: 符号付きで 1 ダブルワードずつシフト

説明

「符号付きで 1 ダブルワードずつシフト」命令を使用して、アキュムレータ 1 の完全な内容を 1 ビットずつ右へシフトします。 シフト中、空きになる位置はビット 31 のシグナル状態(DINT 数の符号ビット)で埋められます。

ビットがシフトされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。 指定する値は 0～31 の値の範囲の数値である必要があります。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定されたシフトする桁数が 31 よりも大きい場合、アキュムレータ 1 のすべてのビットはビット 31 のシグナル状態で埋められます。

指定したシフトする桁数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

シフトする桁数が「符号付きで 1 ダブルワードずつシフト」命令のパラメータとして指定される場合、以下の構文を使用します。

```
SSD  <シフトする桁数>
```

シフトする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「符号付きで 1 ダブルワードずつシフト」命令に以下の構文が使用されます。

```
SSD
```

パラメータ

次の表に、「符号付きで 1 ダブルワードずつシフト」命令のパラメータを示します。

パラメータ	書式	説明
<シフトする桁数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	値がシフトされるビット位置の数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
SSD 7	// アキュムレータ 1 のビット 0～31 を右に 7 桁シフト。 // 空になったビット位置をビット 31 の状態で埋める。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 4	// シフトする桁数をアキュムレータ 1 にロード
L "Tag_Value_2"	// シフトする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
SSD	// アキュムレータ 1 のビット 0～31 を右に 4 桁シフト。 // 空になったビット位置をビット 31 の状態で埋める。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	1000	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	1111	0001	1110	1100	1000	1011	1010
Tag_Value_2	0010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	0010	1000	1010	0010	1001	1011	1100

SLW: 1 ワードずつ左へシフト

説明

「1 ワードずつ左へシフト」命令を使用して、アキュムレータ 1 の右ワード(ビット 0～15)を 1 ビットずつ左へシフトします。 空きになるビット位置はゼロで埋められます。 アキュムレータ 1 のビット 16～31 は変更なしのままです。

ビットがシフトされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。 指定する値は 0～15 の値の範囲の数値である必要があります。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定されたシフトする桁数が 15 よりも大きい場合、アキュムレータ 1 の右ワードのすべてのビットはゼロで埋められます。

指定したシフトする桁数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

シフトする桁数がパラメータとして指定される場合、「1 ワードずつ左へシフト」命令には以下の構文を使用します。

```
SLW <シフトする桁数>
```

シフトする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「1 ワードずつ左へシフト」命令に以下の構文が使用されます。

```
SLW
```

パラメータ

次の表に、「1 ワードずつ左へシフト」命令のパラメータを示します。

パラメータ	書式	説明
<シフトする桁数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	値がシフトされるビット位置の数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
SLW 5	// アキュムレータ 1 のビット 0～15 を左に 5 桁シフト // 空になったビット位置をゼロで埋める。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 4	// シフトする桁数をアキュムレータ 1 にロード
L "Tag_Value_2"	// シフトする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
SLW	// アキュムレータ 1 のビット 0～15 を左に 4 桁シフト。 // 空になったビット位置をゼロで埋める。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	1010	0111	0110	0000
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	1101	0010	1011	0000

SRW: 1 ワードずつ右へシフト

説明

「1 ワードずつ右へシフト」命令を使用して、アキュムレータ 1 の右ワード(ビット 0～15)を 1 ビットずつ右へシフトします。 空きになるビット位置はゼロで埋められます。アキュムレータ 1 のビット 16～31 は変更なしのままです。

ビットがシフトされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。 指定する値は 0～15 の値の範囲の数値である必要があります。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定されたシフトする桁数が 15 よりも大きい場合、アキュムレータ 1 の右ワードのすべてのビットはゼロで埋められます。

指定したシフトする桁数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

シフトする桁数がパラメータとして指定される場合、「1 ワードずつ右へシフト」命令には以下の構文を使用します。

```
SRW  <シフトする桁数>
```

シフトする桁数をアキュムレータ 2 の右ワードを使用して指定する場合、「1 ワードずつ右へシフト」命令に以下の構文が使用されます。

```
SRW
```

パラメータ

次の表に、「1 ワードずつ右へシフト」命令のパラメータを示します。

パラメータ	書式	説明
<シフトする桁数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	値がシフトされるビット位置の数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
SRW 6	// アキュムレータ 1 のビット 0～15 を右に 6 桁シフト。 // 空になったビット位置をゼロで埋める。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 4	// シフトする桁数をアキュムレータ 1 にロード
L "Tag_Value_2"	// シフトする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
SRW	// アキュムレータ 1 のビット 0～15 を右に 4 桁シフト。 // 空になったビット位置をゼロで埋める。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0101	1111	0110	0100	0000	0001	0111	0100
Tag_Value_2	0101	1111	0110	0100	0101	1101	0010	1011
Tag_Result_2	0101	1111	0110	0100	0000	0101	1101	0010

SLD: 1 ダブルワードずつ左へシフト

説明

「1 ダブルワードずつ左へシフト」命令を使用して、アキュムレータ 1 の完全な内容を 1 ビットずつ右へシフトします。空きになるビット位置はゼロで埋められます。

ビットがシフトされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。 指定する値は 0～31 の値の範囲の数値である必要があります。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定されたシフト数が 31 よりも大きい場合、アキュムレータ 1 のすべてのビットはゼロで埋められます。

指定したシフトする桁数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

シフトする桁数がパラメータとして指定される場合、「1 ダブルワードずつ左へシフト」命令には以下の構文を使用します。

```
SLD  <シフトする桁数>
```

シフトする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「1 ダブルワードずつ左へシフト」命令に以下の構文が使用されます。

```
SLD
```

パラメータ

次の表に、「1 ダブルワードずつ左へシフト」命令のパラメータを示します。

パラメータ	書式	説明
<シフトする桁数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	値がシフトされるビット位置の数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
SLD 5	// アキュムレータ 1 のビット 0 ~ 31 を左に 5 桁シフト // 空になったビット位置をゼロで埋める。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 4	// シフトする桁数をアキュムレータ 1 にロード
L "Tag_Value_2"	// シフトする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
SLD	// アキュムレータ 1 のビット 0 ~ 31 を左に 4 桁シフト。 // 空になったビット位置をゼロで埋める。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1110	1100	1000	1011	1010	0111	0110	0000
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	1000	1010	0010	1001	1011	1100	1101	0000

SRD: 1 ダブルワードずつ右へシフト

説明

「1 ダブルワードずつ右へシフト」命令を使用して、アキュムレータ 1 の完全な内容を 1 ビットずつ右へシフトします。空きになるビット位置はゼロで埋められます。

ビットがシフトされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。 指定する値は 0～31 の値の範囲の数値である必要があります。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定されたシフト数が 31 よりも大きい場合、アキュムレータ 1 のすべてのビットはゼロで埋められます。

指定したシフトする桁数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

シフトする桁数がパラメータとして指定される場合、「1 ダブルワードずつ右へシフト」命令には以下の構文を使用します。

```
SRD  <シフトする桁数>
```

シフトする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「1 ダブルワードずつ右へシフト」命令に以下の構文が使用されます。

```
SRD
```

パラメータ

次の表に、「1 ダブルワードずつ右へシフト」命令のパラメータを示します。

パラメータ	書式	説明
<シフトする桁数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	値がシフトされるビット位置の数

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
SRD 7	// アキュムレータ 1 のビット 0～31 を右に 7 桁シフト。 // 空になったビット位置をゼロで埋める。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 4	// シフトする桁数をアキュムレータ 1 にロード
L "Tag_Value_2"	// シフトする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
SRD	// アキュムレータ 1 のビット 0～31 を右に 4 桁シフト。 // 空になったビット位置をゼロで埋める。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	0000	0000	1011	1110	1100	1000	1011	1010
Tag_Value_2	1010	1000	1010	0010	1001	1011	1100	1101
Tag_Result_2	0000	1010	1000	1010	0010	1001	1011	1100

ローテーション

この章には下記に関する情報が記載されています：

- RLD: 1 ダブルワードずつ左へローテーション (S7-1500)
- RRD: 1 ダブルワードずつ右へローテーション (S7-1500)
- RLDA: ステータスビット CC 1 のみ左へローテーション (S7-1500)
- RRDA: ステータスビット CC 1 のみ右へローテーション (S7-1500)

RLD: 1 ダブルワードずつ左へローテーション

説明

「1 ダブルワードずつ左へローテーション」命令を使用して、アキュムレータ 1 の完全な内容を 1 ビットずつ左へローテーションします。 処理中に命令はアキュムレータ 1 のビット 0～31 を 1 ビットずつ左へシフトし、書き込み中に空きになった位置をプッシュアウトされたビット位置によって埋めます。

ビットがローテーションされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定したローテーション数が 32 の場合、アキュムレータ 1 の内容は変化なしのままになります。 32 よりも大きい値の場合、ローテーション数は 32 で除算するモジュロによって計算されます。たとえばローテーション数が 34 の場合、アキュムレータ 1 のビットは 2 ビット位置ずつローテーションされます。

指定されたローテーション数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

ローテーションする桁数がパラメータとして指定される場合、「1 ダブルワードずつ左へローテーション」命令には以下の構文を使用します。

```
RLD <ローテーション数>
```

ローテーションする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「1 ダブルワードずつ左へローテーション」命令に以下の構文が使用されます。

```
RLD
```

パラメータ

次の表に、「1 ダブルワードずつ左へローテーション」命令のパラメータを示します。

パラメータ	書式	説明
<ローテーション数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	ローテーションされるビット位置の番号。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
RLD 4	// アキュムレータ 1 のビット 0～31 を左に 4 桁ローテーション。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 6	// ローテーションする桁数をアキュムレータ 1 にロード。
L "Tag_Value_2"	// ローテーションする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
RLD	// アキュムレータ 1 のビット 0～31 を左に 6 桁ローテーション。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	**0101**	1111	0110	0100	0101	1101	0011	1011
Tag_Result_1	1111	0110	0100	0101	1101	0011	1011	**0101**
Tag_Value_2	**1010**	**10**00	1010	0010	1001	1011	1100	1101
Tag_Result_2	0010	1000	1010	0110	1111	0011	01**10**	**1010**

RRD: 1 ダブルワードずつ右へローテーション

説明

「1 ダブルワードずつ右へローテーション」命令を使用して、アキュムレータ 1 の完全な内容を 1 ビットずつ右へローテーションします。 処理中に命令はアキュムレータ 1 のビット 0～31 を 1 ビットずつ右へシフトし、書き込み中に空きになった位置をプッシュアウトされたビット位置によって埋めます。

ビットがローテーションされるビット位置の数を指定するために以下のオプションを使用できます。

- 命令のパラメータとして正の整数を指定する。
- アキュムレータ 2 の右バイト値を指定する。バイトは正の整数として解釈されます。

この命令は RLO に関係なく実行されます。 ステータスビット CC0 および OV は「0」にリセットされ、ステータスビット CC1 は、シフトアウトされた最後のビットのシグナル状態にセットされます。

指定したローテーション数が 32 の場合、アキュムレータ 1 の内容は変化なしのままになります。 32 よりも大きい値の場合、ローテーション数は 32 で除算するモジュロによって計算されます。たとえばローテーション数が 34 の場合、アキュムレータ 1 のビットは 2 ビット位置ずつローテーションされます。

指定されたローテーション数がゼロの場合でも、命令は実行されます。 ステータスビット CC1 は「0」にセットされます。

構文

ローテーションする桁数がパラメータとして指定される場合、「1 ダブルワードずつ右へローテーション」命令には以下の構文を使用します。

```
RRD  <ローテーション数>
```

ローテーションする桁数をアキュムレータ 2 の右バイトを使用して指定する場合、「1 ダブルワードずつ右へローテーション」命令に以下の構文が使用されます。

```
RRD
```

パラメータ

次の表に、「1 ダブルワードずつ右へローテーション」命令のパラメータを示します。

パラメータ	書式	説明
<ローテーション数>	正の整数: SINT, INT, DINT, USINT, UINT, UDINT	ローテーションされるビット位置の番号。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
RRD 4	// アキュムレータ 1 のビット 0～31 を右に 4 桁ローテーション。
T "Tag_Result_1"	// アキュムレータ 1 の内容をオペランドに転送。
L 6	// ローテーションする桁数をアキュムレータ 1 にロード。
L "Tag_Value_2"	// ローテーションする桁数をアキュムレータ 2 の右バイトに移動。 // オペランドの値をアキュムレータ 1 にロード。
RRD	// アキュムレータ 1 のビット 0～31 を右に 6 桁ローテーション。
T "Tag_Result_2"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値							
Tag_Value_1	0101	1111	0110	0100	0101	1101	0011	**1011**
Tag_Result_1	**1011**	0101	1111	0110	0100	0101	1101	0011
Tag_Value_2	1010	1000	1010	0010	1001	1011	11**00**	**1101**
Tag_Result_2	**0011**	**01**10	1010	0010	1000	1010	0110	1111

RLDA: ステータスビット CC 1 のみ左へローテーション

説明

「ステータスビット CC 1 のみ左へローテーション」命令を使用して、アキュムレータ 1 の内容を 1 ビット左へシフトすることができます。 シフト中に空きとなるビット位置(0)は、ステータスビット CC 1 のシグナル状態で埋められます。 ステータスビット CC 1 は、シフトアウトされたビット(31)のシグナル状態を受け取ります。

この命令は RLO に関係なく実行されます。 この命令により、ステータスビット CC 0 が「0」にリセットされます。

構文

「ステータスビット CC 1 のみ左へローテーション」命令には、以下の構文が使用されます。

```
RLDA
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// オペランドの値をアキュムレータ 1 にロード。
	// アキュムレータ 1 のビット 0～31 を左に 1 桁ローテーション。
RLDA	// ビット 0 をステータスビット CC 1 のシグナル状態で埋める。
	// シフトアウトされたビットをステータスビット CC 1 に書き込み。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	CC 1	値							
Tag_Value	**1**	**0**101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	**0**	1011	1110	1100	1000	1011	1010	0111	011**1**

RRDA: ステータスビット CC 1 のみ右へローテーション

説明

「ステータスビット CC 1 のみ右へローテーション」命令を使用して、アキュムレータ 1 の内容を 1 ビット右へシフトします。 シフト中に空きとなるビット位置(31)は、ステータスビット CC 1 のシグナル状態で埋められます。 ステータスビット CC 1 は、シフトアウトされたビット(0)のシグナル状態を受け取ります。

この命令は RLO に関係なく実行されます。 この命令により、ステータスビット CC0 および OV が「0」にリセットされます。

構文

「ステータスビット CC 1 のみ右へローテーション」命令には、以下の構文を使用します。

```
RRDA
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value"	// オペランドの値をアキュムレータ 1 にロード。
	// アキュムレータ 1 のビット 0～31 を右に 1 桁ローテーション。
RRDA	// ビット 31 をステータスビット CC 1 のシグナル状態で埋める。
	// シフトアウトされたビットをステータスビット CC 1 に書き込み。
T "Tag_Result"	// アキュムレータ 1 の内容をオペランドに転送。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	CC 1	値							
Tag_Value	1	0101	1111	0110	0100	0101	1101	0011	1011
Tag_Result	0	1010	1111	1011	0010	0010	1110	1001	1101

オペランド「Tag_Value」のビット 0 の値が「1」となるため、ステータスビット CC 1 の値も「1」となります。

追加の命令

この章には下記に関する情報が記載されています：

- アキュムレータ (S7-1500)
- アドレスレジスタ (S7-1500)
- NULL 命令 (S7-1500)

アキュムレータ

この章には下記に関する情報が記載されています：

- TAK: アキュムレータ 1 と 2 の内容をスワップ (S7-1500)
- PUSH: 内容を次に上位のアキュムレータへシフト (S7-1500)
- POP: 内容を次に下位のアキュムレータへシフト (S7-1500)

TAK: アキュムレータ 1 と 2 の内容をスワップ

説明

「アキュムレータ 1 と 2 の内容をスワップ」命令を使用して、アキュムレータ 1 の内容とアキュムレータ 2 の内容を入れ替えます。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「アキュムレータ 1 と 2 の内容をスワップ」命令には、以下の構文を使用します。

```
TAK
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。
JC NEXT	// 条件(RLO = 「1」)を満たす場合、ジャンプラベル NEXT にジャンプ。
TAK	// 条件(RLO = 「0」)を満たさない場合、次の命令を実行。 // アキュムレータ 1 および 2 の内容をスワップ
NEXT: -I	// ジャンプラベル「NEXT」 // アキュムレータ 1 の値をアキュムレータ 2 の値から減算。
T "Tag_Output"	// アキュムレータ 1 の内容をオペランド「Tag_Output」に転送。

PUSH: 内容を次に上位のアキュムレータへシフト

説明

「内容を次に上位のアキュムレータへシフト」命令を使用して、アキュムレータ 1 の内容をアキュムレータ 2 に移動します。

アキュムレータ 1 の内容はこの命令に影響されず、命令の処理後も変化なしのままです。 アキュムレータ 2 の内容は消滅します。

次の表に、命令の実行前および実行後のアキュムレータ 1～2 の内容を示します。

ステータス	アキュムレータ	
	1	2
処理前	A	B
処理後	A	A

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「内容を次に上位のアキュムレータへシフト」命令には、以下の構文を使用します。

```
PUSH
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランドの値をアキュムレータ 1 にロード。
PUSH	// アキュムレータ 1～2 の内容を次に上位のアキュムレータへシフト。
L "Tag_Value_2"	// オペランドの値をアキュムレータ 1 にロード。

POP: 内容を次に下位のアキュムレータへシフト

説明

「内容を次に下位のアキュムレータへシフト」命令を使用して、アキュムレータ 2 の内容をアキュムレータ 1 に移動します。

アキュムレータ 2 の内容はこの命令に影響されず、命令の処理後も変化なしのままです。 アキュムレータ 1 の内容は消滅します。

次の表に、命令の実行前および実行後のアキュムレータ 1 ~ 2 の内容を示します。

ステータス	アキュムレータ	
	1	2
処理前	A	B
処理後	B	B

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。命令は、論理演算の結果にもステータスビットにも影響しません。

構文

「内容を次に下位のアキュムレータへシフト」命令には、以下の構文を使用します。

```
POP
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
T "Tag_Value_1"	// アキュムレータ 1 の内容をオペランド「Tag_Value_1」に転送。
POP	// アキュムレータ 2 の内容を下位のアキュムレータ 1 へシフト。
T "Tag_Value_2"	// アキュムレータ 1 の内容をオペランド「Tag_Value_2」に転送。

アドレスレジスタ

この章には下記に関する情報が記載されています：

- +AR1: アキュムレータ 1 を AR1 に加算 (S7-1500)
- +AR2: アキュムレータ 1 を AR2 に加算 (S7-1500)

+AR1: アキュムレータ 1 を AR1 に加算

説明

「アキュムレータ 1 を AR1 に加算」命令を使用して、値をアドレスレジスタ 1 の内容に加算します。アドレスレジスタ 1 にあるポインタのタイプおよびオペランドの範囲は保持されます。

加算する値の指定には、以下のオプションが使用できます。

- 定数による指定 命令はアドレスレジスタ 1 に定数の値を加算します。定数の値は範囲内部ポインタ(POINTER)のフォーマットに対応する必要があります。
- アキュムレータ 1 の右ワードの値を使用した指定 命令はアキュムレータ 1 の右ワードの値を 16 ビット整数として解釈し、正しい符号付きで 24 ビットに拡張します。 次にアキュムレータ 1 の値はアドレスレジスタ 1 にこの命令を追加します。-32 768 ~ +32 767 の値の範囲が許可されています。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

構文

定数を使用して追加値を指定する場合は、「アキュムレータ 1 を AR1 に加算」命令には、以下の構文を使用します。

```
+AR1 <constant>
```

アキュムレータ 1 の右のワードを使用して追加値を指定する場合は、「アキュムレータ 1 を AR1 に加算」命令には、以下の構文を使用します。

```
+AR1
```

パラメータ

次の表に、「アキュムレータ 1 を AR1 に加算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	POINTER	アドレスレジスタ 1 に加算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
+AR1 P#10.0	// アドレスレジスタ 1 へポイントを追加。
L "Tag_Value"	// オペランド「Tag_Value」の値をアキュムレータ 1 の右ワードにロード。

```
+AR1                    // アキュムレータ 1 の値をアドレスレジスタ 1 に加算。
TAR1 %MD24              // アドレスレジスタ 1 の内容をダブルワード MD24 に転送。
```

+AR2: アキュムレータ 1 を AR2 に加算

説明

「アキュムレータ 1 を AR2 に加算」命令を使用して、値をアドレスレジスタ 2 の内容に加算します。アドレスレジスタ 2 にあるポインタのタイプおよびオペランドの範囲は保持されます。

加算する値の指定には、以下のオプションが使用できます。

- 定数による指定 命令はアドレスレジスタ 2 に定数の値を加算します。定数の値は範囲内部ポインタ(POINTER)のフォーマットに対応する必要があります。
- アキュムレータ 1 の右ワードの値を使用した指定 命令はアキュムレータ 1 の右ワードの値を 16 ビット整数として解釈し、正しい符号付きで 24 ビットに拡張します。 次にアキュムレータ 1 の値はアドレスレジスタ 2 にこの命令を追加します。-32 768 ~ +32 767 の値の範囲が許可されています。

CPU は、論理演算の結果とステータスビットとは無関係に命令を実行します。 命令は、論理演算の結果にもステータスビットにも影響しません。

アキュムレータの内容はこの命令によって変更されません。

注記
アドレスレジスタ AR2 は複数インスタンスの処理時に使用します。「アキュムレータ 1 を AR2 に加算」命令をプログラムする場合、まずアドレスレジスタ 2 の内容を「保存」してから後で再ロードする必要があります。

構文

定数を使用して追加値を指定する場合は、「アキュムレータ 1 を AR2 に加算」命令には、以下の構文を使用します。

```
+AR2 <constant>
```

アキュムレータ 1 の右のワードを使用して追加値を指定する場合は、「アキュムレータ 1 を AR2 に加算」命令には、以下の構文を使用します。

```
+AR2
```

パラメータ

次の表に、「アキュムレータ 1 を AR2 に加算」命令のパラメータを示します。

パラメータ	データタイプ	説明
<constant>	POINTER	アドレスレジスタ 1 に加算される値。

例

次の例で、命令がどのように動作するかを示します。

STL	説明
+AR2 P#10.0	// アドレスレジスタ 2 へポイントを追加。
L "Tag_Value"	// オペランド「Tag_Value」の値をアキュムレータ 1 の右ワードにロード。
+AR2	// アキュムレータ 2 の値をアドレスレジスタ 1 に加算。
TAR2 %MD24	// アドレスレジスタ 1 の内容をダブルワード MD24 に転送。

NULL 命令

この章には下記に関する情報が記載されています：

- BLD: プログラム表示(NULL 命令) (S7-1500)
- NOP 0: NULL 命令 (S7-1500)
- NOP 1: NULL 命令 (S7-1500)

BLD: プログラム表示(NULL 命令)

説明

「プログラム表示(NULL 命令)」命令はファンクションを実行せず、ステータスビットに影響しません。 この命令は、パラメータ転送中、または LAD/FBD ネットワークの場合にコードシーケンスを認識するために使用します。 LAD または FBD プログラムが STL に表示されると自動的に作成されます。 パラメータ値は命令の数であり、プログラミングツールによって生成されます。

構文

以下の構文が「プログラム表示(NULL 命令)」命令に使用されます。

```
BLD <ID 番号>
```

パラメータ

次の表に、「プログラム表示(NULL 命令)」命令のパラメータを示します。

パラメータ	データタイプ	説明
<number>	WORD	命令の数

NOP 0: NULL 命令

説明

「NULL 命令」のパラメータが 0 の場合は、ファンクションを実行せず、ステータスビットに影響しません。 この命令コードには、ゼロが 16 個指定されたビットパターンが指定されています。 この命令は、プログラムを表示する場合のプログラミング装置にのみ関係します。

構文

パラメータ 0 の「NULL 命令」には、次の構文を使用します。

```
NOP  0
```

例

次の例で、命令がどのように動作するかを示します。

STL	説明
L "Tag_Value_1"	// オペランド「Tag_Value_1」の値をアキュムレータ 1 にロード。
L "Tag_Value_2"	// アキュムレータ 1 の内容をアキュムレータ 2 へシフト。 // オペランド「Tag_Value_2」の値をアキュムレータ 1 にロード
>I	// アキュムレータ 2 の値がアキュムレータ 1 の値よりも大きいかどうかを比較。
A	// オペランドのシグナル状態「1」を照会し、現在の RLO で論理積演算
JC NEXT	// 条件(RLO =「1」)を満たす場合、ジャンプラベル NEXT にジャンプ。 // 条件(RLO =「0」)を満たさない場合、次の命令を実行。
TAK	// アキュムレータ 1 および 2 の内容をスワップ
NEXT: NOP 0	// ジャンプラベル「NEXT」

NOP 1: NULL 命令

説明

「NULL 命令」のパラメータが 1 の場合は、ファンクションを実行せず、ステータスビットに影響しません。 この命令コードには、1 が 16 個指定されたビットパターンが指定されています。 この命令は、プログラムを表示する場合のプログラミング装置にのみ関係します。

構文

パラメータ 1 付きの「NULL 命令」では、次の構文を使用できます。

```
NOP 1
```

SCL

この章には下記に関する情報が記載されています：

- ビット論理演算 (S7-1200, S7-1500)
- タイマの動作 (S7-1200, S7-1500)
- カウンタ演算 (S7-1200, S7-1500)
- 比較演算 (S7-1200, S7-1500)
- 四則演算 (S7-1200, S7-1500)
- ムーブ操作 (S7-1200, S7-1500)
- 変換操作 (S7-1200, S7-1500)
- プログラム制御演算 (S7-1200, S7-1500)
- ワード論理演算 (S7-1200, S7-1500)
- シフトとローテーション (S7-1200, S7-1500)
- レガシー (S7-1500)

ビット論理演算

この章には下記に関する情報が記載されています：

- R_TRIG:信号立ち上がりエッジの検出 (S7-1200, S7-1500)
- F_TRIG:信号立ち下がりエッジの検出 (S7-1200, S7-1500)

R_TRIG:信号立ち上がりエッジの検出

説明

「信号立ち上がりエッジの検出」命令を使用して、CLK 入力で「0」から「1」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「0」から「1」への状態変化を検出すると、Q 出力で信号立ち上がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

構文

「信号立ち上がりエッジの検出」命令には、以下の構文を使用します。

```
SCL
<Instance>(CLK := <Operand>,
           Q => <Operand>)
```

パラメータ

次の表に、「信号立ち上がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"R_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

CLK 入力のタグでの前の状態は、「R_TRIG_DB」タグに保存されます。「0」から「1」へのシグナル状態の変化が「TagIn_1」および「TagIn_2」オペランド または「TagIn_3」オペランドで検出されると、「TagOut_Q」出力のシグナル状態は 1 サイクルについて「1」になります。

F_TRIG:信号立ち下がりエッジの検出

説明

「信号立ち下がりエッジの検出」命令を使用して、CLK 入力で「1」から「0」への状態変化を検出できます。 この命令は、指定されたインスタンスに保存した前の照会(エッジメモリビット)の状態と CLK 入力の現在値を比較します。 この命令が CLK 入力で「1」から「0」への状態変化を検出すると、Q 出力で信号立ち下がりエッジが生成されます。つまり、正確に 1 サイクルに対する出力値は TRUE すなわち「1」となります。

それ以外のすべての場合、この命令の出力のシグナル状態は「0」です。

プログラム内にこの命令を挿入する際には、[呼び出しオプション]ダイアログが自動的に開きます。このダイアログで、エッジメモリビットを独自のデータブロック(シングルインスタンス)に格納するか、ブロックインターフェース内にローカルタグ(マルチインスタンス)として格納するかを指定できます。

構文

「信号立ち下がりエッジの検出」命令には、以下の構文を使用します。

```
SCL
<Instance>(CLK := <Operand>,
           Q => <Operand>)
```

パラメータ

次の表に、「信号立ち下がりエッジの検出」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CLK	Input	BOOL	I、Q、M、D、L	エッジが照会される受信信号
Q	Output	BOOL	I、Q、M、D、L	エッジ評価の結果

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"F_TRIG_DB"(CLK := "TagIn",
            Q => "TagOut");
```

CLK 入力のタグでの前の状態は、「F_TRIG_DB」タグに保存されます。 「TagIn」オペランドで「1」から「0」へのシグナル状態の変化が検出されると、TagOut 出力のシグナル状態は「1」になります。

タイマの動作

この章には下記に関する情報が記載されています：

- TP: パルスの生成 (S7-1200, S7-1500)
- TON: オンディレータイマ (S7-1200, S7-1500)
- TOF: オフディレータイマ (S7-1200, S7-1500)
- TONR: タイムアキュムレータ (S7-1200, S7-1500)
- RESET_TIMER: タイマのリセット (S7-1200, S7-1500)
- PRESET_TIMER: 持続時間のロード (S7-1200, S7-1500)
- レガシー (S7-1500)

TP: パルスの生成

説明

「パルス生成」命令を使用して、持続時間 PT の Q パラメータを設定できます。 この命令は、IN パラメータの論理演算結果(RLO)が「0」から「1」(信号の立ち上がりエッジ)に切り替わるときに開始します。 命令が開始されると、プログラムされた時間 PT が開始します。 以降の入力信号の切り替えに関わらず、Q パラメータが時間 PT に対してセットされます。 信号の新しい立ち上がりエッジが検出されても、持続時間 PT が有効である限り、Q パラメータのシグナル状態は影響を受けません。

現在の時間値は、ET パラメータ内で照会することができます。 時間値は T#0s で開始し、持続時間 PT の値に達すると終了します。 持続時間 PT が経過したときに IN パラメータのシグナル状態が「0」の場合、ET パラメータはリセットされます。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、出力 ET はタイマの期限が切れるとすぐに定数値を返します。

「パルス生成」命令の各呼び出しは、命令データが保存される IEC タイマに割り当てる必要があります。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TP_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- TP_TIME タイプのローカルタグとして、ブロックの「Static」セクション内で宣言(#MyTP_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TP_TIME」または「TP_LTIME」のローカルタグとしての宣言(#MyTP_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET にアクセスするたびに更新されます。

構文

「パルス生成」命令には、以下の構文を使用します。

- システムデータタイプ IEC_Timer (グローバル DB)のデータブロック:

```
SCL
<IEC_Timer_DB> TP(IN := <Operand>,
                  PT := <Operand>,
                  Q => <Operand>,
                  ET => <Operand>)
```

- ローカルタグ:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

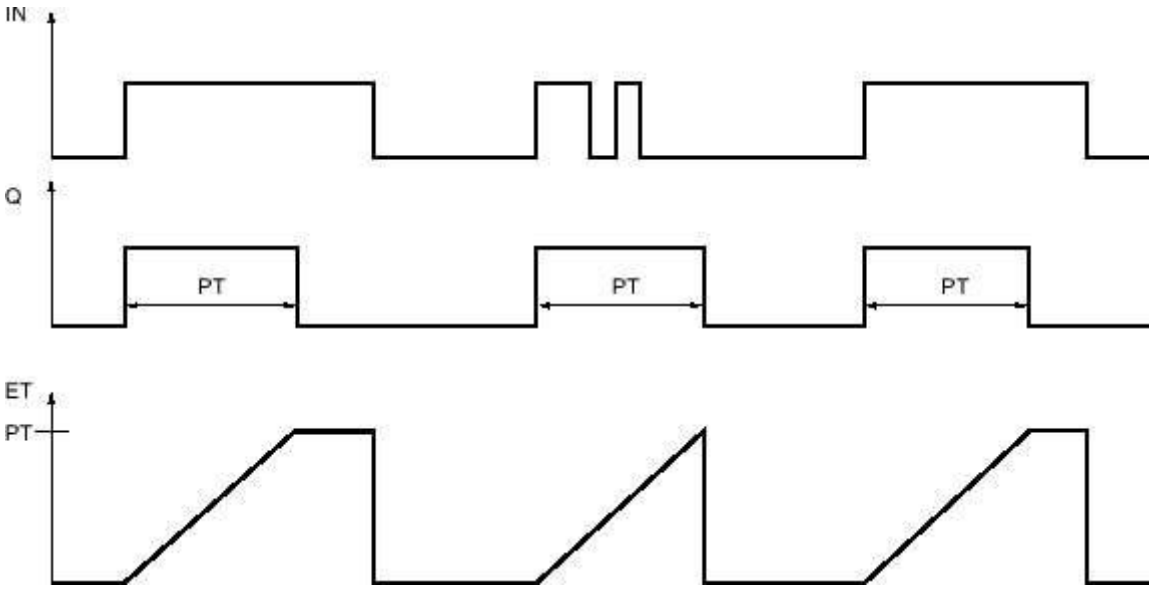
命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I、Q、M、D、L、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、P	パルスの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L、P	PT 持続時間にセットされたオペランド。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「パルスタイマ」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
SCL
"TP_DB".TP(IN := "Tag_Start",
           PT := "Tag_PresetTime",
           Q => "Tag_Status",
           ET => "Tag_ElapsedTime");
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わるときに、PT パラメータに対してプログラムされた時間が開始され、オペランド「Tag_Status」が「1」にセットされます。 現在の時間値は、オペランド「Tag_ElapsedTime」内に保存されます。

TON: オンディレータイマ

説明

「オンディレータイマ」命令を使用し、プログラムされた持続時間 PT の Q パラメータのセットを遅延することができます。 この命令は、IN パラメータの論理演算結果(RLO)が「0」から「1」(信号の立ち上がりエッジ)に切り替わるときに開始します。 命令が開始されると、プログラムされた時間 PT が開始します。 持続時間 PT が経過すると、Q パラメータがシグナル状態「1」を返します。 Q パラメータは、開始入力が「1」の間はセットされた状態が続きます。 IN パラメータのシグナル状態が「1」から「0」に切り替わると、Q パラメータがリセットされます。 IN パラメータで信号の新しい立ち上がりエッジが検出されると、タイマファンクションは再開されます。

現在の時間値は、ET パラメータ内で照会することができます。 時間値は T#0s で開始し、持続時間 PT の値に達すると終了します。 IN パラメータがシグナル状態「0」に切り替わると、ET パラメータは直ちにリセットされます。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、出力 ET はタイマの期限が切れるとすぐに定数値を返します。

「オンディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TON_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- TON_TIME タイプのローカルタグとして、ブロックの「Static」セクション内で宣言(#MyTON_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TON_TIME」または「TON_LTIME」のローカルタグとしての宣言(#MyTON_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET にアクセスするたびに更新されます。

構文

「オンディレータイマ」命令には、以下の構文を使用します。

- システムデータタイプ IEC_Timer (グローバル DB)のデータブロック:

```
SCL
<IEC_Timer_DB> TON(IN := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- ローカルタグ:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

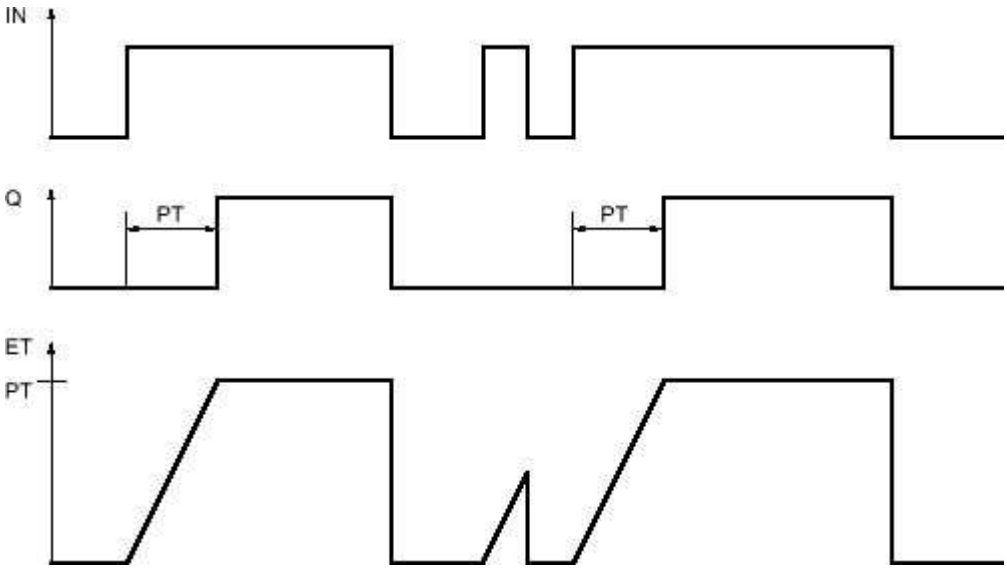
命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I、Q、M、D、L、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、P	オンディレーの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L、P	タイマ PT の経過時にセットされるオペランド。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オンディレータイマ」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
SCL
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わると、PT パラメータにプログラムされた時間が開始します。 時間の終了後、オペランド「Tag_Status」がシグナル状態「1」にセットされます。 オペランド「Tag_Start」のシグナル状態が「1」である限り、オペランド「Tag_Status」はシグナル状態「1」がセットされた状態のまま維持されます。 現在の時間値は、オペランド「Tag_ElapsedTime」内に保存されます。 オペランド「Tag_Start」のシグナル状態が「1」から「0」に切り替わると、オペランド「Tag_Status」がリセットされます。

TOF: オフディレータイマ

説明

「オフディレータイマ」命令を使用し、プログラムされた持続時間 PT の Q パラメータのリセットを遅延することができます。 Q パラメータは、IN パラメータの論理演算結果(RLO)が「0」から「1」(信号の立ち上がりエッジ)に切り替わるときにセットされます。 IN パラメータがシグナル状態「0」に戻ると、プログラムされた時間 PT が開始します。 Q パラメータは、持続時間 PT が継続中の場合はセットされた状態が続きます。 時間 PT が経過すると、Q パラメータはリセットされます。 持続時間 PT が経過する前に IN パラメータのシグナル状態が「1」に切り替わると、タイマがリセットされます。 Q パラメータのシグナル状態は、「1」にセットされた状態が継続します。

現在の時間値は、ET パラメータ内で照会することができます。 時間値は T#0s で開始し、持続時間 PT の値に達すると終了します。 持続時間 PT が経過すると、ET パラメータは、IN パラメータが「1」に戻るまで、現在の値がセットされた状態が続きます。 時間 PT が経過する前に IN パラメータが「1」に切り替わると、ET パラメータは値 T#0s にリセットされます。

注記
タイマがプログラム内でたとえばスキップされて呼び出されない場合は、出力 ET はタイマの期限が切れるとすぐに定数値を返します。

「オフディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TOF_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- TOF_TIME タイプのローカルタグとして、ブロックの「Static」セクション内で宣言(#MyTOF_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TOF_TIME」または「TOF_LTIME」のローカルタグとしての宣言(#MyTOF_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET にアクセスするたびに更新されます。

構文

「オフディレータイマ」命令には、以下の構文を使用します。

- システムデータタイプ IEC_Timer (グローバル DB)のデータブロック:

```
SCL
<IEC_Timer_DB> TOF(IN := <Operand>,
                   PT := <Operand>,
                   Q => <Operand>,
                   ET => <Operand>)
```

- ローカルタグ:

```
SCL
#myLocal_timer(IN := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

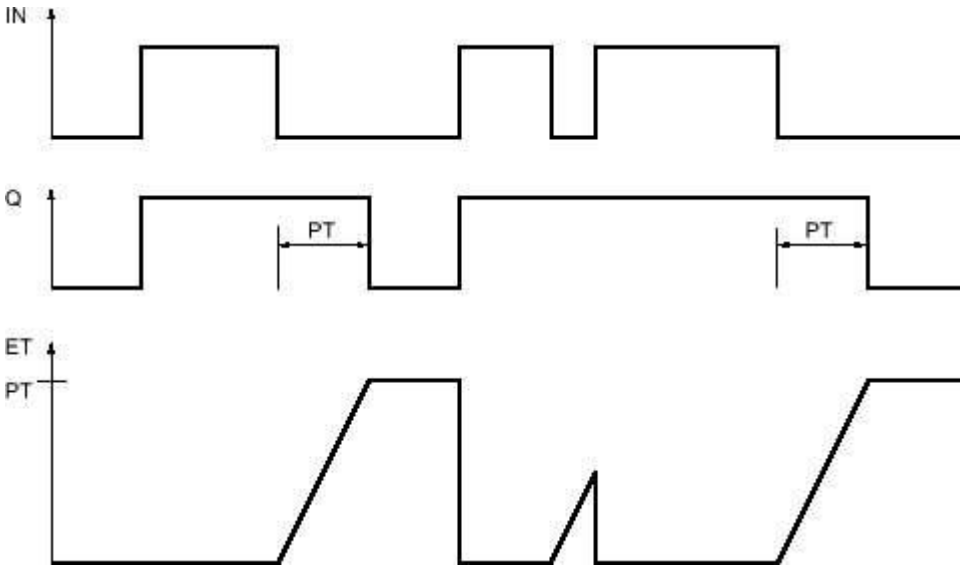
命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I、Q、M、D、L、P	開始入力
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、P	オフディレーの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L、P	時間 PT の経過時にリセットされるオペランド。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L、P	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オフディレータイマ」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
SCL
"TOF_DB".TOF(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わると、オペランド「Tag_Status」がセットされます。 オペランド「Tag_Start」のシグナル状態が「1」から「0」に切り替わると、PT パラメータにプログラムされた時間が開始します。 時間が継続中の場合は、オペランド「Tag_Status」はセットされた状態が続きます。 時間が経過すると、オペランド「Tag_Status」はリセットされます。 現在の時間値は、オペランド「Tag_ElapsedTime」内に保存されます。

TONR: タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用して、PT パラメータによってセットされた時間内の時間値を累積します。 IN パラメータのシグナル状態が「1」に切り替わると命令が実行され、持続時間 PT が開始します。 持続時間 PT が継続中の場合は、IN パラメータのシグナル状態が「1」の時に記録された時間値が累積されます。 累積時間は ET パラメータ内に出力され、そこで照会することができます。 持続時間 PT が経過すると、Q パラメータのシグナル状態は「1」になります。 IN パラメータがシグナル状態「0」に切り替わっても、Q パラメータは「1」にセットされた状態が継続します。

IN パラメータのシグナル状態に関係なく、R パラメータは ET および Q パラメータをリセットします。

「タイムアキュムレータ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。

S7-1200 CPU の場合

IEC タイマは、データタイプ IEC_TIMER または TONR_TIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- TONR_TIME タイプのローカルタグとして、ブロックの「Static」セクション内で宣言(#MyTONR_TIMER など)

S7-1500 CPU の場合

IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ「TONR_TIME」または「TONR_LTIME」のローカルタグとしての宣言(#MyTONR_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET にアクセスするたびに更新されます。

構文

「タイムアキュムレータ」命令には、以下の構文を使用します。

- システムデータタイプ IEC_Timer (グローバル DB)のデータブロック:

```
SCL
<IEC_Timer_DB> TONR(IN := <Operand>,
```

```
                R := <Operand>,
                PT := <Operand>,
                Q => <Operand>,
                ET => <Operand>)
```

- ローカルタグ:

```
SCL
#myLocal_timer(IN := <Operand>,
               R := <Operand>,
               PT := <Operand>,
               Q => <Operand>,
               ET => <Operand>)
```

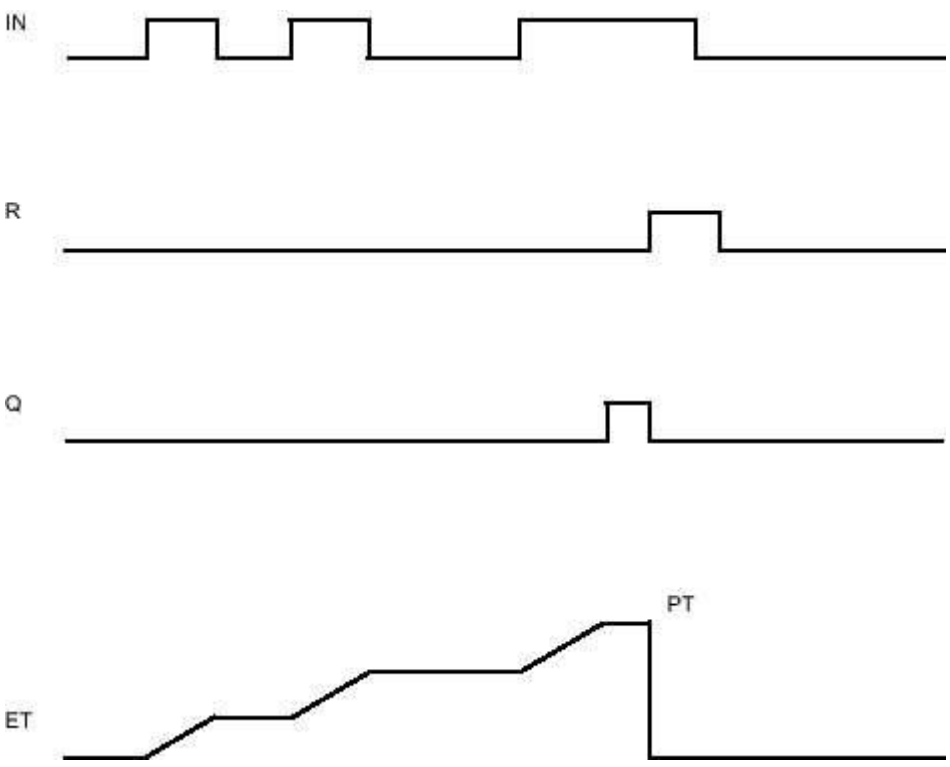
命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	BOOL	BOOL	I、Q、M、D、L、P	開始入力
R	Input	BOOL	BOOL	I、Q、M、D、L、P	ET および Q パラメータのリセット
PT	Input	TIME	TIME, LTIME	I、Q、M、D、L、P	時間記録の最大持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	BOOL	I、Q、M、D、L、P	タイマ PT の経過時にセットされた状態で維持されるオペランド。
ET	Output	TIME	TIME, LTIME	I、Q、M、D、L、P	累積時間

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「タイムアキュムレータ」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
SCL
"TONR_DB".TONR(IN := "Tag_Start",
               R := "Tag_Reset",
               PT := "Tag_PresetTime",
               Q => "Tag_Status",
               ET => "Tag_Time");
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わると、PT パラメータにプログラムされた時間が開始します。 タイマの継続中は、オペランド「Tag_Start」のシグナル状態が「1」のときに記録された時間値が累積されます。 累積時間はオペランド「Tag_Time」内に保存されます。 PT パラメータに表示される時間値に達すると、「Tag_Status」オペランドがシグナル状態「1」にセットされます。 現在の時間値は、オペランド「Tag_Time」内に保存されます。

RESET_TIMER: タイマのリセット

説明

「タイマのリセット」命令を使用し、IEC タイマを「0」にリセットすることが可能です。 指定されたデータブロック内のタイマの構造体コンポーネントが「0」にリセットされます。

この命令は、RLO に影響を与えません。 TIMER パラメータで、「タイマのリセット」命令に、プログラムで宣言された IEC タイマが割り当てられます。 この命令は、IF 命令でプログラミングされる必要があります。 命令のデータは命令が呼び出されるときのみ更新され、割り当てられた IEC タイマがアクセスされるたびには更新されません。 データを照会する場合、命令の呼び出しから次の命令の呼び出しまでのみ同一になります。

構文

「タイマのリセット」命令には、以下の構文を使用します。

SCL
```
RESET_TIMER(TIMER := <IEC timer>)
```

パラメータ

次の表に、「タイマのリセット」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER、IEC_LTIMER、TP_TIME、TP_LTIME、TON_TIME、TON_LTIME、TOF_TIME、TOF_LTIME、TONR_TIME、TONR_LTIME	D、L	リセットされる IEC タイマ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
END_IF;

IF "TON_DB".ET < T#25s THEN
RESET_TIMER(TIMER := "TON_DB");
#started := false;
END_IF;
```

タグ#started のシグナル状態が「0」であるときに、オペランド「Tag_Start」で信号立ち上がりエッジが発生すると、「オンディレータイマ」命令が実行されます。 インスタンスデータブロック「TON_DB」に保存された IEC タイマは、オペランド「Tag_PresetTime」で指定された持続時間で起動します。 オペランド「Tag_Status」は、オペランド「Tag_PresetTime」によって指定された持続時間が期限切れになるとセットされます。 オペランド「Tag_Start」のシグナル状態が「1」である限り、パラメータ Q はセットされた状態となります。 開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータのオペランドがリセットされます。

IEC タイマ「TON_DB」の期限切れ時間が 25 秒未満である場合、「タイマのリセット」命令が実行され、「TON_DB」インスタンスデータブロックに格納されているタイマがリセットされます。

PRESET_TIMER: 持続時間のロード

説明

「持続時間のロード」命令を使用して、IEC タイマの時間を設定することが可能です。 命令の入力において、論理演算の結果(RLO)のシグナル状態が「1」の場合、サイクルごとにこの命令を実行します。 この命令を実行すると、指定された IEC タイマの構造体に指定された時間を書き込みます。

注記
命令の実行中に、指定された IEC タイマが動作している場合、指定された IEC タイマの現在の時刻が上書きされます。 これによって、IEC タイマのタイマステータスを変更できます。

プログラムで宣言した IEC タイマを「持続時間のロード」命令に割り当てます。

命令のデータは命令が呼び出されるとき、および割り当てられた IEC タイマがアクセスされるたびにのみ更新されます。 Q または ET (たとえば"MyTimer".Q または"MyTimer".ET)で照会すると、IEC_TIMER 構造体が更新されます。

構文

「持続時間のロード」命令には、以下の構文を使用します。

```
SCL
PRESET_TIMER(PT := <Operand>,
             TIMER := <IEC timer)
```

パラメータ

次の表に、「持続時間のロード」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Time duration>	Input	TIME	TIME, LTIME	I、Q、M、D、L	IEC タイマが動作する時間。
<IEC timer>	Output	IEC_TIMER, TP_TIME, TON_TIME, TOF_TIME, TONR_TIME	IEC_TIMER, IEC_LTIMER, TP_TIME, TP_LTIME, TON_TIME, TON_LTIME, TOF_TIME, TOF_LTIME, TONR_TIME, TONR_LTIME	D、L	持続時間を設定する IEC タイマ。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
IF #started = false THEN
"TON_DB".TON(IN := "Tag_Start",
             PT := "Tag_PresetTime",
             Q => "Tag_Status",
             ET => "Tag_ElapsedTime");
#started := true;
#preset = true
END_IF;

IF "TON_DB".ET < T#10s AND #preset = true THEN
PRESET_TIMER(PT := T#25s,
             TIMER := "TON_DB");
#preset := false;
END_IF;
```

タグ#started のシグナル状態が「0」であるときに、オペランド「Tag_Start」で信号立ち上がりエッジが発生すると、「オンディレータイマ」命令が実行されます。 インスタンスデータブロック「TON_DB」に保存された IEC タイマは、オペランド「Tag_PresetTime」で指定された持続時間で起動します。 オペランド「Tag_Status」は、オペランド「Tag_PresetTime」によって指定された持続時間 PT が期限切れになるとセットされます。 オペランド「Tag_Start」のシグナル状態が「1」である限り、パラメータ Q はセットされた状態となります。 開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータのオペランドがリセットされます。

IEC タイマ「TON_DB」の期限切れ時間が 10 秒未満であり、タグ#preset のシグナル状態が「1」であるときは、「持続時間のロード」命令が実行されます。 この命令がパラメータ PT で指定される持続時間をインスタンスデータブロック「TON_DB」に書き込む。これにより、インスタンスデータブロック内のオペランド「Tag_PresetTime」の時間値が上書きされる。 タイマステータスのシグナル状態は、次の照会時、または "TON_DB".Q または "TON_DB".ET へのアクセス時に変化する可能性があります。

レガシー

この章には下記に関する情報が記載されています：

- S_PULSE: パルスタイマパラメータの割り当てと開始 (S7-1500)
- S_PEXT: 拡張パルスタイマパラメータの割り当てと開始 (S7-1500)
- S_ODT: オンディレータイマパラメータの割り当てと開始 (S7-1500)
- S_ODTS: 保持型オンディレータイマパラメータの割り当てと開始 (S7-1500)
- S_OFFDT: オフディレータイマパラメータの割り当てと開始 (S7-1500)

S_PULSE: パルスタイマパラメータの割り当てと開始

説明

「パルスタイマパラメータの割り当てと開始」命令は、S パラメータの論理演算結果(RLO)内で「0」から「1」(信号立ち上がりエッジ)の切り替えを検出すると、T_NO パラメータ内でプログラムされた時間を開始します。 このタイマは、S パラメータのシグナル状態が「1」である間は、プログラムされた時間(TV)の間動作します。

プログラムされた時間が経過する前に S パラメータがシグナル状態「0」に切り替わると、タイマは停止し Q パラメータは「0」にリセットされます。

内部的には、この時間は時間値とタイムベースで構成されていて、TV パラメータ内にプログラムされます。 この命令が開始すると、プログラムされた時間値は 0 までカウントダウンされます。 タイムベースは、時間値の変更単位となる時間インクリメントを指定します。 現在の時間値は、BI パラメータで与えられます。

タイマが動作していて、かつ入力 R のシグナル状態が「1」に切り替わった場合、現在の時間値およびタイムベースともに 0 にセットされます。 タイマが動作していない場合、R 入力のシグナル状態が「1」であっても効果はありません。

タイマが動作しているときに S パラメータのシグナル状態が「1」の場合、Q パラメータがシグナル状態「1」を返します。 プログラムされた時間が経過する前に S パラメータがシグナル状態「0」に切り替わると、Q パラメータはシグナル状態「0」を返します。 タイマがパラメータ R によってリセットされるか、またはタイマの時間を経過すると、パラメータ Q もシグナル状態「0」を返します。

命令データはアクセスのたびに更新されます。 そのため、サイクル開始時に照会されるデータがサイクル終了時の値とは異なる可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「パルスタイマパラメータの割り当てと開始」命令には次の構文を使用します。

```
SCL
S_PULSE(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI => <Operand>)
```

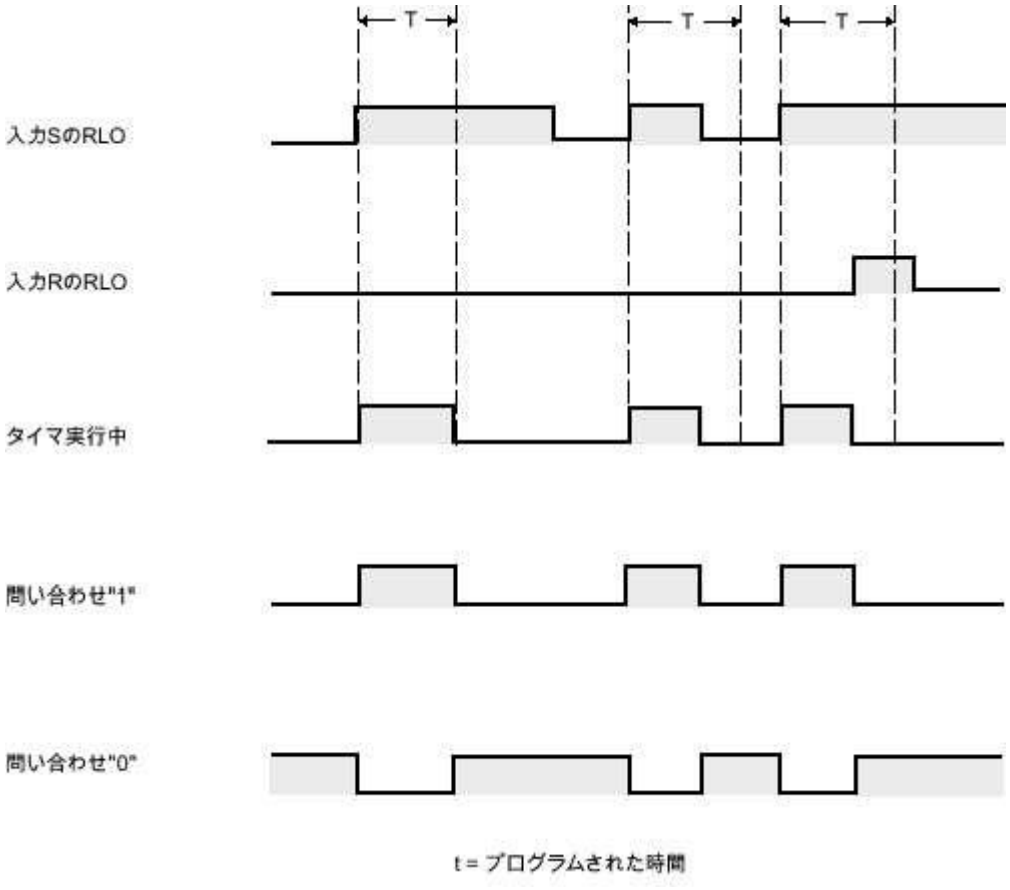
命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
T_NO	Input	TIMER, INT	T	開始されるタイマ。 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L	プリセット時間値
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス
BI	Output	WORD	I、Q、M、D、L、P	現在のデュアルコーディングされたタイマ値
ファンクション値		S5TIME	I、Q、M、D、L	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「パルスタイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := S_PULSE(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
                        Q := "Tag_Status",
                        BI := "Tag_Value");
```

"オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、「Timer_1」が開始します。タイマは、オペランド「Tag_1」がシグナル状態「1」を返すまでオペランド「Tag_Number」の時間値でカウントダウンします。

プログラムされた時間が経過する前に S パラメータがシグナル状態「0」に切り替わると、オペランド「Tag_Status」が「0」にリセットされます。 タイマが R パラメータによってリセットされるか、またはタイマの時間を経過すると、「Tag_Status」アドレスもシグナル状態「0」を返します。

現在の時間値は二重符号化されてオペランド「Tag_Value」に格納され、かつファンクション値として返されます。

S_PEXT: 拡張パルスタイマパラメータの割り当てと開始

説明

「拡張パルスタイマパラメータの割り当てと開始」命令は、S パラメータで信号の立ち上がり工ッジが検出されると、プログラムされたタイマを開始します。 このタイマは、S パラメータがシグナル状態「0」に切り替わっても、プログラムされた時間(TV)実行されます。 タイマが実行されている間は、Q パラメータがシグナル状態「1」を返します。

タイマの時間が経過すると、パラメータ Q が「0」にリセットされます。 タイマの実行中に S パラメータでシグナル状態が「0」から「1」に切り替わると、タイマは TV パラメータでプログラムされた時間で再開されます。

内部的には、この時間は時間値とタイムベースで構成されていて、TV パラメータ内にプログラムされます。 この命令が開始すると、プログラムされた時間値は 0 までカウントダウンされます。 タイムベースは、時間値の変更単位となる時間インクリメントを指定します。 現在の時間値は、BI パラメータで与えられます。

タイマが動作しているときにパラメータ R のシグナル状態が「1」に切り替わった場合、現在の時間値およびタイムベースともに 0 にセットされます。 タイマが動作していない場合、パラメータ R のシグナル状態が「1」であっても効果はありません。

命令データはアクセスのたびに更新されます。 そのため、サイクル開始時に照会されるデータがサイクル終了時の値とは異なる可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「拡張パルスタイマパラメータの割り当てと開始」命令には次の構文を使用します。

```
SCL
S_PEXT(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R:= <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

命令の構文は以下の部分で構成されます。

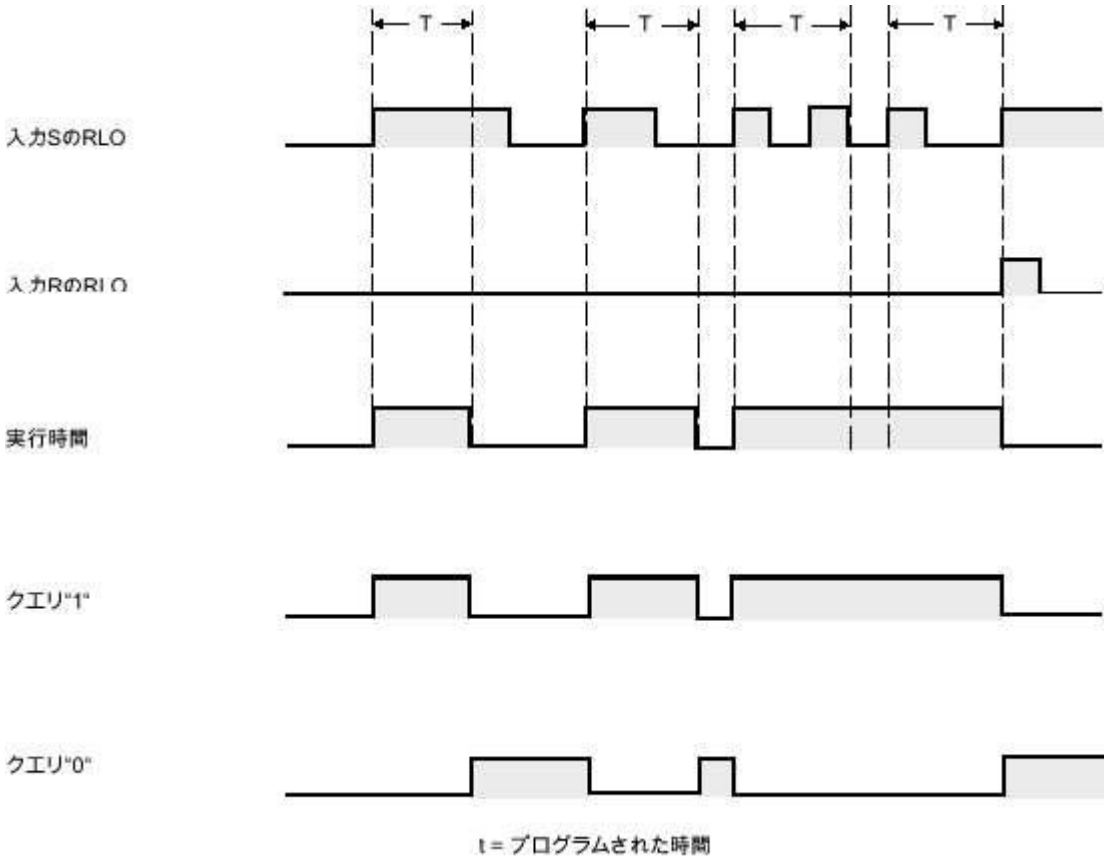
パラメータ	宣言	データタイプ	メモリ領域	説明
T_NO	Input	TIMER, INT	T	開始されるタイマ。

				タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L	プリセット時間値
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス
BI	Output	WORD	I、Q、M、D、L、P	現在のデュアルコーディングされたタイマ値
ファンクション値		S5TIME	I、Q、M、D、L	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「拡張パルスタイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。

```
SCL
"Tag_Result" := S_PEXT(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
                       BI := "Tag_Value");
```

"オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、「Timer_1」が開始します。タイマが実行されている間は、オペランド「Tag_Status」がシグナル状態「1」を返します。 タイマの時間が経過すると、オペランド「Tag_Status」が「0」にリセットされます。 タイマの実行中に S 入力でシグナル状態が「0」から「1」に切り替わると、タイマは時間「Tag_Number」で再開されます。

現在の時間値は二重符号化されてオペランド「Tag_Value」に格納され、かつファンクション値として返されます。

S_ODT: オンディレータイマパラメータの割り当てと開始

説明

「オンディレータイマパラメータの割り当てと開始」命令は、S パラメータで信号の立ち上がりエッジが検出されると、プログラムされたタイマをオンディレーとして開始します。 このタイマは、S パラメータのシグナル状態が「1」である間は、プログラムされた時間(TV)の間動作します。

タイマの時間が正常に経過しているときにパラメータ S のシグナル状態が「1」のままである場合、パラメータ Q がシグナル状態「1」を返します。 タイマの実行中に S パラメータのシグナル状態が「1」から「0」に切り替わると、タイマは停止します。 この場合、出力 Q がシグナル状態「0」にリセットされます。

内部的には、この時間は時間値とタイムベースで構成されていて、TV パラメータ内にプログラムされます。 この命令が開始すると、プログラムされた時間値は 0 までカウントダウンされます。 タイムベースは、時間値の変更単位となる時間インクリメントを指定します。 現在の時間値は、BI パラメータで与えられます。

タイマが動作していて、かつ入力 R のシグナル状態が「0」から「1」に切り替わった場合、現在の時間値およびタイムベースともに 0 にセットされます。 この場合、パラメータ Q のシグナル状態は「0」です。 タイマが実行されていない場合や、S パラメータの論理演算結果(RLO)が「1」である場合でも、R パラメータのシグナル状態が「1」であればタイマはリセットされます。

命令データはアクセスのたびに更新されます。 そのため、サイクル開始時に照会されるデータがサイクル終了時の値とは異なる可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「オンディレータイマパラメータの割り当てと開始」命令には次の構文を使用します。

```
SCL
S_ODT(T_NO := <Operand>,
      S := <Operand>,
      TV := <Operand>,
      R := <Operand>,
      Q => <Operand>,
      BI =><Operand>)
```

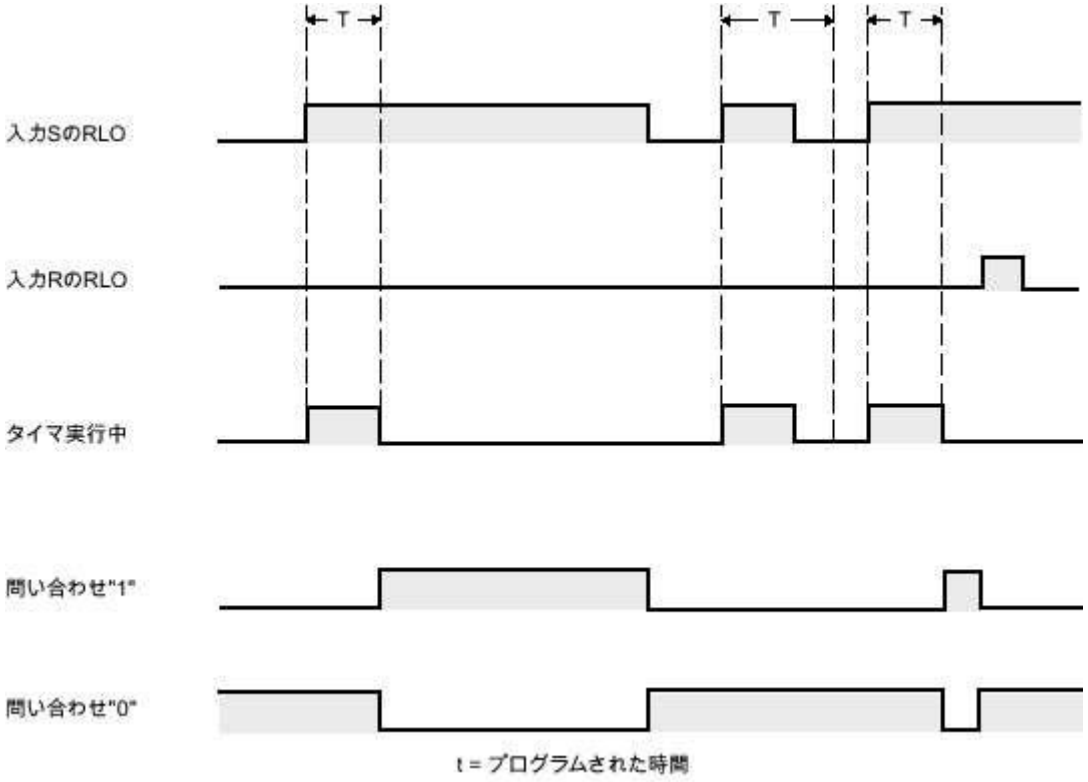
命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
T_NO	Input	TIMER, INT	T	開始されるタイマ。 タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L	プリセット時間値
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス
BI	Output	WORD	I、Q、M、D、L、P	現在のデュアルコーディングされたタイマ値
ファンクション値		S5TIME	I、Q、M、D、L	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オンディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。

```
SCL
"Tag_Result" := S_ODT(T_NO := "Timer_1",
                      S := "Tag_1",
                      TV := "Tag_Number",
                      R := "Tag_Reset",
                      Q := "Tag_Status",
                      BI := "Tag_Value");
```

"オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、「Timer_1」が開始します。このタイマは、オペランド「Tag_1」のシグナル状態が「1」である間は、持続時間「Tag_Number」の間動作します。

タイマの時間が正常に経過し、かつオペランド「Tag_Status」のシグナル状態が「1」である場合は、オペランド「Tag_Status」がシグナル状態「1」にリセットされます。 タイマの実行中にオペランド「Tag_1」のシグナル状態が「1」から「0」に切り替わった場合、タイマは停止します。 この場合、オペランド「Tag_Status」はシグナル状態「0」を返します。

現在の時間値は二重符号化されてオペランド「Tag_Value」に格納され、かつファンクション値として返されます。

S_ODTS: 保持型オンディレータイマパラメータの割り当てと開始

説明

「保持型オンディレータイマパラメータの割り当てと開始」命令は、S パラメータで信号の立ち上がりエッジが検出されると、プログラムされたタイマを開始します。 このタイマは、S パラメータがシグナル状態「0」に切り替わっても、プログラムされた時間(TV)実行されます。

タイマの期限が切れた場合、Q パラメータは、S パラメータでのシグナル状態に関わらず「1」を返します。 タイマの実行中に S パラメータでシグナル状態が「0」から「1」に切り替わると、タイマはプログラムされた時間「TV」で再開されます。

内部的には、この時間は時間値とタイムベースで構成されていて、TV パラメータ内にプログラムされます。 この命令が開始すると、プログラムされた時間値は 0 までカウントダウンされます。 タイムベースは、時間値の変更単位となる時間インクリメントを指定します。 現在の時間値は、BI パラメータで与えられます。

R パラメータのシグナル状態「1」は、S パラメータのシグナル状態に関わらず、現在の時間値およびタイムベースを「0」にリセットします。 この場合、パラメータ Q のシグナル状態は「0」です。

命令データはアクセスのたびに更新されます。 そのため、サイクル開始時に照会されるデータがサイクル終了時の値とは異なる可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「保持型オンディレータイマパラメータの割り当てと開始」命令には次の構文を使用します。

```
SCL
S_ODTS(T_NO := <Operand>,
       S := <Operand>,
       TV := <Operand>,
       R := <Operand>,
       Q => <Operand>,
       BI =><Operand>)
```

命令の構文は以下の部分で構成されます。

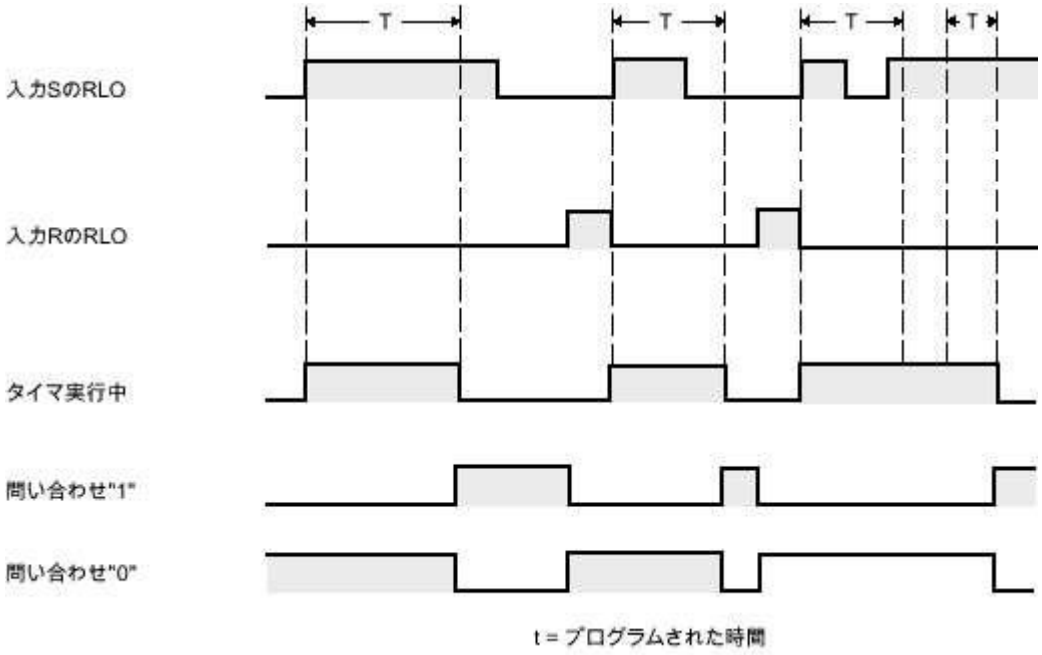
パラメータ	宣言	データタイプ	メモリ領域	説明
T_NO	Input	TIMER, INT	T	開始されるタイマ。 タイマの数は CPU によって異なります。

S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L	プリセット時間値
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス
BI	Output	WORD	I、Q、M、D、L、P	現在のデュアルコーディングされたタイマ値
ファンクション値		S5TIME	I、Q、M、D、L	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「保持型オンディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := S_ODTS(T_NO := "Timer_1",
                       S := "Tag_1",
                       TV := "Tag_Number",
                       R := "Tag_Reset",
                       Q := "Tag_Status",
```

```
                    BI := "Tag_Value");
```

"オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、「Timer_1」が開始します。タイマは持続時間「Tag_Number」の間動作します。

タイマの時間が経過すると、オペランド「Tag_Status」はオペランド「Tag_1」のシグナル状態に関わらずシグナル状態「1」を返します。 タイマの実行中にオペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、タイマは時間「Tag_Number」で再開されます。

現在の時間値は二重符号化されてオペランド「Tag_Value」に格納され、かつファンクション値として返されます。

S_OFFDT: オフディレータイマパラメータの割り当てと開始

説明

「オフディレータイマパラメータの割り当てと開始」命令は、S パラメータで信号の立ち下がりエッジが検出されると、プログラムされた時間を開始します。 タイマはプログラムされた時間「TV」の間動作します。 タイマが動作している場合、または S パラメータがシグナル状態「1」を返す場合は、Q パラメータのシグナル状態は「1」となります。

タイマの時間が経過し、かつシグナル状態が「0」の場合は、Q パラメータがシグナル状態「0」にリセットされます。 タイマの実行中にパラメータ S のシグナル状態が「0」から「1」に切り替わると、タイマは停止します。 タイマは、S パラメータで信号立ち下がりエッジが検出された後にのみ再開されます。

内部的には、この時間は時間値とタイムベースで構成されていて、TV パラメータ内にプログラムされます。 この命令が開始すると、プログラムされた時間値は 0 までカウントダウンされます。 タイムベースは、時間値の変更単位となる時間インクリメントを指定します。 現在の時間値は、BI パラメータで与えられます。

R パラメータのシグナル状態「1」は、現在の時間値およびタイムベースを「0」にリセットします。この場合、パラメータ Q のシグナル状態は「0」です。

命令データはアクセスのたびに更新されます。 そのため、サイクル開始時に照会されるデータがサイクル終了時の値とは異なる可能性があります。

注記
時間セルでは、オペレーティングシステムは、時間値が「0」と等しくなるまで、タイムベースによって指定された間隔で、時間値を 1 単位ずつ減少させます。 このデクリメントは、ユーザープログラムに対して非同期的に行われます。 このため、結果のタイマは目的のタイムベースよりも最大で間隔が 1 時間短くなります。

時間セルを形成する方法の例は、以下で確認できます。 関連項目「L: タイマ値のロード」。

構文

「オフディレータイマパラメータの割り当てと開始」命令には次の構文を使用します。

```
SCL
S_OFFDT(T_NO := <Operand>,
        S := <Operand>,
        TV := <Operand>,
        R := <Operand>,
        Q => <Operand>,
        BI =><Operand>)
```

命令の構文は以下の部分で構成されます。

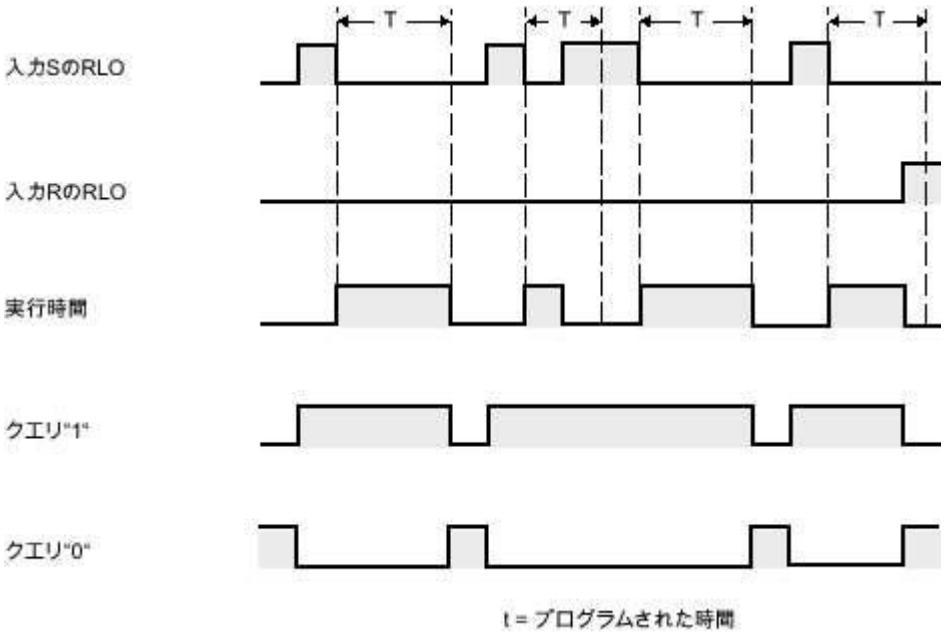
パラメータ	宣言	データタイプ	メモリ領域	説明
T_NO	Input	TIMER, INT	T	開始されるタイマ。

				タイマの数は CPU によって異なります。
S	Input	BOOL	I、Q、M、D、L	開始入力
TV	Input	S5TIME, WORD	I、Q、M、D、L	プリセット時間値
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
Q	Output	BOOL	I、Q、M、D、L、P	タイマのステータス
BI	Output	WORD	I、Q、M、D、L、P	現在のデュアルコーディングされたタイマ値
ファンクション値		S5TIME	I、Q、M、D、L	現在の時間値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスタイミング図

次の図に、「オフディレータイマパラメータの割り当てと開始」命令のパルスタイミング図を示します。



例

次の例で、命令がどのように機能するかを示します。

```
SCL
"Tag_Result" := S_OFFDT(T_NO := "Timer_1",
                        S := "Tag_1",
                        TV := "Tag_Number",
                        R := "Tag_Reset",
```

```
                    Q := "Tag_Status",
                    BI := "Tag_Value");
```

"オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、「Timer_1」が開始します。タイマは持続時間「Tag_Number」の間動作します。 タイマが動作している場合、またはオペランド「Tag_1」がシグナル状態「1」を返す場合は、オペランド「Tag_Status」のシグナル状態は「1」となります。

タイマの時間が経過し、かつオペランド「Tag_1」のシグナル状態が「0」の場合は、オペランド「Tag_Status」がシグナル状態「0」にリセットされます。 タイマの実行中にオペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わった場合、タイマがリセットされます。 タイマは、S パラメータで立ち下がりエッジが検出された後にのみ再開されます。

現在の時間値は二重符号化されてオペランド「Tag_Value」に格納され、かつファンクション値として返されます。

カウンタ演算

この章には下記に関する情報が記載されています：

- CTU: カウントアップ (S7-1200, S7-1500)
- CTD: カウントダウン (S7-1200, S7-1500)
- CTUD: カウントアップ/カウントダウン (S7-1200, S7-1500)
- レガシー (S7-1500)

CTU: カウントアップ

説明

CV パラメータの値のインクリメントには、「カウントアップ」命令を使用できます。 CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わるとこの命令が実行され、CV パラメータのカウンタ現在値が 1 インクリメントされます。 立ち上がりエッジを検出するたびに、CV パラメータにおいて指定されたデータタイプの上限値に達するまで、カウンタ値がインクリメントされます。 上限値に達すると、CU パラメータのシグナル状態はこの命令には影響を与えなくなります。

Q パラメータのカウンタステータスを照会できます。 Q パラメータのシグナル状態は、PV パラメータによって決まります。 カウンタ現在値が PV パラメータの値以上の場合、Q パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、Q パラメータのシグナル状態は「0」です。PV パラメータには定数も指定できます。

R パラメータのシグナル状態が「1」に切り替わると、CV パラメータの値はゼロにリセットされます。 R パラメータのシグナル状態が「1」である限り、CU パラメータのシグナル状態はこの命令には影響を与えません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTU_SINT / CTU_USINT • CTU_INT / CTU_UINT • CTU_DINT / CTU_UDINT • CTU_LINT / CTU_ULINT

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- CTU_<Data type>タイプのローカルタグとして、ブロックの「Static」セクション内で宣言(#MyCTU_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。 インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

構文

「カウントアップ」命令には、以下の構文を使用します。

システムデータタイプ IEC_counter (グローバル DB)のデータブロック

```
SCL
<IEC_Counter_DB>.CTU(CU := <Operand>,
                     R := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

ローカルタグ

```
SCL
#myLocal_counter(CU := <Operand>,
                 R := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
CU	Input	BOOL	I、Q、M、D、L	カウント入力
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
PV	Input	整数	I、Q、M、D、L、P	Q 出力がセットされる値
Q	Output	BOOL	I、Q、M、D、L	カウンタステータス

CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"IEC_COUNTER_DB".CTU(CU := "Tag_Start",
                     R := "Tag_Reset",
                     PV := "Tag_PresetValue",
                     Q => "Tag_Status",
                     CV => "Tag_CounterValue");
```

「Tag_Start」オペランドのシグナル状態が「0」から「1」に切り替わると、「カウントアップ」命令が実行され、「Tag_CounterValue」カウンタ現在値が 1 インクリメントされます。 信号立ち上がりエッジが検出されるたびにカウント値がインクリメントされ、指定されたデータタイプの上限値(INT = 32767)に達するまでインクリメントされます。

カウンタ現在値がオペランド「Tag_PresetValue」の値以上の場合、「Tag_Status」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_Status」出力のシグナル状態は「0」になります。 カウンタ現在値は、オペランド「Tag_CounterValue」内に保存されます。

CTD: カウントダウン

説明

「カウントダウン」命令は、CV パラメータのデクリメントに使用します。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わるとこの命令が実行され、CV パラメータのカウンタ現在値が 1 デクリメントされます。 指定されたデータタイプの下限値に達するまで、信号の立ち上がりエッジが検出されるたびにカウンタがデクリメントされます。 下限値に達すると、CD パラメータのシグナル状態はこの命令には影響を与えなくなります。

Q パラメータのカウンタステータスを照会できます。 カウンタ現在値がゼロ以下の場合、Q パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、Q パラメータのシグナル状態は「0」です。 PV パラメータには定数も指定できます。

CV パラメータの値は、LD パラメータのシグナル状態が「1」に切り替わると PV パラメータの値にセットされます。 LD パラメータのシグナル状態が「1」である限り、CD パラメータのシグナル状態はこの命令には影響を与えません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTD_SINT / CTD_USINT • CTD_INT / CTD_UINT • CTD_DINT / CTD_UDINT • CTD_LINT / CTD_ULINT

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- CTD_<Data type>タイプのローカルタグとして、ブロックの「Static」セクション内で宣言 (#MyCTD_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成し

た場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

構文

「カウントダウン」命令には、以下の構文を使用します。

システムデータタイプ IEC_counter (グローバル DB)のデータブロック

```
SCL
<IEC_Counter_DB>.CTD(CD := <Operand>,
                     LD := <Operand>,
                     PV := <Operand>,
                     Q => <Operand>,
                     CV => <Operand>)
```

ローカルタグ

```
SCL
#myLocal_counter(CD := <Operand>,
                 LD := <Operand>,
                 PV := <Operand>,
                 Q => <Operand>,
                 CV => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
CD	Input	BOOL	I、Q、M、D、L	カウント入力
LD	Input	BOOL	I、Q、M、D、L、P	ロード入力
PV	Input	整数	I、Q、M、D、L、P	CV 出力が LD = 1 にセットされる値。
Q	Output	BOOL	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"IEC_SCOUNTER_DB".CTD(CD := "Tag_Start",
                      LD := "Tag_Load",
                      PV := "Tag_PresetValue",
                      Q => "Tag_Status",
                      CV => "Tag_CounterValue");
```

「Tag_Start」オペランドのシグナル状態が「0」から「1」に切り替わると、命令が実行され、「Tag_CounterValue」オペランドの値が 1 デクリメントされます。 指定されたデータタイプの下限値(-128)に達するまで、信号の立ち上がりエッジが検出されるたびにカウンタ値がデクリメントされます。

カウンタ現在値がゼロ以下である限り、「Tag_Status」オペランドのシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_Status」出力のシグナル状態は「0」になります。 カウンタ現在値は、オペランド「Tag_CounterValue」内に保存されます。

CTUD: カウントアップ/カウントダウン

説明

CV パラメータでのカウンタ値のインクリメント、またはデクリメントには、「カウントアップ/カウントダウン」命令を使用します。 CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、CV パラメータのカウンタ現在値が 1 インクリメントされます。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、CV パラメータのカウンタ値が 1 デクリメントされます。 プログラムサイクル内の CU および CD 入力で信号の立ち上がりエッジが存在すると、CV パラメータのカウンタ現在値は変化しません。

カウンタ値は、CV パラメータで指定したデータタイプの上限値に達するまでインクリメントすることができます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。 カウンタ値は、指定されたデータタイプの下限値に達するとデクリメントされなくなります。

LD パラメータのシグナル状態が「1」に切り替わると、CV パラメータのカウンタ値が PV パラメータの値にセットされます。 LD パラメータのシグナル状態が「1」である限り、CU パラメータおよび CD パラメータのシグナル状態は命令に対する効力を持ちません。

R パラメータのシグナル状態が「1」に切り替わると、カウンタ値は 0 にセットされます。 R パラメータのシグナル状態が「1」である限り、CU、CD パラメータおよび LD パラメータのシグナル状態の変化は「カウントアップ/カウントダウン」命令に影響しません。

QU パラメータでアップカウンタのステータスを照会できます。 カウンタ現在値が PV パラメータの値以上の場合、QU パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、QU パラメータのシグナル状態は「0」です。 PV パラメータには定数も指定できます。

QD パラメータでダウンカウンタのステータスを照会できます。 カウンタ現在値がゼロ以下の場合、QD パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、QD パラメータのシグナル状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ/カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

S7-1200 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER • IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER	• CTUD_SINT / CTUD_USINT • CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT

S7-1500 CPU の場合

システムデータタイプ IEC_<Counter> (共有 DB) のデータブロック	ローカルタグ
• IEC_SCOUNTER / IEC_USCOUNTER	• CTUD_SINT / CTUD_USINT

• IEC_COUNTER / IEC_UCOUNTER • IEC_DCOUNTER / IEC_UDCOUNTER • IEC_LCOUNTER / IEC_ULCOUNTER	• CTUD_INT / CTUD_UINT • CTUD_DINT / CTUD_UDINT • CTUD_LINT / CTUD_ULINT

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- CTUD_<Data type>タイプのローカルタグとして、ブロックの「Static」セクション内で宣言(#MyCTUD_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

構文

「カウントアップ/カウントダウン」命令には、以下の構文を使用します。

システムデータタイプ IEC_counter (グローバル DB)のデータブロック

```
SCL
<IEC_Counter_DB>.CTUD(CU := <Operand>,
                      CD := <Operand>,
                      R := <Operand>,
                      LD := <Operand>,
                      PV := <Operand>,
                      QU=> <Operand>,
                      QD := <Operand>,
                      CV => <Operand>)
```

ローカルタグ

```
SCL
myLocal_counter(CU := <Operand>,
                CD := <Operand>,
                R := <Operand>,
```

```
LD := <Operand>,
PV := <Operand>,
QU=> <Operand>,
QD := <Operand>,
CV => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
CU	Input	BOOL	I、Q、M、D、L	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L	カウントダウン入力
R	Input	BOOL	I、Q、M、D、L、P	リセット入力
LD	Input	BOOL	I、Q、M、D、L、P	ロード入力
PV	Input	整数	I、Q、M、D、L、P	QU 出力がセットされる値。 /CV 出力が LD = 1 にセットされる値。
QU	Output	BOOL	I、Q、M、D、L	アップカウンタのステータス
QD	Output	BOOL	I、Q、M、D、L	カウントダウンのステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

SCL

```
"IEC_COUNTER_DB".CTUD(CU := "Tag_Start1",
                      CD := "Tag_Start2",
                      LD := "Tag_Load",
                      R := "Tag_Reset",
                      PV := "Tag_PresetValue",
                      QU => "Tag_CU_Status",
                      QD => "Tag_CD_Status",
                      CV => "Tag_CounterValue");
```

オペランド「Tag_Start1」のシグナル状態で信号の立ち上がりエッジが検出されると、カウンタ現在値が 1 インクリメントされ、オペランド「Tag_CounterValue」に格納されます。 オペランド「Tag_Start2」のシグナル状態で信号の立ち上がりエッジが検出されると、カウンタ値が 1 デクリメントされ、これもオペランド「Tag_CounterValue」に格納されます。 カウンタ値は、指定されたデータ

タイプ(INT)の上限値に達するまで、CU パラメータの信号の立ち上がりエッジ検出時にインクリメントされます。 CD パラメータに信号の立ち上がりエッジが検出されると、指定されたデータタイプ(INT)の下限値に達するまでカウンタ値がデクリメントされます。

カウンタ現在値がオペランド「Tag_PresetValue」の値以上である限り、オペランド「Tag_CU_Status」のシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_CU_Status」出力のシグナル状態は「0」になります。

カウンタ現在値がゼロ以下である限り、「Tag_CD_Status」オペランドのシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_CD_Status」出力のシグナル状態は「0」になります。

レガシー

この章には下記に関する情報が記載されています：

- S_CU: パラメータおよびカウントアップの割り当て (S7-1500)
- S_CD: パラメータおよびカウントダウンの割り当て (S7-1500)
- S_CUD: パラメータおよびカウントアップ/カウントダウンの割り当て (S7-1500)

S_CU: パラメータおよびカウントアップの割り当て

説明

「パラメータおよびカウントアップの割り当て」命令を使用して、カウンタ値のインクリメントが可能です。 CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、カウンタ現在値が 1 インクリメントされます。 カウンタ現在値は、CV パラメータで与えられます。カウンタ値は、限界値の「999」に達するまでインクリメントされます。 制限値に達すると、信号立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。

S パラメータのシグナル状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタがセットされているときに、入力での論理演算結果 (RLO)CU が「1」の場合、信号のエッジの切り替えが検出されなくても次のサイクルでカウンタが 1 回カウントされます。

R パラメータのシグナル状態が「1」に切り替わると、カウンタ値は 0 にセットされます。 R パラメータのシグナル状態が「1」である限り、CU および S パラメータのシグナル状態が切り替わっても カウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、パラメータ Q のシグナル状態は「1」となります。 カウンタ値がゼロと等しい場合、パラメータ Q はシグナル状態「0」を返します。

注記
カウンタエラーを回避するには、カウンタはプログラム内の 1 か所でのみ使用します。

構文

「パラメータおよびカウントアップの割り当て」命令には、以下の構文を使用します。

```
SCL
S_CU(C_NO := <Operand>,
     CU := <Operand>,
     S := <Operand>,
     PV := <Operand>,
     R := <Operand>,
     Q => <Operand>,
     CV => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
C_NO	Input	COUNTER, INT	C	カウンタ カウンタの数は CPU によって異なります。
CU	Input	BOOL	I、Q、M、D、L	カウントアップ入力

S	Input	BOOL	I、Q、M、D、L	カウンタのプリセットのための入力
PV	Input	WORD	I、Q、M、D、L	BCD フォーマットのプリセット済みカウンタ値(C#0 から C#999)
R	Input	BOOL	I、Q、M、D、L	リセット入力
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値
ファンクション値		WORD, S5TIME, DATE	I、Q、M、D、L	BCD フォーマットのカウンタ現在値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := S_CU(C_NO := "Counter_1",
                     CU := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

「Tag_Start」パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わり、かつカウンタ現在値が「999」未満の場合、カウンタ値が 1 インクリメントされます。 入力「Tag_1」のシグナル状態が「0」から「1」に切り替わると、BCD フォーマットのカウンタ値がオペランド「Tag_PresetValue」の値にセットされます。 オペランド「Tag_Reset」のシグナル状態が「1」の場合、カウンタ値が「0」にリセットされます。

カウンタ現在値は、オペランド「Tag_Value」内に 16 進数形式で格納されます。

カウンタ現在値が「0」と等しくない場合、出力「Tag_Status」のシグナル状態は「1」です。 カウンタ現在値はオペランド「Tag_Value」内にファンクション値として返されます。

S_CD: パラメータおよびカウントダウンの割り当て

説明

「パラメータおよびカウントダウンの割り当て」命令を使用して、カウンタ値のデクリメントが可能です。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、カウンタ現在値が 1 デクリメントされます。 カウンタ現在値は、CV パラメータで与えられます。 カウンタ値は、下限値の「0」に達するまでデクリメントされます。 下限値に達すると、立ち上がりエッジでカウンタ値はそれ以上デクリメントされません。

S パラメータのシグナル状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタがセットされているときにパラメータ CD での論理演算結果(RLO)が「1」の場合、信号のエッジの切り替えが検出されなくても次のサイクルでカウンタが 1 回カウントされます。

R パラメータのシグナル状態が「1」に切り替わると、カウンタ値は 0 にセットされます。 R パラメータのシグナル状態が「1」である限り、CD および S パラメータのシグナル状態が切り替わってもカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、パラメータ Q のシグナル状態は「1」となります。 カウンタ値がゼロと等しい場合、パラメータ Q はシグナル状態「0」を返します。

注記
カウンタエラーを回避するには、カウンタはプログラム内の 1 か所でのみ使用します。

構文

「パラメータおよびカウントダウンの割り当て」命令には、以下の構文を使用します。

```
SCL
S_CD(C_NO:= <Operand>,
     CD  := <Operand>,
     S  := <Operand>,
     PV  := <Operand>,
     R  := <Operand>,
     Q  => <Operand>,
     CV  => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
C_NO	Input	COUNTER, INT	C	カウンタ カウンタの数は CPU によって異なります。
CD	Input	BOOL	I、Q、M、D、L	カウントダウン入力

S	Input	BOOL	I、Q、M、D、L	カウンタのプリセットのための入力
PV	Input	WORD	I、Q、M、D、L	BCD フォーマットのプリセット済みカウンタ値(C#0 から C#999)
R	Input	BOOL	I、Q、M、D、L	リセット入力
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス
CV	Output	WORD, S5TIME, WORD	I、Q、M、D、L	カウンタ現在値
ファンクション値		WORD, S5TIME, DATE	I、Q、M、D、L	BCD フォーマットのカウンタ現在値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CD := "Tag_Start",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わり(信号立ち上がりエッジ)、カウンタ現在値が「0」よりも大きい場合、カウンタ値が 1 デクリメントされます。 オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、BCD フォーマットのカウンタ値がオペランド「Tag_PresetValue」の値にセットされます。 オペランド「Tag_Reset」のシグナル状態が「1」の場合、カウンタ値が「0」にリセットされます。

カウンタ現在値は、オペランド「Tag_Value」内に保存されます。

カウンタ現在値が「0」と等しくない場合、オペランド「Tag_Status」はシグナル状態「1」を返します。 カウンタ現在値はオペランド「Tag_Value」内にファンクション値として返されます。

S_CUD: パラメータおよびカウントアップ/カウントダウンの割り当て

説明

「パラメータおよびカウントアップ/カウントダウンの割り当て」命令を使用して、カウンタ値のインクリメントおよびデクリメントが可能です。 CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、カウンタ現在値が 1 インクリメントされます。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、カウンタ値が 1 減少します。 カウンタ現在値は、CV パラメータで与えられます。 1 つのプログラムサイクル内のパラメータ CU および CD パラメータで信号の立ち上がりエッジが存在すると、カウンタ値は変化しません。

カウンタ値は、上限値の「999」に達するまでインクリメントされます。上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。下限値「0」に達すると、それ以降、カウンタ値はデクリメントされません。

S パラメータのシグナル状態が「0」から「1」に切り替わると、カウンタ値が PV パラメータの値にセットされます。 カウンタがセットされているときにパラメータ CU および CD の論理演算結果(RLO)が「1」の場合、信号のエッジの切り替えが検出されなくても次のサイクルでカウンタが 1 回カウントされます。

R パラメータのシグナル状態が「1」に切り替わると、カウンタ値は 0 にセットされます。 R パラメータのシグナル状態が「1」である限り、CU、 CD および S パラメータのシグナル状態の処理は、このカウンタ値には影響を与えません。

カウンタ値がゼロよりも大きい場合、パラメータ Q のシグナル状態は「1」となります。 カウンタ値がゼロと等しい場合、パラメータ Q はシグナル状態「0」を返します。

注記
カウンタエラーを回避するには、カウンタはプログラム内の 1 か所でのみ使用します。

構文

「パラメータおよびカウントアップ/カウントダウンの割り当て」命令には、以下の構文を使用します。

```
SCL
S_CUD(C_NO:= <Operand>,
      CU  := <Operand>,
      CD  := <Operand>,
      S   := <Operand>,
      PV  := <Operand>,
      R   := <Operand>,
      Q   => <Operand>,
      CV  => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
C_NO	Input	COUNTER, INT	C	カウンタ

				カウンタの数は CPU によって異なります。
CU	Input	BOOL	I、Q、M、D、L	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L	カウントダウン入力
S	Input	BOOL	I、Q、M、D、L	カウンタのプリセットのための入力
PV	Input	WORD	I、Q、M、D、L	BCD フォーマットのプリセット済みカウンタ値(C#0 から C#999)
R	Input	BOOL	I、Q、M、D、L	リセット入力
Q	Output	BOOL	I、Q、M、D、L	カウンタのステータス
CV	Output	WORD, S5TIME, DATE	I、Q、M、D、L	カウンタ現在値(16 進数)
ファンクション値		WORD, S5TIME, DATE	I、Q、M、D、L	BCD フォーマットのカウンタ現在値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := S_CD(C_NO := "Counter_1",
                     CU := "Tag_CU",
                     CD := "Tag_CD",
                     S := "Tag_1",
                     PV := "Tag_PresetValue",
                     R := "Tag_Reset",
                     Q => "Tag_Status",
                     CV => "Tag_Value");
```

オペランド「Tag_CU」のシグナル状態で信号の立ち上がりエッジが検出され、かつカウンタ現在値が「999」未満の場合、カウンタ値が 1 インクリメントされます。 オペランド「Tag_CD」のシグナル状態で信号の立ち上がりエッジが検出され、かつカウンタ現在値が「0」よりも大きい場合、カウンタ値が 1 デクリメントされます。

オペランド「Tag_1」のシグナル状態が「0」から「1」に切り替わると、BCD フォーマットのカウンタ値がオペランド「Tag_PresetValue」の値にセットされます。 オペランド「Tag_Reset」のシグナル状態が「1」の場合、カウンタ値が「0」にリセットされます。

カウンタ現在値は、オペランド「Tag_Value」内に保存されます。

カウンタ現在値が「0」と等しくない場合、オペランド「Tag_Status」はシグナル状態「1」を返します。 カウンタ現在値はオペランド「Tag_Value」内にファンクション値として返されます。

比較演算

この章には下記に関する情報が記載されています：

- TypeOf: VARIANT タグのデータタイプのチェック (S7-1200, S7-1500)
- TypeOfElements: VARIANT タグの配列要素のデータタイプのチェック (S7-1200, S7-1500)
- IS_ARRAY: 配列のチェック (S7-1200, S7-1500)

TypeOf: VARIANT タグのデータタイプのチェック

説明

「VARIANT タグのデータタイプのチェック」命令を使用して、VARIANT タグが指しているタグのデータタイプを照会できます。 ブロックインターフェースで宣言した<Operand>タグのデータタイプをもう 1 つのタグのデータタイプまたはデータタイプと直接比較し、それらが「等しい」か、または「等しくない」かを識別できます。

このオペランドのデータタイプは、VARIANT であることが必要です。 比較オペランドは、基本データタイプまたは PLC データタイプです。

「VARIANT タグのデータタイプのチェック」命令は、IF 命令内のみで使用できます。

構文

「VARIANT タグのデータタイプのチェック」命令には、以下の構文を使用します。

```
SCL
TypeOf(<Operand>)
```

パラメータ

次の表に、「VARIANT タグのデータタイプのチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	最初のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、別のタグとの比較を示します。

```
SCL
IF TypeOf(#TagVARIANT) = TypeOf("TagBYTE") THEN
...;
END_IF;
```

次の例で、特定のデータタイプとの比較を示します。

```
SCL
IF TypeOf(#TagVARIANT) = BYTE THEN
...;
END_IF;
```

VARIANT #TagVARIANT がポイントしているオペランドのデータタイプが BYTE である場合、比較条件が一致しました。

TypeOfElements: VARIANT タグの配列要素のデータタイプのチェック

説明

「VARIANT タグの配列要素のデータタイプのチェック」命令を使用して、VARIANT タグが指しているタグのデータタイプを照会できます。 このタグのデータタイプをブロックインターフェースで宣言したタグのデータタイプと比較し、それらが「等しい」か、または「等しくない」かを識別します。

このオペランドのデータタイプは、VARIANT であることが必要です。 比較オペランドは、基本データタイプまたは PLC データタイプです。

VARIANT タグのデータタイプが ARRAY の場合、ARRAY エレメントのデータタイプが比較されます。

この命令は IF 命令内でのみ使用できます。

構文

「VARIANT タグの配列要素のデータタイプのチェック」命令には、以下の構文を使用します。

```
SCL
TypeOfElements(<Operand>)
```

パラメータ

次の表に、「VARIANT タグの配列要素のデータタイプのチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	最初のオペランド

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
IF TypeOfElements(#Tag_VARIANT) = TypeOf("Glob-
alDB".Product[1]) THEN
"Tag_Result" := "GlobalDB".Product[1] * 3;
END_IF;
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
#Tag_VARIANT	1.5
"GlobalDB".Product[1]	3.5

VARIANT がタグをポイントし、「GlobalDB」である場合、Product[1] オペランドは REAL データタイプになり、「GlobalDB」です。次に Product[1]オペランドが 3 で乗算され、結果が「Tag_Result」オペランドに書き込まれます。

IS_ARRAY: 配列のチェック

説明

「配列のチェック」命令を使用して、VARIANT が ARRAY データタイプのタグを指すかどうかを照会できます。

<Operand>のデータタイプは、VARIANT データタイプであることが必要です。

「配列のチェック」命令は、IF 命令内のみで使用できます。

構文

「配列のチェック」命令には、以下の構文を使用します。

```
SCL
IS_ARRAY(<Operand>)
```

パラメータ

次の表に、「配列のチェック」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	ARRAY に関する照会が行われるオペランド
ファンクション値		UDINT	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
SCL
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANT);
END_IF;
```

VARIANT が指すタグが ARRAY の場合、ARRAY エレメントの数が出力されます。

四則演算

この章には下記に関する情報が記載されています：

ABS: 絶対値の形成

説明

「絶対値の形成」命令を使用して、入力値の絶対値を計算し、結果を指定したオペランド内に保存します。

構文

「絶対値の形成」命令には、次の構文を使用します。

```
SCL
ABS(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Expression>	Input	SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	入力値
ファンクション値		SINT、INT、DINT、浮動小数点数	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	入力値の絶対値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := ABS("Tag_Value");
"Tag_Result2" := ABS("Tag_Value1"*"Tag_Value2");
```

入力値の絶対値は、ファンクション値として入力値の形式で返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	-2
Tag_Result1	2
Tag_Value1	4
Tag_Value2	-1

Tag_Result2	4

MIN: 最小値の取得

説明

「最小値の取得」命令は、使用可能な入力の値を比較し、最小値を結果として返します。

この命令では、最小で 2、最大で 32 の入力を指定することが可能です。

次のいずれかの条件が満たされる場合、結果は無効です。

- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

構文

以下の構文が「最小値の取得」命令に使用されます。

```
SCL
MIN(IN1:= <Operand>,
    IN2 := <Operand>)
```

パラメータ

次の表に、「最小値の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	最初の入力値
IN2	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	2 番目の入力値
INn	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、	I、Q、M、D、L、P	値が比較される追加挿入された入力

			LTOD、DATE、LDT、DT、DTL		
ファンクション値		整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	命令の結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := MIN(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	12222
IN2	Tag_Value2	14444
IN3	Tag_Value3	13333
ファンクション値	Tag_Result	12222

この命令は、使用可能な入力の値を比較し、最小値(「Tag_Value1」)をオペランド「Tag_Result」にコピーします。

MAX: 最大値の取得

説明

「最大値の取得」命令は、使用可能な入力の値を比較し、最大値を結果として返します。

この命令では、最小で 2、最大で 32 の入力を指定することが可能です。

次のいずれかの条件が満たされる場合、結果は無効です。

- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

構文

以下の構文が「最大値の取得」命令に使用されます。

```
SCL
MAX(IN1:= <Operand>,
    IN2 := <Operand>)
```

パラメータ

次の表に、「最大値の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	最初の入力値
IN2	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	2 番目の入力値
INn	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、	I、Q、M、D、L、P	値が比較される追加挿入された入力

			DATE、LDT、DT、DTL		
ファンクション値		整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	命令の結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := MAX(IN1 := "Tag_Value1",
                    IN2 := "Tag_Value2",
                    IN3 := "Tag_Value3");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	Tag_Value1	12 222
IN2	Tag_Value2	14 444
IN3	Tag_Value3	13 333
ファンクション値	Tag_Result	14 444

この命令は、指定されたオペランドの値を比較し、最大値(「Tag_Value2」)をオペランド「Tag_Result」にコピーします。

LIMIT: 制限値の設定

説明

「制限値の設定」命令は、IN パラメータの値を MN および MX パラメータの値に制限します。 MN パラメータの値は、MX パラメータの値を超えてはなりません。

IN パラメータの値が MN 条件<=IN <= MX を満足する場合、この命令の結果として返されます。 この条件が満たされず、かつ IN 入力値が MN 下限値未満の場合、MN パラメータの値が結果として返されます。 上限値 MX を超過する場合、MX パラメータの値が結果として返されます。

MN 入力の値が MX 入力の値よりも大きい場合、結果が未定義なります。

この命令は、すべてのパラメータのオペランドのデータタイプが同じ場合にのみ実行されます。

構文

以下の構文が「制限値の設定」命令に使用されます。

```
SCL
LIMIT(MN  := <Operand>,
      IN  := <Operand>,
      MX  := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
MN	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	下限値
IN	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	入力値
MX	Input	整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、	I、Q、M、D、L、P	上限値

			LTOD、DATE、LDT、DT、DTL		
ファンクション値		整数、浮動小数点数、TIME、TOD、DATE、DTL	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT、DT、DTL	I、Q、M、D、L、P	命令の結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := LIMIT(MN := "Tag_Minimum",
                      IN := "Tag_Value",
                      MX := "Tag_Maximum");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MN	Tag_Minimum	12 000
IN	Tag_Value	8 000
MX	Tag_Maximum	16 000
ファンクション値	Tag_Result	12 000

オペランド「Tag_Value」の値が、オペランド「Tag_Minimum」および「Tag_Maximum」の値と比較されます。 オペランド「Tag_Value」の値が下限値未満であるため、オペランド「Tag_Minimum」の値がオペランド「Tag_Result」にコピーされます。

SQR: 2 乗値の形成

説明

「2 乗値の形成」命令を使用して、入力値を 2 乗し、結果を指定したオペランド内に保存します。

構文

「2 乗値の形成」命令には、次の構文を使用します。

```
SCL
SQR(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	入力値の 2 乗値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := SQR("Tag_Value");
"Tag_Result2" := SQR((SQR("Tag_Value1"))*"Tag_Val-
ue2");
```

入力値の 2 乗値は、ファンクション値としてオペランド「Tag_Resultxy」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	2.5
Tag_Result1	6.25
Tag_Value1	6.0
Tag_Value2	2.0
Tag_Result2	5184.0

SQRT: 平方根の形成

説明

「平方根の形成」命令を使用して、入力値の平方根を計算し、結果を指定したオペランド内に保存します。 この命令は、入力値がゼロよりも大きい場合に正の結果になります。 入力値がゼロ未満の場合、この命令は無効な浮動小数点数を返します。 入力値が「0」の場合、結果も「0」になります。

構文

「平方根の形成」命令には、以下の構文を使用します。

```
SCL
SQRT(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	入力値の平方根

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := SQRT("Tag_Value");
"Tag_Result2" := SQRT((SQR("Tag_Value1"))+"Tag_Val-
ue2");
```

入力値の平方根は、ファンクション値としてオペランド「Tag_Resultxy」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	4.0
Tag_Result1	2.0
Tag_Value1	3.0
Tag_Value2	16.0
Tag_Result2	5.0

LN: 自然対数の形成

説明

「自然対数の形成」命令を使用して、入力値の底 e (e=2.718282)に対する自然対数を計算できます。この命令は、入力値がゼロよりも大きい場合に正の結果になります。 入力値がゼロ未満の場合、この命令は無効な浮動小数点数を返します。

構文

「自然対数の形成」命令には、以下の構文を使用します。

```
SCL
LN(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expres-sion>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	入力値の自然対数

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := LN("Tag_Value");
"Tag_Result2" := LN("Tag_Value1"+"Tag_Value2");
```

命令の結果は、ファンクション値としてオペランド「Tag_Resultxy」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	2.5
Tag_Result1	0.916
Tag_Value1	1.5
Tag_Value2	3.2
Tag_Result2	1.548

EXP: 指数値の形成

説明

「指数値の形成」命令は、底 e (e = 2,718282)および入力値からの指数を計算し、指定されたオペランドに結果を保存します。

構文

「指数値の形成」命令には、以下の構文を使用します。

```
SCL
EXP(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expres-sion>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	入力値の指数値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := EXP("Tag_Value");
"Tag_Result2" := EXP("Tag_Value1"/"Tag_Value2");
```

命令の結果は、ファンクション値としてオペランド「Tag_Resultxy」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	20.5
Tag_Result1	799 902 200
Tag_Value1	15.5
Tag_Value2	30.2
Tag_Result2	1.671

SIN: 正弦値の形成

説明

入力値の正弦を計算するには、「正弦値の形成」命令を使用します。 入力値はラジアン単位で指定する必要があります。

構文

「正弦値の形成」命令には、次の構文を使用します。

```
SCL
SIN(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値(ラジアン単位の角度のサイズ)
ファンクション値		浮動小数点数	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := SIN("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	+1.570796 (π/2)
Tag_Result	1.0

COS: 余弦値の形成

説明

入力値の余弦を計算するには、「余弦値の形成」命令を使用します。 入力値はラジアン単位で指定する必要があります。

構文

「余弦値の形成」命令には、次の構文を使用します。

```
SCL
COS(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値(ラジアン単位の角度のサイズ)
ファンクション値		浮動小数点数	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := COS("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	+1.570796 (π/2)
Tag_Result	0

TAN: 正接値の形成

説明

入力値の正弦を計算するには、「正接値の形成」命令を使用します。 入力値はラジアン単位で指定する必要があります。

構文

「正接値の形成」命令には、次の構文を使用します。

```
SCL
TAN(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値(ラジアン単位の角度のサイズ)
ファンクション値		浮動小数点数	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := TAN("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	+3.141593 (π)
Tag_Result	0

ASIN: 逆正弦値の形成

説明

「逆正弦値の形成」命令を使用すると、この値に対応する正弦値から角度のサイズを計算できます。範囲-1～+1 の有効な浮動小数点数のみを入力値として指定できます。 計算された角度サイズはラジアン単位で示され、-π/2～+π/2 の値の範囲にすることができます。

構文

「逆正弦値の形成」命令には、次の構文を使用します。

```
SCL
ASIN(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	正弦値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ASIN("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	1.0
Tag_Result	+1.570796 (π/2)

ACOS: 逆余弦値の形成

説明

「逆余弦値の形成」命令を使用すると、この値に対応する余弦値から角度のサイズを計算できます。範囲-1～+1 の有効な浮動小数点数のみを入力値として指定できます。 計算された角度サイズはラジアン単位で示され、0～+π の値の範囲にすることができます。

構文

「逆余弦値の形成」命令には、次の構文を使用します。

```
SCL
ACOS(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	余弦値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ACOS("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	0
Tag_Result	+1.570796 (π/2)

ATAN: 逆正接値の形成

説明

「逆正接値の形成」命令を使用すると、この値に対応する正接値から角度のサイズを計算できます。入力値には、有効な浮動小数点数(または-NaN/+NaN)のみを指定することができます。 計算された角度サイズはラジアン単位で示され、-π/2～+π/2 の値の範囲にすることができます。

構文

「逆正接値の形成」命令には、以下の構文を使用します。

```
SCL
ATAN(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	正接値
ファンクション値		浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ATAN("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	1.0
Tag_Result	+0.785398 (π/4)

FRAC: 小数部を返す

説明

「小数部を返す」命令は、値の小数部を返します。 たとえば入力値が 123.4567 の場合は、値 0.4567 を返します。

構文

「小数部を返す」命令には、以下の構文を使用します。

SCL

```
FRAC(<Expression>)
FRAC_<Data type>(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
_<Data type>		浮動小数点数 既定値: REAL	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		浮動小数点数	I、Q、M、D、L、P	入力値の小数位

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := FRAC("Tag_Value");
"Tag_Result2" := FRAC_LREAL("Tag_Value");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Value	2.555	-1.4421
Tag_Result1	0.555	-0.4421

ムーブ操作

この章には下記に関する情報が記載されています：

Deserialize: 逆シリアル化

説明

「逆シリアル化」命令を使用して、PLC データタイプ(UDT)のシーケンシャル表示を PLC データタイプに変換して戻し、その内容全体を表示することができます。

PLC データタイプのシーケンシャル表示を配置するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。変換の前に、十分なメモリ領域が存在することを確認してください。

この命令では、変換された PLC データタイプの複数のシーケンシャル表示をそれらの元の状態に、順を追って変換し直すことができます。

PLC データタイプ(UDT)の単一のシーケンシャル表示のみを変換して元に戻す場合は、命令「TRCV: 通信接続経由のデータ受信」を直接に使用することもできます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

構文

「逆シリアル化」命令には、以下の構文を使用します。

```
SCL
Deserialize(SRC_ARRAY := <Operand>,
            DEST_VARIABLE:= <Operand>,
            POS := <Operand>)
```

パラメータ

次の表に、「逆シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_ARRAY	Input	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるグローバルデータブロック
DEST_VARIABLE	InOut	VARIANT	I、Q、M、L	返された変換済み PLC データタイプ(UDT)が格納されるタグ
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
ファンクション値		INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_ARRAY パラメータと DEST_VARIABLE パラメータのためのメモリ領域がオーバーラップしています。
8136	DEST_VARIABLE パラメータのデータブロックが標準アクセスによるブロックではありません。
8150	SRC_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8151	SRC_ARRAY パラメータでコード生成エラーが発生しました
8153	SRC_ARRAY パラメータのメモリの空き領域が不足しています。
8250	DEST_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8251	DEST_VARIABLE パラメータでコード生成エラーが発生しました
8254	DEST_VARIABLE パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buf-
fer".Field,
                           DEST_VARIABLE := "Tar-
get".Client,
                           POS := #BufferPos);

#Tag_RetVal := Deserialize(SRC_ARRAY := "Buf-
fer".Field,
                           DEST_VARIABLE := #Label,
                           POS := #BufferPos);

IF #Label = 'arti' THEN
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buf-
fer".Field,
                           DEST_VARIABLE := "Tar-
get".Article[#DeliverPos],
                           POS := #BufferPos);
ELSIF #Label = 'Bill' THEN
```

```
#Tag_RetVal := Deserialize(SRC_ARRAY := "Buf-
fer".Field,
                           DEST_VARIABLE := "Tar-
get".Bill[#DeliverPos],
                           POS := #BufferPos);
;
ELSE
;
END_IF;
```

「逆シリアル化」命令は、Buffer」データブロックのカスタマデータのシーケンシャル表示を逆シリアル化し、「Target」データブロックに書き込みます。 変換済みカスタマデータで占有されるバイト数は、#BufferPos オペランドに格納されます。

「逆シリアル化」命令は、「Buffer」データブロックの、カスタマデータの後にシーケンシャル表示で保存されたセパレータシートのシーケンシャル表示を逆シリアル化し、これらの文字を#Label オペランドに書き込みます。 これらの文字は、「arti」および「Bill」用の比較命令を使用して比較されます。「arti」 = TRUE の場合の比較では、データは、逆シリアル化され、「Target」データブロックに書き込まれたアーティクルデータです。 「Bill」 = TRUE の場合の比較では、データは、逆シリアル化され、「Target」データブロックに書き込まれたビリングデータです。

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言をリストします。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Target	Client	「クライアント」(PLC データタイプ)

	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
	Bill	INT の配列[0..10]
Buffer	Field	バイトの配列[0..294]

Serialize: シリアル化

説明

「シリアル化」命令を使用して、データタイプの構造体の部分を失うことなく、複数の PLC データタイプ(UDT)を 1 つのシーケンシャル表示に変換することができます。

この命令を使用して、ユーザプログラムの複数の構造体データ項目を一時的にバッファ(グローバルデータブロックを推奨)に保存し、別の CPU に送信することができます。 変換済み PLC データタイプを格納するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。 標準メモリ領域の容量は 64 KB です。 変換の前に、十分なメモリ領域が存在することを確認してください。

POS パラメータのオペランドには、変換済み PLC データタイプによって使用されるバイト数に関する情報が収納されています。

単一の PLC データタイプ(UDT)を送信する場合、直接に命令「TSEND: 通信接続経由のデータ送信」を呼び出すことができます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

構文

「シリアル化」命令には、以下の構文を使用します。

```
SCL
Serialize(SRC_VARIABLE := <Operand>,
          DEST_ARRAY := <Operand>,
          POS := <Operand>)
```

パラメータ

次の表に、「シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_VARIABLE	Input	VARIANT	I、Q、M、L	シーケンシャル表示に変換される PLC データタイプ (UDT)。
DEST_ARRAY	InOut	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるデータブロック。
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。

ファンクション値	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_VARIABLE パラメータと DEST_ARRAY パラメータのためのメモリ領域がオーバーラップしています。
8150	SRC_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8152	SRC_VARIABLE パラメータでコード生成エラーが発生しました
8236	DEST_ARRAY パラメータのデータブロックが標準アクセスによるブロックではありません。
8250	DEST_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8252	DEST_ARRAY パラメータでコード生成エラーが発生しました
8253	DEST_ARRAY パラメータのメモリの空き領域が不足しています。
8254	DEST_ARRAY パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Cli-
ent,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Label := STRING_TO_WSTRING('arti');

#Tag_RetVal := Serialize(SRC_VARIABLE := #Label,
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);

#Tag_RetVal := Serialize(SRC_VARIABLE := "Source".Ar-
ticle[#DeliverPos],
                         DEST_ARRAY := "Buffer".Field,
                         POS := #BufferPos);
```

「シリアル化」命令は、「Source」データブロックのカスタマデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックに書き込みます。 シーケンシャル表示によって占有されたバイト数は、#BufferPos オペランドに格納されます。

後でのシーケンシャル表示の逆シリアル化を容易にするために、その時点では、一種のセパレータシートが挿入されます。 「文字列のムーブ」命令は、「arti」文字を「#Label」オペランドに移動します。 「シリアル化」命令は、これらの文字を「Buffer」データブロックのカスタマデータの後に書き込みます。 文字が必要とするバイト数が、#BufferPos オペランドに格納済みの数に加算されます。

「シリアル化」命令は、「Source」データブロックの、ランタイム時に計算される特定のアーティクルのデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックの「arti」文字の後に書き込みます。

次の表に、オペランドの宣言を示します。

オペランド	データタイプ	宣言
DeliverPos	INT	ブロックインターフェースの「入力」セクションで宣言
BufferPos	DINT	ブロックインターフェースの「一時」セクションで宣言
Error	INT	ブロックインターフェースの「一時」セクションで宣言
Label	STRING[4]	ブロックインターフェースの「一時」セクションで宣言

次の表に、PLC データタイプの宣言をリストします。

PLC データタイプの名前	名前	データタイプ
Article	Number	DINT
	Declaration	STRING
	Colli	INT
Client	Title	INT
	First name	STRING[10]
	Surname	STRING[10]

次の表に、データブロックの宣言を示します。

データブロックの名前	名前	データタイプ
Source	Client	「クライアント」(PLC データタイプ)
	Article	「アーティクル」(PLC データタイプ)の配列[0..10]
Buffer	Field	バイトの配列[0..294]

MOVE_BLK: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 宛先領域にコピーするエレメントの数は、COUNT パラメータで指定できます。 移動されるエレメントの幅は、ソース領域内の最初のエレメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

OUT 出力の値は、次の条件が満たされると無効です。

- 移動されるデータが、IN パラメータまたは OUT パラメータで使用可能なデータよりも多いこと。

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
SCL
MOVE_BLK(IN := <Operand>,
         COUNT := <Operand>,
         OUT => <Operand>)
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P	ソース領域から宛先領域にコピーするエレメントの数。
OUT [1]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント
[1] 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
MOVE_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。 これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。

MOVE_BLK_VARIANT: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 配列全体または配列の1エレメントを同じデータタイプの別の配列にコピーすることができます。 ソース配列と宛先配列のサイズ(エレメントの数)は異なる場合があります。配列内の複数エレメントまたは単一エレメントをコピーすることができます。

ブロックの作成時にこの命令を使用する場合、VARIANT ごとにソースおよび宛先が転送されるため、配列はまだ既知である必要はありません。

SRC_INDEX および DEST_INDEX パラメータでのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

コピーされるデータが使用可能なデータよりも多い場合、この命令は実行されません。

注記

BOOL データタイプと接続している VARIANT

VARIANT データタイプのパラメータ(ソース領域または宛先領域)と BOOL データタイプのタグまたは ARRAY of BOOL を相互接続する場合、以下のオプションがあります。

1. シンボリックにアドレス指定することができます。

   例: SRC パラメータ: "Data_block".myArray
2. 任意のポインタによってアドレス指定することができます。ただし、領域で指定された長さは 8 で割り切れるようにする必要があります。割り切れない場合、この命令は実行されません。

   例: SRC パラメータ: P#DB123.DBX456.0 BOOL 1000

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
SCL
MOVE_BLK_VARIANT(SRC := <Operand>,
                 COUNT := <Operand>,
                 SRC_INDEX := <Operand>,
                 DEST_INDEX := <Operand>,
                 DEST => <Operand>)
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	VARIANT (配列または個々の配列エレメントをポイン	I、Q、M、L	コピー元のソースブロック

		トする)、<Data_type>の配列		
COUNT	Input	UDINT	I、Q、M、D、L	コピーされるエレメント数 SRC が DEST パラメータまたは ARRAY パラメータで指定されていない場合は、COUNT パラメータに値「1」を割り当てます。
SRC_INDEX	Input	DINT	I、Q、M、D、L	• SRC_INDEX パラメータは、ゼロベースで計算されます。 SRC がパラメータ ARRAY で指定されると、パラメータ SRC_INDEX の整数はコピー元のソース領域内の最初のエレメントを指定します。 宣言された配列の限界値には関係ありません。 • SRC がパラメータ ARRAY で指定されないか、または配列の 1 つの単一エレメントのみが指定された場合、パラメータ SRC_INDEX に値「0」を割り当てます。
DEST_INDEX	Input	DINT	I、Q、M、D、L	• DEST_INDEX パラメータは、ゼロベースで計算されます。 DEST がパラメータ ARRAY で指定されると、パラメータ DEST_INDEX の整数はコピー先の宛先領域内の最初のエレメントを指定します。 宣言された配列の限界値には関係ありません。 • DEST が ARRAY パラメータで指定されていない場合は、DEST_INDEX パラメータに値「0」を割り当てます。
DEST	Output [1)]	VARIANT	I、Q、M、L	ソースブロックの内容がコピーされる宛先領域。
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	エラー情報
1) データがタグに流れるため、DEST パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	データタイプが一致していません
8151	SRC パラメータにアクセスできません。
8152	SRC パラメータのオペランドが入力されていません。
8153	SRC パラメータでコード生成エラーが発生しました
8154	SRC パラメータのオペランドのデータタイプが BOOL です。
8281	COUNT パラメータの値が無効です
8382	SRC_INDEX パラメータの値が VARIANT の限界値外です。
8383	SRC_INDEX パラメータの値が配列の上限値外です
8482	DEST_INDEX パラメータの値が VARIANT の限界値外です。
8483	DEST_INDEX パラメータの値が配列の上限値外です
8534	DEST パラメータが書き込み保護されています
8551	DEST パラメータにアクセスできません。
8552	DEST パラメータのオペランドが入力されていません。
8553	DEST パラメータでコード生成エラーが発生しました
8554	DEST パラメータのオペランドのデータタイプが BOOL です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := MOVE_BLK_VARIANT(SRC := #SrcField,
                                 COUNT := "Tag_Count",
                                 SRC_INDEX :=
"Tag_Src_Index",
                                 DEST_INDEX :=
"Tag_Dest_Index",
                                 DEST => #DestField);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
SRC	Input	#SrcField	ローカルオペランド #SrcField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。

			("MOVE_UDT"の ARRAY[0..10]
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	ローカルオペランド #DestField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。 ("MOVE_UDT"の ARRAY[10..20]

UDT の配列の 4 番目のエレメントから始めて、2 つのエレメントが、ソース領域から宛先領域に移動されます。 コピーが、UDT の配列の 4 番目のエレメントから始めて、挿入されます。

UMOVE_BLK: 割り込みなしブロック転送

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 この命令に割り込みをかけることはできません。 宛先領域にコピーする工レメントの数は、COUNT パラメータで指定できます。 移動される工レメントの幅は、ソース領域内の最初の工レメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

OUT 出力の値は、次の条件が満たされると無効です。

- 移動されるデータが、IN パラメータまたは OUT パラメータで使用可能なデータよりも多いこと。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

構文

以下の構文が「割り込みなしブロック転送」命令に使用されます。

```
SCL
UMOVE_BLK(IN  := <Operand>,
          COUNT := <Operand>,
          OUT => <Operand>)
```

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	コピーされるソース領域の最初の工レメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P	ソース領域から宛先領域にコピーする工レメントの数。

OUT [1)]	Output	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	ソース領域の内容がコピーされる宛先領域の最初の工レメント
[1)] 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
UMOVE_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。 これは、INT データタイプの 6 つの工レメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つの工レメントで構成されます。

この命令は、3 番目の工レメントから開始して、#a_array タグから 3 つの INT 工レメントを選択し、2 番目の工レメントで開始して、その内容を#b_array 出カタグにコピーします。 他のオペレーティングシステムのアクティビティによって、コピー操作に割り込みをかけることはできません。

FILL_BLK: フィル命令

説明

「フィル」命令は、メモリ領域(ソース領域)の内容を選択したメモリ領域(宛先領域)にコピーするために使用します。 コピー操作の繰り返し回数は、COUNT パラメータの値で指定されます。 命令が実行されると、ソース領域が選択され、COUNT パラメータの値で指定された回数で宛先領域に移動されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

OUT 出力の値は、次の条件が満たされると無効です。

- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

構文

「フィル」命令には、以下の構文を使用します。

```
SCL
FILL_BLK(IN  := <Operand>,
         COUNT  := <Operand>,
         OUT  => <Operand>)
```

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、DATE、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	I、Q、M、D、L、P	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P	コピー操作の繰り返し回数
OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、DATE、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	フィル命令が開始する宛先領域のアドレス

[1] 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。

2) 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
FILL_BLK(IN := #a_array[2],
         COUNT := "Tag_Count",
         OUT => #b_array[1]);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。 これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。 これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。

UFILL_BLK: 割り込みなしフィル命令

説明

「割り込み不可能なフィル」命令は、メモリ領域(ソース領域)の内容を選択したメモリ領域(宛先領域)にコピーするために使用します。 コピー操作の繰り返し回数は、COUNT パラメータの値で指定されます。 命令が実行されると、IN 入力の値が選択され、COUNT パラメータの値で指定された回数で宛先領域にコピーされます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。 このため、「割り込み不可能なフィル」命令の実行中には CPU の割り込み応答時間が長くなります。

OUT 出力の値は、次の条件が満たされると無効です。

- 入力 IN または出力 OUT で使用可能のデータよりも多くのデータが移動された。

「割り込み不可能なフィル」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

構文

「割り込み不可能なフィル」命令には、以下の構文を使用します。

```
SCL
UFILL_BLK(IN := <Operand>,
          COUNT := <Operand>,
          OUT => <Operand>)
```

パラメータ

次の表に、「割り込み不可能なフィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	I、Q、M、D、L、P	宛先領域を埋めるために使用するエレメント
COUNT	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P	コピー操作の繰り返し回数

OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、DATE、CHAR、WCHAR	2 進数、整数、浮動小数点数、タイマ、DATE、CHAR、WCHAR、TOD、LTOD	D、L	フィル命令が開始する宛先領域のアドレス
[1] 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。 [2] 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。					

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
UFILL_BLK(IN := #a_array[2],
          COUNT := "Tag_Count",
          OUT => #b_array[1]);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つの工レメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つの工レメントで構成されます。

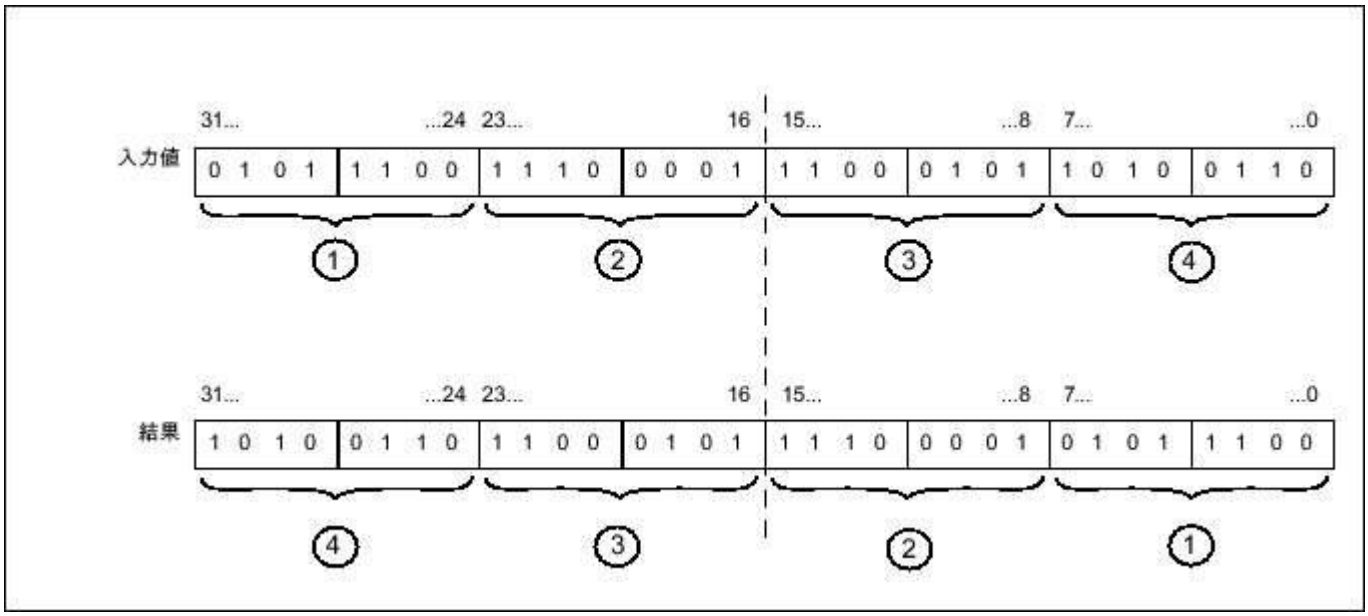
この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出カタグに 3 回コピーします。 他のオペレーティングシステムのアクティビティによって、コピー操作に割り込みをかけることはできません。

SWAP: スワップ

説明

「スワップ」命令を使用して、入力値のバイトの配置を変更し、結果を指定したオペランド内に保存することができます。

次の図に、DWORD データタイプのオペランドのバイトが、「スワップ」命令を使用してスワップされる様子を示します。



構文

「スワップ」命令には、以下の構文を使用します。

```
SCL
SWAP(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
<Expression>	Input	WORD, DWORD	WORD, DWORD, LWORD	I、Q、M、D、L、P	入力値
ファンクション値		WORD, DWORD	WORD, DWORD, LWORD	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := SWAP("Tag_Value");
```

命令の結果は、ファンクション値として返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_Value	0000 1111 0101 0101
Tag_Result	0101 0101 0000 1111

配列 DB

この章には下記に関する情報が記載されています：

- ReadFromArrayDB: 配列データブロックからの読み出し (S7-1500)
- WriteToArrayDB: 配列データブロックへの書き込み (S7-1500)
- ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し (S7-1500)
- WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み (S7-1500)

ReadFromArrayDB: 配列データブロックからの読み出し

説明

「配列データブロックからの読み出し」命令を使用し、配列データブロックからデータを読み出して、宛先領域に書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

構文

「ARRAY データブロックからの読み出し」命令には、以下の構文を使用します。

SCL

```
ReadFromArrayDB(DB := <Operand>,
                INDEX := <Operand>,
                VALUE => <Operand>)
```

パラメータ

次の表に、「配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P	読み出されるエレメント
VALUE	Output 1)	VARIANT	I、Q、M、L	読み出されて出力される値
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	命令の結果
1) データがタグに流れるため、VALUE パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、書き込み保護されているか、またはロードメモリ内に存在します。
8135	配列データブロックに無効な値が含まれます。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8450	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8452	コード生成エラー
8453	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"TagResult" := ReadFromArrayDB(DB := "ArrayDB",
                               INDEX := "Ar-
rayDB"."THIS"[2],
                               VALUE => "Target-
Field[10]".Data2[1]);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB."THIS"[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"TargetField[10]".Data2[1]	「TargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[10 to 20]です。

3 番目のエレメントは「ArrayDB」から読み出され、"TargetField[10]".Data2[1]オペランドに書き込まれます。

WriteToArrayDB: 配列データブロックへの書き込み

説明

「配列データブロックへの書き込み」命令を使用して、配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の工レメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

構文

「配列データブロックへの書き込み」命令には、以下の構文を使用します。

```
SCL
WriteToArrayDB(DB := <Operand>,
               INDEX := <Operand>,
               VALUE := <Operand>)
```

パラメータ

次の表に、「配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P	データが書き込まれる DB 内の工レメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存された工レメントのデータタイプが、VARIANT で転送される工レメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。

8134	データブロックが書き込み保護されています。
8135	データブロックが配列データブロックではありません。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8350	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8352	コード生成エラー
8353	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"TagResult" := WriteToArrayDB(DB := "ArrayDB",
                              INDEX := "Ar-
rayDB"."THIS"[2],
                              VALUE := "Source-
Field[1]".Data1[6]);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB."THIS"[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceField[1]".Data1[6]	「SourceField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。

「SourceField」オペランドから、2 番目のエレメントの「Data1[6]」エレメントが「ArrayDB」に書き込まれます。 3 番目のエレメントは、「ArrayDB」に書き込まれます。

ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し

説明

「ロードメモリの配列データブロックからの読み出し」命令を使用して、ロードメモリの配列データブロックからデータを読み出します。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 この命令は、BUSY パラメータで信号立ち下がりエッジが検出された場合に終了します。 1 プログラムサイクルの間、DONE パラメータのシグナル状態が「1」になり、このサイクル内に、読み取られた値が VALUE パラメータに出力されます。 他のすべてのプログラムサイクルでは、VALUE パラメータの値は変更されません。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

構文

「ロードメモリの ARRAY データブロックからの読み出し」命令には、以下の構文を使用します。

SCL
```
"インスタンス DB"(REQ := <Operand>,
                  DB := <Operand>,
                  INDEX := <Operand>,
                  VALUE := <Operand>,
                  BUSY => <Operand>,
                  DONE => <Operand>,
                  ERROR => <Operand>)
```

パラメータ

次の表に、「ロードメモリの配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	REQ =「1」: 配列 DB の読み出しから開始

DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P	読み出されるエレメント
VALUE	InOut	VARIANT	I、Q、M、L	読み出されて出力される値 TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ読み出し中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL: ロードメモリのデータブロックからの読み出し」および「WRIT_DBL: データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"ReadFromArrayDBL_DB"(REQ := "TagReg",
                      DB := "ArrayDB",
                      INDEX := "ArrayDB"."THIS"[2],
                      VALUE := "SourceTarget-
Field[1]".Data1[6],
                      BUSY => "TagBusy",
                      DONE => "TagDone",
                      ERROR => "TagError");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB."THIS"[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceTargetField[1]".Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagReq」オペランドで立ち上がりエッジが検出されると、「ロードメモリの ARRAY データブロックからの読み出し」命令が実行されます。 3 番目のエレメントは「ArrayDB」から読み出され、"SourceTargetField[1]".Data1[6]オペランドに出力されます。 「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値は変更されません。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み

説明

「ロードメモリの配列データブロックへの書き込み」命令を使用して、ロードメモリの配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 BUSY パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、VALUE パラメータの値がデータブロックに書き込まれます。 DONE1 パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、パラメータの値がデータブロックに書き込まれます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

構文

「ロードメモリの ARRAY データブロックへの書き込み」命令には、以下の構文を使用します。

SCL
```
"インスタンス DB"(REQ := <Operand>,
                  DB := <Operand>,
                  INDEX := <Operand>,
                  VALUE := <Operand>,
                  BUSY => <Operand>,
                  DONE => <Operand>,
                  ERROR => <Operand>)
```

パラメータ

次の表に、「ロードメモリの配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	REQ =「1」: 配列 DB への書き込みを開始

DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P	データが書き込まれる DB 内のエレメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値 TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ書き込み中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8234	データブロックが書き込み保護されています。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL: ロードメモリのデータブロックからの読み出し」および「WRIT_DBL: データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"WriteToArrayDBL_DB"(REQ := "TagReg",
                     DB := "ArrayDB",
                     INDEX := "ArrayDB"."THIS"[2],
                     VALUE := "SourceTarget-
Field[1]".Data1[6],
                     BUSY => "TagBusy",
                     DONE => "TagDone",
                     ERROR => "TagError");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	ArrayDB."THIS"[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceTargetField[1]".Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagReq」オペランドで立ち上がりエッジが検出されると、「ロードメモリの ARRAY データブロックへの書き込み」命令が実行されます。「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値が「ArrayDB」の 3 番目のエレメントに書き込まれます。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

読み取り/書き込みアクセス

この章には下記に関する情報が記載されています：

PEEK: メモリアドレスの読み取り

説明

「メモリアドレスの読み取り」命令は、メモリ領域からのデータタイプを指定しないメモリアドレスの読み取りに使用します。

AREA パラメータのデータブロックで 16#84 領域を指定した場合、「標準」ブロックプロパティを使用してのみデータブロックにアクセスできます。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリアドレスの読み取り」命令には、以下の構文を使用します。

```
SCL
PEEK(AREA:= <Operand>,
     DBNUMBER := <Operand>,
     BYTEOFFSET:= <Operand>)
```

```
SCL
PEEK_<Data type>(AREA := <Operand>,
                 DBNUMBER := <Operand>,
                 BYTEOFFSET:= <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#1: 周辺機器入力(S7-1500 のみ)
DBNUM-BER	Input	DINT, DB_ANY	D	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	読み取り元のアドレス 16 個の最下位ビットのみが使用されます。
_<Data type>		ビット列	-	ファンクション値のデータタイプ:

	既定値: BYTE		1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値	ビット列	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

注記
入力、出力またはビットメモリ領域からメモリアドレスを読み取る場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := PEEK(AREA := "Tag_Area",
                      DBNUMBER := "Tag_DBNumber",
                      BYTEOFFSET := "Tag_Byte");
```

```
SCL
"Tag_Result2" := PEEK_WORD(AREA := "Tag_Area",
                           DBNUMBER := "Tag_DBNumber",
                           BYTEOFFSET := "Tag_Byte");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
ファンクション値	Tag_Result1	バイト値「20」
ファンクション値	Tag_Result2	ワード値「20」

この命令は、データブロック「5」のオペランド「Tag_Byte」からアドレス「20」の値を読み取り、結果をオペランド「Tag_Result」にファンクション値として返します。

PEEK_BOOL: メモリビットの読み取り

説明

「メモリビットの読み取り」命令は、メモリビットからのデータタイプを指定しないメモリアドレスの読み取りに使用します。

AREA パラメータのデータブロックで 16#84 領域を指定した場合、「標準」ブロックプロパティを使用してのみデータブロックにアクセスできます。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリビットの読み取り」命令には、以下の構文を使用します。

```
SCL
PEEK_BOOL(AREA := <Operand>,
          DBNUMBER := <Operand>,
          BYTEOFFSET := <Operand>,
          BITOFFSET := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#1: 周辺機器入力(S7-1500 のみ)
DBNUM-BER	Input	DINT, DB_ANY	D	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	読み取り元のアドレス 16 個の最下位ビットのみが使用されます。
BITOFFSET	Input	INT	I、Q、M、D、L	読み取り元のビット
ファンクション値		BOOL	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

注記
入力、出力またはビットメモリ領域からメモリビットを読み取る場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := PEEK_BOOL(AREA := "Tag_Area",
                          DBNUMBER := "Tag_DBNumber",
                          BYTEOFFSET := "Tag_Byte",
                          BITOFFSET := "Tag_Bit");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
ファンクション値	Tag_Result	3

この命令は、データブロック「5」のバイト「20」でオペランド「Tag_Bit」からメモリビット「3」の値を読み取り、結果をファンクション値としてオペランド「Tag_Result」に返します。

POKE: メモリアドレスの書き込み

説明

「メモリアドレスの書き込み」命令は、メモリ領域へのデータタイプを指定しないメモリアドレスの書き込みに使用します。

AREA パラメータのデータブロックで 16#84 領域を指定した場合、「標準」ブロックプロパティを使用してのみデータブロックにアクセスできます。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリアドレスの書き込み」命令には、以下の構文を使用します。

```
SCL
POKE(AREA  := <Operand>,
     DBNUMBER  := <Operand>,
     BYTEOFFSET  := <Operand>,
     VALUE  := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#2: 周辺機器出力(S7-1500 のみ)
DBNUM-BER	Input	DINT, DB_ANY	D	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	書き込まれるアドレス 16 個の最下位ビットのみが使用されます。
VALUE	Input	ビット列	I、Q、M、D、L	書き込まれる値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

注記
入力、出力またはビットメモリ領域にメモリアドレスを書き込む場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
POKE(AREA := "Tag_Area",
     DBNUMBER := "Tag_DBNumber",
     BYTEOFFSET := "Tag_Byte"),
     VALUE := "Tag_Value";
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
VALUE	Tag_Value	16#11

この命令は、データブロック「5」内のメモリ領域「20」を値「16#11」で上書きします。

POKE_BOOL: メモリビットの書き込み

説明

「メモリビットの書き込み」命令は、メモリビットへのデータタイプを指定しないメモリアドレスの書き込みに使用します。

AREA パラメータのデータブロックで 16#84 領域を指定した場合、「標準」ブロックプロパティを使用してのみデータブロックにアクセスできます。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリビットの書き込み」命令には、以下の構文を使用します。

```
SCL
POKE_BOOL(AREA := <Operand>,
          DBNUMBER := <Operand>,
          BYTEOFFSET := <Operand>,
          BITOFFSET := <Operand>,
          VALUE := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA	Input	BYTE	I、Q、M、D、L	以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB • 16#2: 周辺機器出力(S7-1500 のみ)
DBNUM-BER	Input	DINT, DB_ANY	D	AREA = DB ならばデータブロックの番号、それ以外は「0」
BYTEOFF-SET	Input	DINT	I、Q、M、D、L	書き込まれるアドレス 16 個の最下位ビットのみが使用されます。
BITOFFSET	Input	INT	I、Q、M、D、L	書き込まれるビット
VALUE	Input	BOOL	I、Q、M、D、L	書き込まれる値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

注記
入力、出力またはビットメモリ領域にメモリビットを書き込む場合、DBNUMBER パラメータに値「0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
POKE_BOOL(AREA := "Tag_Area",
          DBNUMBER := "Tag_DBNumber",
          BYTEOFFSET := "Tag_Byte",
          BITOFFSET := "Tag_Bit",
          VALUE := "Tag_Value");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA	Tag_Area	16#84
DBNUMBER	Tag_DBNumber	5
BYTEOFFSET	Tag_Byte	20
BITOFFSET	Tag_Bit	3
VALUE	Tag_Value	M0.0

この命令は、バイト「20」内のデータブロック「5」のメモリビット「3」を値「M0.0」で上書きします。

POKE_BLK: メモリ領域の書き込み

説明

「メモリ領域の書き込み」命令は、メモリ領域の内容をデータタイプを指定せずに他のメモリ領域にコピーします。

AREA パラメータのデータブロックで 16#84 領域を指定した場合、「標準」ブロックプロパティを使用してのみデータブロックにアクセスできます。

注記
この命令は、「標準」メモリ領域へのアクセスにのみ使用可能です。

構文

「メモリ領域の書き込み」命令には、以下の構文を使用します。

```
SCL
POKE_BLK(AREA_SRC:= <Operand>,
         DBNUMBER_SRC:= <Operand>,
         BYTEOFFSET_SRC:= <Operand>,
         AREA_DEST:= <Operand>,
         DBNUMBER_DEST:= <Operand>,
         BYTEOFFSET_DEST:= <Operand>,
         COUNT  := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
AREA_SRC	Input	BYTE	I、Q、M、D、L	ソースのメモリ領域で、以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB
DBNUMBER_SRC	Input	DINT, DB_ANY	D	AREA = DB ならばソースのメモリ領域内のデータブロックの番号、それ以外は「0」
BYTEOFF-SET_SRC	Input	DINT	I、Q、M、D、L	書き込まれるソースのメモリ領域のアドレス 16 個の最下位ビットのみが使用されます。

AREA_DEST	Input	BYTE	I、Q、M、D、L	宛先のメモリ領域で、以下の領域を選択できます。 • 16#81: 入力 • 16#82: 出力 • 16#83: ビットメモリ • 16#84: DB
DBNUM-BER_DEST	Input	DINT, DB_ANY	D	AREA = DB ならば宛先のメモリ領域内のデータブロックの番号、それ以外は「0」
BYTEOFF-SET_DEST	Input	DINT	I、Q、M、D、L	書き込まれる宛先のメモリ領域のアドレス 16 個の最下位ビットのみが使用されます。
COUNT	Input	DINT	I、Q、M、D、L	コピーされるバイト数

有効なデータタイプの追加情報については、「関連項目」を参照してください。

注記
入力、出力またはビットメモリ領域にメモリアドレスを書き込む場合、DBNUMBER パラメータに値「 0」を割り当てる必要があります。割り当てないと、命令が無効になります。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
POKE_BLK(AREA_SRC := "Tag_Source_Area",
         DBNUMBER_SRC := "Tag_Source_DBNumber",
         BYTEOFFSET_SRC := "Tag_Source_Byte"),
         AREA_DEST := "Tag_Destination_Area",
         DBNUMBER_DEST := "Tag_Destination_DBNumber",
         BYTEOFFSET_DEST := "Tag_Destination_Byte",
         COUNT := "Tag_Count");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
AREA_SRC	Tag_Source_Area	16#84
DBNUMBER_SRC	Tag_Source_DBNumber	5
BYTEOFFSET_SRC	Tag_Source_Byte	20
AREA_DEST	Tag_Destination_Area	16#83
DBNUMBER_DEST	Tag_Destination_DBNumber	0
BYTEOFFSET_DEST	Tag_Destination_Byte	30
COUNT	Tag_Count	100

この命令は、アドレス「30」から始まるビットメモリのメモリ領域内で、アドレス「20」から始まるデータブロック「5」から 100 バイトを書き込みます。

READ_LITTLE: リトルエンディアン形式のデータの読み出

説明

「リトルエンディアン形式のデータの読み出し命令を使用し、メモリ領域からデータを読み出して、これをリトルエンディアンバイトシーケンスで単一のタグに書き込みます。 リトルエンディアン形式では、最下位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_ARRAY および DEST_VARIABLE パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 DEST_VARIABLE パラメータの VARIANT は、基本データタイプをポイントする必要があります。 SRC_ARRAY パラメータの VARIANT が、データが読み出されるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、読み出しが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの読み出し

VARIANT が指すタグを読み出す場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを読み出す場合、「スライスアクセス」を使用します。

構文

「リトルエンディアン形式のデータの読み出し」命令には、以下の構文を使用します。

```
SCL
READ_LITTLE(SRC_ARRAY := <Operand>,
            DEST_VARIABLE => <Operand>,
            POS := <Operand>)
```

パラメータ

次の表に、「リトルエンディアン形式のデータの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
SRC_ARRAY	Input	バイトの配列	バイトの配列	I、Q、M、D、L	読み取り元のメモリ領域
DEST_VARIABLE	Output	BIT 文字列、整数、浮動小数点数、TOD、DATE、CHAR、WCHAR	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	読み出し値
POS	InOut	DINT	DINT	I、Q、M、D、L	読み出しが開始される位置を決定します。POS パラメータは、ゼ

					口ベースで計算されます。
ファンクション値 (RET_VAL)		INT	INT	I、Q、M、D、L	エラー情報

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#TagResult := READ_LITTLE(SRC_ARRAY := #SourceField,
                          DEST_VARIABLE => #DINTVaria-
ble,
                          POS := #TagPos);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_ARRAY	#SourceField	BYTE の ARRAY [0..3] := 16#1A、16#2B、16#3C、16#4D
DEST_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
POS	#TagPos	0 => 4

この命令は、#SourceField メモリ領域から整数 1_295_788_826 を読み出して、これを#DINTVariable オペランドに、リトルエンディアン形式で書き込みます。 DEST_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。 個数 4 が#TagPos オペランドに格納されます。

WRITE_LITTLE: リトルエンディアン形式のデータの書き込

説明

「リトルエンディアン形式のデータの書き込み」命令を使用して、リトルエンディアンバイトシーケンスの単一のタグのデータをメモリ領域に書き込みます。 リトルエンディアン形式では、最下位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_VARIABLE および DEST_ARRAY パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 SRC_VARIABLE パラメータの VARIANT は、基本データタイプをポイントする必要があります。 DEST_ARRAY パラメータの VARIANT が、データが書き込まれるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、書き込みが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの書き込み

VARIANT が指すタグを書き込む場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを書き込む場合、「スライスアクセス」を使用します。

構文

「リトルエンディアン形式のデータの書き込み」命令には、以下の構文を使用します。

```
SCL
WRITE_LITTLE(SRC_VARIABLE := <Operand>,
             DEST_ARRAY := <Operand>,
             POS := <Operand>)
```

パラメータ

次の表に、「リトルエンディアン形式のデータの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
SRC_VARIABLE	Input	BIT 文字列、整数、浮動小数点数、TOD、DATE、CHAR、WCHAR	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	書き込み元のタグ
DEST_ARRAY	InOut	バイトの配列	バイトの配列	I、Q、M、D、L	データが書き込まれるメモリ領域
POS	InOut	DINT	DINT	I、Q、M、D、L	書き込みが開始される位置を決定します。POS パラメータは、ゼ

					口ベースで計算されます。
ファンクション値 (RET_VAL)		INT	INT	I、Q、M、D、L	エラー情報

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#TagResult := WRITE_LITTLE(SRC_VARIABLE := #DINTVaria-
ble,
                           DEST_ARRAY := #TargetField,
                           POS := #TagPos);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_VARIABLE	#DINTVariable	1295788826 16#4D3C2B1A
DEST_ARRAY	#TargetField	BYTE の ARRAY [0..10] = 16#1A、16#2B、16#3C、16#4D
POS	#TagPos	0 => 4

この命令は、#DINTVariable オペランドから整数 1_295_788_826 を#TargetField メモリ領域に、リトルエンディアン形式で書き込みます。 SRC_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。 個数 4 が#TagPos オペランドに格納されます。

READ_BIG: ビッグエンディアン形式のデータの読み出し 

説明

「ビッグエンディアン形式のデータの読み出し命令を使用し、メモリ領域からデータを読み出して、これをビッグエンディアンバイトシーケンスで単一のタグに書き込みます。 ビッグエンディアン形式では、最上位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_ARRAY および DEST_VARIABLE パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 DEST_VARIABLE パラメータの VARIANT は、基本データタイプをポイントする必要があります。 SRC_ARRAY パラメータの VARIANT が、データが読み出されるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、読み出しが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの読み出し

VARIANT が指すタグを読み出す場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを読み出す場合、「スライスアクセス」を使用します。

構文

「ビッグエンディアン形式のデータの読み出し」命令には、以下の構文を使用します。

```
SCL
READ_BIG(SRC_ARRAY := <Operand>,
         DEST_VARIABLE => <Operand>,
         POS := <Operand>)
```

パラメータ

次の表に、「ビッグエンディアン形式のデータの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
SRC_ARRAY	Input	バイトの配列	バイトの配列	I、Q、M、D、L	読み取り元のメモリ領域
DEST_VARIABLE	Output	BIT 文字列、整数、浮動小数点数、TOD、DATE、CHAR、WCHAR	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	読み出し値
POS	InOut	DINT	DINT	I、Q、M、D、L	読み出しが開始される位置を決定します。POS

					パラメータは、ゼロベースで計算されます。
ファンクション値 (RET_VAL)		INT	INT	I、Q、M、D、L	エラー情報

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#TagResult := READ_BIG(SRC_ARRAY := #SourceField,
                       DEST_VARIABLE => #DINTVariable,
                       POS := #TagPos);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_ARRAY	#SourceField	BYTE の ARRAY [0..10] := 16#1A、16#2B、16#3C、16#4D
DEST_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
POS	#TagPos	0 => 4

この命令は、#SourceField メモリ領域から整数 439_041_101 を読み出して、これを#DINTVariable オペランドに、ビッグエンディアン形式で書き込みます。DEST_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。 個数 4 が#TagPos オペランドに格納されます。

WRITE_BIG: ビッグエンディアン形式のデータの書き込み 

説明

「ビッグエンディアン形式のデータの書き込み」命令を使用して、ビッグエンディアンバイトシーケンスの単一のタグのデータをメモリ領域に書き込みます。 ビッグエンディアン形式では、最上位ビットのバイトが最初(最下位メモリアドレス)に保存されます。

SRC_VARIABLE および DEST_ARRAY パラメータのデータタイプは、VARIANT です。 ただし、パラメータを相互接続できるデータタイプに関する、いくつかの制限事項があります。 SRC_VARIABLE パラメータの VARIANT は、基本データタイプをポイントする必要があります。 DEST_ARRAY パラメータの VARIANT が、データが書き込まれるメモリ領域をポイントしており、これはバイトの配列であることが必要です。

POS パラメータのオペランドは、書き込みが開始されるメモリ領域内の位置を決定します。

注記

データタイプ VARIANT または BOOL のタグの書き込み

VARIANT が指すタグを書き込む場合、「シリアル化」または「逆シリアル化」命令を使用します。

データタイプ BOOL のタグを書き込む場合、「スライスアクセス」を使用します。

構文

「ビッグエンディアン形式のデータの書き込み」命令には、以下の構文を使用します。

```
SCL
WRITE_BIG(SRC_VARIABLE := <Operand>,
          DEST_ARRAY := <Operand>,
          POS := <Operand>)
```

パラメータ

次の表に、「ビッグエンディアン形式のデータの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
SRC_VARIABLE	Input	BIT 文字列、整数、浮動小数点数、TOD、DATE、CHAR、WCHAR	BIT 文字列、整数、浮動小数点数、LDT、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L	書き込み元のタグ
DEST_ARRAY	InOut	バイトの配列	バイトの配列	I、Q、M、D、L	データが書き込まれるメモリ領域
POS	InOut	DINT	DINT	I、Q、M、D、L	書き込みが開始される位置を決定します。

					POS パラメータは、ゼロベースで計算されます。
ファンクション値 (RET_VAL)		INT	INT	I、Q、M、D、L	エラー情報

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	SRC_ARRAY パラメータのデータタイプがバイトの配列ではありません。
8382	POS パラメータの値が、配列の限界値外です。
8383	POS パラメータの値は配列の限界値内にありますが、メモリ領域のサイズが配列の上限値を超過しています。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#TagResult := WRITE_BIG(SRC_VARIABLE := #DINTVariable,
                        DEST_ARRAY := #TargetField,
                        POS := #TagPos);
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
SRC_VARIABLE	#DINTVariable	439041101 16#1A2B3C4D
DEST_ARRAY	#TargetField	BYTE の ARRAY [0..10] = 16#1A、16#2B、16#3C、16#4D
POS	#TagPos	0 => 4

この命令は、#DINTVariable オペランドから整数 439_041_101 を#TargetField メモリ領域に、ビッグエンディアン形式で書き込みます。SRC_VARIABLE パラメータのデータタイプは、読み出すバイト数を指定します。 個数 4 が#TagPos オペランドに格納されます。

VARIANT

この章には下記に関する情報が記載されています：

- VariantGet: VARIANT タグ値の読み出し (S7-1200, S7-1500)
- VariantPut: VARIANT タグ値の書き込み (S7-1200, S7-1500)
- CountOfElements: 配列要素の数を取得 (S7-1200, S7-1500)

VariantGet: VARIANT タグ値の読み出し

説明

「VARIANT タグ値の読み出し」命令を使用して、SRC パラメータの VARIANT が指すタグの値を読み出し、その値を DST パラメータのタグに書き込むことができます。

SRC パラメータのデータタイプは、VARIANT です。 DST パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

DST パラメータで指定されたタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT: ブロックムーブ」命令を使用できます。 追加情報については、「関連項目」を参照してください。

構文

VARIANT タグ値の読み出し」命令には、以下の構文を使用します。

```
SCL
VariantGet(SRC := <Operand>,
           DST => <Operand>)
```

パラメータ

次の表に、「VARIANT タグ値の読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	VARIANT	I、Q、M、L	読み出されるタグ
DST	Output	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
VariantGet(SRC := #TagIn_Source,
           DST => "TagOut_Dest");
```

「#TagIn_Source」オペランドの VARIANT が指すタグの値が読み出され、「TagOut_Dest」オペランドに書き込まれます。

VariantPut: VARIANT タグ値の書き込み

説明

「VARIANT タグ値の書き込み」命令を使用して、SRC パラメータのタグの値を VARIANT が指す DST パラメータのタグに書き込むことができます。

DST パラメータのデータタイプは、VARIANT です。 SRC パラメータでは、VARIANT を除く任意のデータタイプを指定できます。

SRC パラメータのタグのデータタイプは、VARIANT が指すデータタイプと一致する必要があります。

注記
構造体および配列をコピーする場合、「MOVE_BLK_VARIANT: ブロックムーブ」命令を使用できます。 追加情報については、「関連項目」を参照してください。

構文

VARIANT タグ値の書き込み」命令には、以下の構文を使用します。

```
SCL
VariantPut(SRC := <Operand>,
           DST := <Operand>)
```

パラメータ

次の表に、「VARIANT タグ値の書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	ビット列、整数、浮動小数点数、タイマ、日付と時刻、文字列、配列要素、PLC データタイプ	I、Q、M、D、L	読み出されるタグ
DST	Input	VARIANT	I、Q、M、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
VariantPut(SRC := "TagIn_Source",
           DST := "#TagIn_Dest");
```

「TagIn_Source」オペランドの値が、「#TagIn_Dest」オペランドの VARIANT が指すタグに書き込まれます。

CountOfElements: 配列要素の数を取得

説明

「ARRAY エレメント数の取得」命令を使用して、VARIANT が指しているタグが所有する ARRAY エレメント数を照会できます。

配列が一次元の ARRAY の場合、上限値と下限値の差 + 1 が結果として出力されます。 配列が多次元の ARRAY の場合、すべての次元の積が結果として出力されます。

<Operand>のデータタイプは、VARIANT データタイプであることが必要です。

VARIANT タグが ARRAY でない場合、結果は「0」になります。

VARIANT が ARRAY of BOOL を指す場合、FILL エレメントはカウントに含まれます。 (たとえば、ブールの配列[0..1]の場合、8 が返されます)

構文

「ARRAY エレメント数の取得」命令には、以下の構文を使用します。

```
SCL
CountOfElements(<Operand>)
```

パラメータ

次の表に、「ARRAY エレメント数の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	VARIANT	I、Q、M、L	照会するタグ
ファンクション値		UDINT	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
SCL
IF IS_ARRAY(#Tag_VARIANTToArray) THEN
"Tag_Result" := CountOfElements(#Tag_VARIANT);
END_IF;
```

VARIANT が指すタグが ARRAY の場合、ARRAY エレメントの数が出力されます。

レガシー

この章には下記に関する情報が記載されています：

- BLKMOV: ブロックムーブ (S7-1500)
- UBLKMOV: 割り込み不可能なブロックムーブ (S7-1500)
- FILL: フィル命令 (S7-1500)

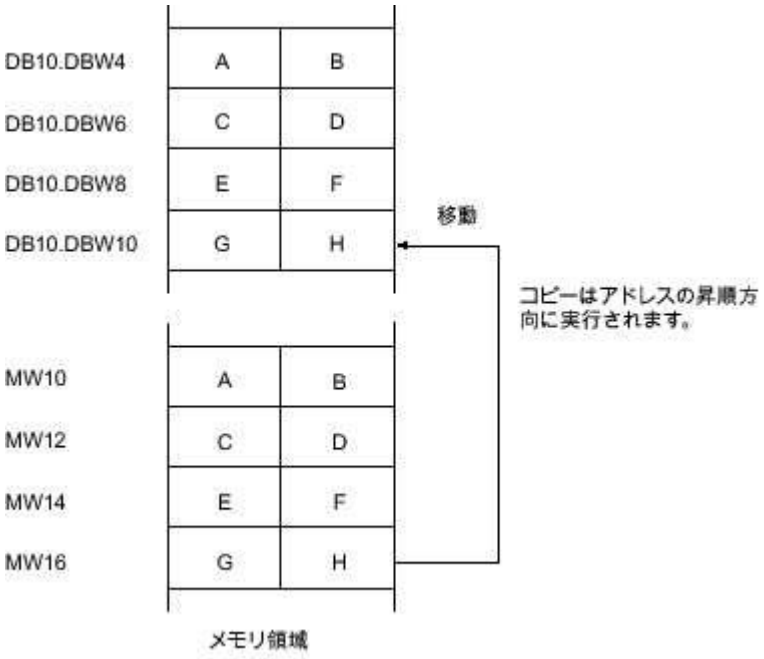
BLKMOV: ブロックムーブ

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

次の図に、ムーブ操作の原理を示します。



ソース領域および宛先領域の整合性

「ブロックの移動」命令の実行中は、宛先データの整合性を保証するため、ソースデータは変更されないことに注意してください。

割り込み機能

ネストレベルに対する制限はありません。

メモリ領域

「ブロックの移動」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力

- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。 ソース領域と宛先領域の長さが異なる場合、小さな領域の長さのみが移動します。

ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

文字列の移動のルール

STRING データタイプの移動元および移動先の領域の移動に「ブロックの移動」命令を使用することも可能です。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報も、宛先領域に書き込まれます。 ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。

文字列の最大長および実際の長さに関する情報を移動するには、SRCBLK および DSTBLK パラメータにバイト単位で領域を指定します。

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
SCL
BLKMOV(SRCBLK := <Operand>,
       DSTBLK => <Operand>)
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
DSTBLK	Output 1)	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	エラー情報
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目:"GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_RetVal" := BLKMOV(SRCBLK := P#M100.0 BYTE 10,
                       DSTBLK => P#DB1.DBX0.0 BYTE
10);
```

この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。 ムーブ操作中にエラーが発生した場合、「Tag_RetVal」タグにエラーコードが出力されます。

UBLKMOV: 割り込み不可能なブロックムーブ

説明

「割り込みなしブロック転送」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動することができます。 ムーブ操作は、アドレスの昇順方向に行われます。 VARIANT を使用して、ソース領域と宛先領域を定義します。

他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。結果として、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

メモリ領域

「割り込みなしブロック転送」命令を使用すると、次のメモリ領域を移動することが可能です。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

「割り込みなしブロック転送」命令の実行中は、ソース領域および宛先領域が重複してはなりません。ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。 宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

仮パラメータとして定義されたソース領域または宛先領域が、SRCBLK または DSTBLK パラメータで指定された宛先領域またはソース領域よりも小さい場合、データは転送されません。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

文字列の移動のルール

「割り込み不可能なブロックムーブ」命令を使用して、データタイプ STRING のソース領域および宛先領域を移動することもできます。 ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報は、宛先領域には書き込まれません。 ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。 STRING データタイプの領域を移動する場合、領域の長さとして「1」を指定します。

構文

以下の構文が「割り込みなしブロック転送」命令に使用されます。

```
SCL
UBLKMOV(SRCBLK := <Operand>,
        DSTBLK => <Operand>)
```

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
DSTBLK	Output 1)	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	エラー情報
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
8091	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_RetVal" := UBLKMOV(SRCBLK := P#M100.0 BYTE 10,
                        DSTBLK => P#DB1.DBX0.0 BYTE
10);
```

この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。ムーブ操作中にエラーが発生した場合、「Tag_RetVal」タグにエラーコードが出力されます。

FILL: フィル命令

説明

「フィル」命令を使用して、メモリ領域(宛先領域)を別のメモリ領域(ソース領域)の内容によって埋めることができます。 「フィル」命令は、宛先領域がすべて書き込まれるまでソース領域の内容を宛先領域に移動します。 ムーブ操作は、アドレスの昇順方向に行われます。

VARIANT を使用して、ソース領域と宛先領域を定義します。

注記

ANY データタイプを使用して、ソース領域と宛先領域を定義することもできます。

ANY データタイプを使用する場合、STRING データタイプおよび WSTRING データタイプに関して以下の事項に従う必要があります。

- STRING (宛先領域)の後に ANY を使用して STRING(ソース領域)を割り当てる場合、STRING の内容が、完全になるまで宛先領域に繰り返しコピーされます。

  ソース領域: String#'STEP7-SCL-TIA-Portal'

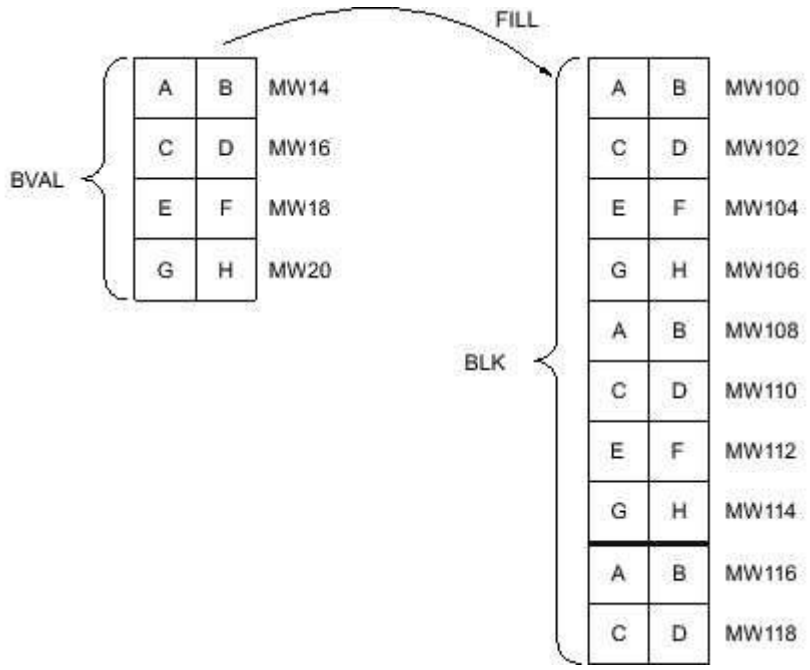
  宛先領域:'STEP7-SCL-TIA-PortalSTEP7-SCL-TIA-PortalSTEP7-SCL'

注記

この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。 保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

「最適化したブロックアクセス」属性を持つブロックの場合、「FILL_BLK: フィル」命令を使用できます。

次の図に、ムーブ操作の原理を示します。



例: MW100～MW118 の範囲の内容は、メモリワード MW14～MW20 の内容で事前割り当てされます。

ソース領域および宛先領域の整合性

「フィル」命令の実行中は、宛先データの整合性を保証するため、ソースデータは変更されないことに注意してください。

メモリ領域

「フィル」命令を使用して、以下のメモリ領域を移動することができます。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。 プリセットする宛先領域が BVAL 入力パラメータの長さの整数倍ではなくても、宛先領域は最後のバイトまで書き込まれます。

プリセットする宛先領域がソース領域よりも小さい場合は、ソース領域に含まれているデータのうち、宛先領域に書き込める量のみがコピーされます。

実際に存在する宛先領域またはソース領域がソース領域または宛先領域に割り当てられたメモリ領域よりも小さい場合(BVAL、BLK パラメータ)、データは転送されません。

ANY ポインタ(ソースまたは宛先)がデータタイプ BOOL の場合、これを絶対アドレス指定する必要があり、かつ指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行されません。

宛先領域が STRING データタイプの場合、この命令は管理情報を含む文字列全体を書き込みます。

構文

「フィル」命令には、以下の構文を使用します。

```
SCL
FILL(BVAL := <Operand>,
     BLK => <Operand>)
```

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
BVAL	Input	VARIANT	I、Q、M、L、P	その内容に内容が BLK パラメータの宛先領域を埋めるのに使用されるメモリ領域(ソース領域)の指定。
BLK	Output [1)]	VARIANT	I、Q、M、L、P	ソース領域の内容で埋められるメモリ領域の指定。
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	エラー情報

1) データがタグに流れるため、BLK パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

BVAL パラメータ

構造体を入力パラメータとして転送する際には、構造体の長さのバイト数が必ず偶数になるように注意してください。 構造体を奇数バイトによって宣言する場合は、追加の 1 バイトが必要になります。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみに存在します。
8152	WSTRING、WCHAR および BOOL データタイプは BVAL パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは BLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_RetVal" := FILL(BVAL := P#M14.0 WORD 4,
                     BLK => P#M100.0 WORD 10);
```

この命令は、ソース領域 MW14～MW20 をコピーし、BVAL パラメータのメモリ領域に含まれる 4 ワードの内容で宛先領域 MW100～MW118 を埋めます。

変換操作

この章には下記に関する情報が記載されています：

CONVERT: 値の変換

説明

「データの変換」命令を使用して、明示的な変換をプログラムします。 この命令が挿入されると、[CONVERT]ダイアログが開きます。 このダイアログで、変換のソースデータタイプと宛先データタイプを指定します。 ソース値が読み出され、指定された宛先データタイプに変換されます。

実行可能な変換に関する情報は、「関連項目」の「明示的な変換」のセクションを参照してください。

構文

「値の変換」命令には、以下の構文を使用します。

```
SCL
<DestinationValue> := <???_TO_???(<SourceValue>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Source_value>	Input	2 進数、整数、浮動小数点数、時刻、日付と時刻、文字列、BCD16、BCD32	I、Q、M、D、L、P、または定数	変換する値
???_TO_???	-	-		変換されるデータタイプを指定するファンクション。
<Target_value>	Output	2 進数、整数、浮動小数点数、時刻、日付と時刻、文字列、BCD16、BCD32	I、Q、M、D、L、P、または定数	変換の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_INT" := REAL_TO_INT("Tag_REAL");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	データタイプ	値
Tag_REAL	REAL	20.56
Tag_INT	INT	21

変換中にオペランド「Tag_REAL」の値は近似の整数に四捨五入され、オペランド「Tag_INT」内に保存されます。

ROUND: 数値の四捨五入

説明

「数値の四捨五入」命令は、入力 IN の値を近似の整数に四捨五入するのに使用します。 この命令は、入力 IN の値を浮動小数点数として解釈し、整数または浮動小数点数に変換します。 入力値が偶数と奇数のちょうど間にある場合、偶数が選択されます。

構文

「数値の四捨五入」命令には、以下の構文を使用します。

```
SCL
ROUND(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	四捨五入される入力値
ファンクション値		整数、浮動小数点数	I、Q、M、D、L、P	四捨五入の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ROUND("Tag_Value");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Value	1.50000000	-1.50000000
Tag_Result	2	-2

CEIL: 浮動小数点数から次に大きい整数を生成

説明

「浮動小数点数から次に大きい整数を生成」命令を使用し、値を近似の整数に四捨五入します。 この命令は入力値を浮動小数点数として解釈し、次に大きい整数に変換します。 ファンクション値は、入力値以上になることがあります。

構文

「浮動小数点数から次に大きい整数を生成」命令には、次の構文を使用します。

SCL

```
CEIL(<Expression>)
CEIL_<Data type>(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
_<Data type>		整数、浮動小数点数。 既定値: DINT	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		整数、浮動小数点数	I、Q、M、D、L	四捨五入された入力値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := CEIL("Tag_Value");
"Tag_Result2" := CEIL_REAL("Tag_Value");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Value	0.5	-0.5
Tag_Result1	1	0
Tag_Result2	1.0	0.0

命令の結果は、ファンクション値としてオペランド「Tag_Resultxy」内に返されます。

FLOOR: 浮動小数点数から次に小さい整数を生成

説明

「浮動小数点数から次に小さい整数を生成」命令を使用し、浮動小数点数の値を次に小さい整数に変換します。 この命令は入力値を浮動小数点数として解釈し、浮動小数点数の値を次に小さい整数に変換します。 ファンクション値は、入力値以下になることがあります。

構文

「浮動小数点数から次に小さい整数を生成」命令には、次の構文を使用します。

```
SCL
FLOOR(<Expression>)
FLOOR_<Data type>(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L、P	入力値
_<Data type>		整数、浮動小数点数。既定値:DINT	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		整数、浮動小数点数	I、Q、M、D、L	四捨五入された入力値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := FLOOR("Tag_Value");
"Tag_Result2" := FLOOR_REAL("Tag_Value");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Value	0.5	-0.5
Tag_Result1	0	-1
Tag_Result2	0.0	-1.0

命令の結果は、ファンクション値としてオペランド「Tag_Resultxy」内に返されます。

TRUNC: 値の切り捨て

説明

「数値を切り捨てる」命令は、四捨五入せずに入力値から整数を生成するのに使用します。 この命令は入力値の整数部のみを選択し、小数を含めずにこの部分をファンクション値として返します。

構文

「数値を切り捨てる」命令には、以下の構文を使用します。

SCL

```
TRUNC(<Expression>)
TRUNC_<Data type>(<Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Expression>	Input	浮動小数点数	I、Q、M、D、L	入力値
_<Data type>		整数、浮動小数点数 既定値:DINT	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		整数、浮動小数点数	I、Q、M、D、L	入力値の整数コンポーネント

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := TRUNC("Tag_Value1");
"Tag_Result2" := TRUNC("Tag_Value2"+"Tag_Value3");
"Tag_Result3" := TRUNC_SINT("Tag_Value4");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

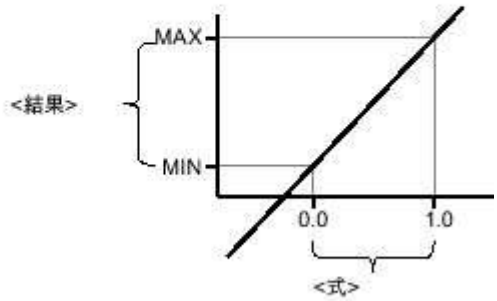
オペランド	値
Tag_Value1	-1.5
Tag_Result1	-1
Tag_Value2	2.1
Tag_Value3	3.2
Tag_Result2	5
Tag_Result3	2
Tag_Value4	2.4

SCALE_X:スケール

説明

「スケール」命令を使用し、浮動小数点数を指定した値の範囲にマッピングしてスケーリングします。MIN および MAX パラメータで値の範囲を指定します。 スケーリングの結果は整数になります。

次の図に、値をどのようにスケーリングできるかを示します。



「スケール」命令は、以下の等式で機能します。

OUT = [VALUE * (MAX – MIN)] + MIN

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

構文

「スケール」命令には、以下の構文を使用します。

```
SCL
SCALE_X(MIN := <Operand>,
        VALUE := <Operand>,
        MAX := <Operand>)
SCALE_X_<Data type>(MIN := <Operand>,
                    VALUE := <Operand>,
                    MAX := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
MIN	Input	整数、浮動小数点数	I、Q、M、D、L	値の範囲の下限値
VALUE	Input	浮動小数点数	I、Q、M、D、L	スケーリングされる値 定数が指定された場合、これを宣言する必要があります。
MAX	Input	整数、浮動小数点数	I、Q、M、D、L	値の範囲の上限値
_<Data type>		整数、浮動小数点数 既定値: INT	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		整数、浮動小数点数	-	スケーリングの結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

SCL

```
"Tag_Result1" := SCALE_X(MIN := "Tag_Value1",
                         VALUE := "Tag_Real",
                         MAX := "Tag_Value2");
"Tag_Result2" := SCALE_X_REAL(MIN := "Tag_Value1",
                              VALUE := "Tag_Real",
                              MAX := "Tag_Value2");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

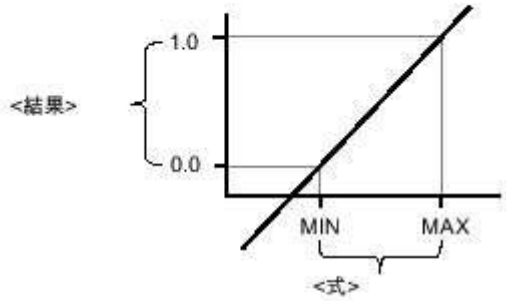
オペランド	値
Tag_Real	0.5
Tag_Value1	10
Tag_Value2	30
Tag_Result1	20
Tag_Result2	20.0

NORM_X: 正規化

説明

「正規化」命令を使用して、VALUE 入力のタグの値をリニアスケーリングにマッピングして正規化することができます。 MIN および MAX パラメータを使用し、スケーリングに適用する値の範囲の限界を定義できます。 この値の範囲の正規化された値の位置に応じて、出力 OUT で結果が計算され、浮動小数点数として格納されます。 正規化される値が入力 MIN の値と等しい場合、この命令は「0.0」を結果として返します。 正規化される値が入力 MAX の値と等しい場合、この命令は「1.0」を結果として返します。

次の図に、値が正規化される例を示します。



「正規化」命令は、以下の等式で機能します。

OUT = (VALUE – MIN) / (MAX – MIN)

注記
アナログ値の変換についての詳細は、それぞれのマニュアルを参照してください。

構文

「正規化」命令には、以下の構文を使用します。

```
SCL
NORM_X(MIN := <Operand>,
       VALUE := <Operand>,
       MAX := <Operand>)
NORM_X_<Data type>(MIN := <Operand>,
```

```
VALUE := <Operand>,
MAX := <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
MIN 1)	Input	整数、浮動小数点数	I、Q、M、D、L	値の範囲の下限値
VALUE 1)	Input	整数、浮動小数点数	I、Q、M、D、L	正規化する値
MAX 1)	Input	整数、浮動小数点数	I、Q、M、D、L	値の範囲の上限値
_<Data type>		浮動小数点数 既定値:REAL	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		浮動小数点数	I、Q、M、D、L	正規化の結果
1) これらの 3 つのパラメータで定数を使用する場合は、そのいずれかを宣言するのみで済みます。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result1" := NORM_X(MIN := "Tag_Value1",
                        VALUE := "Tag_InputValue",
                        MAX := "Tag_Value2");
"Tag_Result2" := NORM_X_LREAL(MIN := "Tag_Value1",
                              VALUE := "Tag_InputVal-
ue",
```

```
                                        MAX := "Tag_Value2");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InputValue	20
Tag_Value1	10
Tag_Value2	30
Tag_Result1	0.5
Tag_Result2	0.5

VARIANT

この章には下記に関する情報が記載されています：

- VARIANT_TO_DB_ANY: VARIANT の DB_ANY への変換 (S7-1200, S7-1500)
- DB_ANY_TO_VARIANT: DB_ANY の VARIANT への変換 (S7-1200, S7-1500)

VARIANT_TO_DB_ANY: VARIANT の DB_ANY への変換

説明

「VARIANT の DB_ANY への変換」命令を使用して、IN パラメータがアドレス指定するオペランドのデータブロック番号を照会します。 インスタンスデータブロックまたは配列データブロックにすることはできません。 IN パラメータのオペランドはデータタイプ VARIANT です。つまり、プログラムの作成時に照会する番号のデータブロックのデータタイプが既知である必要があります。 データブロック番号はランタイム中に読み取られ、RET_VAL パラメータで指定されたオペランドに書き込まれます。

必要条件

必要条件が満たされる場合は、命令が実行されます。 必要条件が満たされない場合は、データブロック番号として「0」が出力されます。

出力タグ...	原点復帰...	変換オプション
VARIANT	... データブロックであり、PLC データタイプまたはシステムデータタイプ(SDT)のインスタンスデータブロックです。	出力タグをデータブロック番号に変換できます。
VARIANT	... 配列 DB であるデータブロック。	出力タグをデータブロック番号に変換できます。
VARIANT	... 基本データタイプのタグ。	データブロックを 1 つの基本データタイプのみで構成することはできないため、出力タグをデータベース番号に変換できません。
VARIANT	... データブロック内の構造体。	これはデータブロック内の一部にすぎないため、出力タグをデータベース番号に変換できません。

構文

「VARIANT の DB_ANY への変換」命令には、以下の構文を使用します。

```
SCL
VARIANT_TO_DB_ANY(IN  := <Operand>,
                  ERR => <Operand>)
```

パラメータ

次の表に、「VARIANT の DB_ANY への変換」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	VARIANT	I、Q、M、L	読み出されるタグ

ERR	Output	INT	I、Q、M、D、L	エラー情報
ファンクション値 (RET_VAL)		DB_ANY	I、Q、M、D、L	結果: DB の番号

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR パラメータ

次の表に、ERR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
252C	IN パラメータの VARIANT データタイプの値は「0」であり、CPU が STOP モードに切り替わります。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8131	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。
8132	データブロックが小さすぎるか、または配列データブロックではありません。
8150	IN パラメータのデータタイプ VARIANT が、値「0」を提供します。 このエラーメッセージを受信するには、「ブロック内でのエラー処理」ブロックプロパティが有効であることが必要です。 有効でない場合、CPU が STOP モードに切り替わり、エラーコード 16#252C を送信します。
8153	IN パラメータの VARIANT データタイプが配列データブロックの開始を指していないか、または VARIANT の長さがデータブロックの長さと一致しません。
8154	データブロックのデータタイプが不正です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"OutputDBNumber" := VARIANT_TO_DB_ANY(IN := "Input-
Tag",
                                      ERR := "Tag_Er-
ror");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
IN	Input	#InputTag	-
<Function value>	Output	OutputDBNumber	11

#InputTag オペランドで指定されたデータブロックの番号が読み取られます。 オペランドのデータタイプが VARIANT であるため、プログラムの作成中にタグのデータタイプを知っている必要はありません。 この番号が、データタイプが DB_ANY である「OutputDBNumber」タグに書き込まれます。

DB_ANY_TO_VARIANT: DB_ANY の VARIANT への変換

説明

「DB_ANY の VARIANT への変換」命令を使用して、以下にリストした要件を満たすデータブロックから VARIANT タグを生成します。 IN パラメータのオペランドはデータタイプ DB_ANY です。つまり、プログラムの作成時にデータブロックが既知である必要はありません。 データブロックが番号がランタイム中に読み取られます。

必要条件

必要条件が満たされる場合は、命令が実行されます。 要件が満たされない場合またはデータブロックが存在しない場合、RET_VAL パラメータで値 NULL が出力されます。 RET_VAL タグによるその他すべてのアクセスは失敗します。

データタイプの入力タグ...	原点復帰...	変換オプション
DB_ANY	...データブロックであり、PLC データタイプまたはシステムデータタイプ(SDT)のインスタンスデータブロック。	変換が可能です。
DB_ANY	...配列 DB であるデータブロック。	変換が可能です。
DB_ANY	...ファンクションブロックのインスタンスデータブロックまたはグローバルデータブロックであるデータブロック。	変換が不可能です。

構文

「DB_ANY の VARIANT への変換」命令には、以下の構文を使用します。

```
SCL
DB_ANY_TO_VARIANT(IN  := <Operand>,
                  ERR => <Operand>)
```

パラメータ

次の表に、「DB_ANY の VARIANT への変換」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	DB_ANY	I、Q、M、D、L	参照され、読み出されるタグ
ERR	Output	INT	I、Q、M、D、L	エラー情報
ファンクション値 (RET_VAL) 1)		VARIANT	I、Q、M、L	データブロックの番号
1) データがタグに流れるため、RET_VAL パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR パラメータ

次の表に、ERR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8130	データブロックの番号が「0」です
8131	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。
8132	データブロックが小さすぎるか、または配列データブロックではありません。
8134	データブロックが書き込み保護されています。
8154	データブロックのデータタイプが不正です。
8155	データブロックのデータタイプが不明です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#tempVARIANT := DB_ANY_TO_VARIANT(IN := "InputDB",
                                  ERR := "Tag_Error");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
IN	Input	InputDB	11
<Function value>	Temp	#tempVARIANT	-

「InputDB」オペランドで指定された任意のデータブロックの番号が、そのデータブロックをアドレス指定するデータタイプ VARIANT のタグの生成に使用されます。 IN パラメータのオペランドのデータタイプは DB_ANY であるため、ランタイム中に使用されるデータブロックはプログラムの作成時に既知である必要はありません(データブロックの名前も番号も不明でかまいません)。 RET_VAL パラメータのオペランドのデータタイプは VARIANT であるため、プログラムの作成時にデータブロックのデータタイプが既知である必要はありません。

レガシー

この章には下記に関する情報が記載されています：

- SCALE: スケール (S7-1500)
- UNSCALE: スケール解除 (S7-1500)

SCALE: スケール

説明

「スケール」命令は、IN パラメータの整数を下限値と上限値間の物理単位でスケーリングが可能な浮動小数点数に変換します。 LO_LIM および HI_LIM パラメータを使用して、入力値をスケーリングする範囲の下限値と上限値を指定することができます。 命令の結果は、OUT パラメータに出力されます。

「スケール」命令は、以下の等式で機能します。

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は-27648.0 となり、定数「K2」の値は+27648.0 となります。
- シグナル状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は 0.0 となり、定数「K2」の値は+27648.0 となります。

IN パラメータの値が定数値「K2」よりも大きい場合、この命令の結果が上限値(HI_LIM)にセットされ、エラーが出力されます。

IN パラメータの値が定数値「K1」未満の場合、この命令の結果が下限値(LO_LIM)にセットされ、エラーが出力されます。

示された下限値が上限値よりも大きい(LO_LIM >HI_LIM)場合、結果は入力値に反比例してスケーリングされます。

構文

「スケール」命令には、以下の構文を使用します。

```
SCL
SCALE(IN := <Expression>,
      HI_LIM := <Operand>,
      LO_LIM := <Operand>,
      BIPOLAR := <Operand>,
      OUT => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	INT	I、Q、M、D、L、P	スケーリングする入力値。
HI_LIM	Input	REAL	I、Q、M、D、L、P	上限値

LO_LIM	Input	REAL	I、Q、M、D、L、P	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
OUT	Output	REAL	I、Q、M、D、L、P	命令の結果
ファンクション値 (RET_VAL)		WORD	I、Q、M、D、L、P	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が 27 648 を超えているか、または 0 (ユニポーラの場合)や-27 648 (バイポーラの場合)未満になっています。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_ErrorCode" := SCALE(IN := "Tag_InputValue",
                         HI_LIM := "Tag_HighLimit",
                         LO_LIM := "Tag_LowLimit",
                         BIPOLAR := "Tag_Bipolar",
                         OUT => "Tag_Result");
```

エラー情報は、ファンクション値としてオペランド「Tag_ErrorCode」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22

HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	50.03978588
RET_VAL	Tag_ErrorCode	W#16#0000

UNSCALE: スケール解除

説明

「スケール解除」命令は、IN パラメータの浮動小数点数を下限値と上限値間の物理単位にスケーリング解除し、それらを整数に変換します。 LO_LIM および HI_LIM パラメータを使用して、入力値をスケーリング解除する範囲の下限値と上限値を指定することができます。 命令の結果は、OUT パラメータに出力されます。

「スケール解除」命令は、以下の等式で機能します。

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値がバイポーラであり、-27648 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は-27648.0 となり、定数「K2」の値は+27648.0 となります。
- シグナル状態「0」: IN パラメータの値がユニポーラであり、0 から 27648 の範囲の値にあると想定されます。この場合、定数「K1」の値は 0.0 となり、定数「K2」の値は+27648.0 となります。

IN パラメータの値が定数値「HI_LIM」よりも大きい場合、この命令の結果が定数値(K2)にセットされ、エラーが出力されます。

IN パラメータの値が下限値の定数値「LO_LIM」未満の場合、この命令の結果が定数値(K1)にセットされ、エラーが出力されます。

構文

「スケール解除」命令には、以下の構文を使用します。

```
SCL
UNSCALE(IN := <Expression>,
        HI_LIM := <Operand>,
        LO_LIM := <Operand>,
        BIPOLAR := <Operand>,
        OUT => <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	REAL	I、Q、M、D、L、P	整数値にスケーリング解除する入力値。
HI_LIM	Input	REAL	I、Q、M、D、L、P	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P	下限値

BIPOLAR	Input	BOOL	I、Q、M、D、L	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
OUT	Output	INT	I、Q、M、D、L、P	命令の結果
ファンクション値 (RET_VAL)		WORD	I、Q、M、D、L、P	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が上限値(HI_LIM)を超えているか、または下限値(LO_LIM)未満になっています。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
"Tag_ErrorCode" := UNSCALE(IN := "Tag_InputValue",
                           HI_LIM := "Tag_HighLimit",
                           LO_LIM := "Tag_LowLimit",
                           BIPOLAR := "Tag_Bipolar",
                           OUT => "Tag_Result");
```

エラー情報は、ファンクション値としてオペランド「Tag_ErrorCode」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0

BIPOLAR	Tag_Bipolar	1
OUT	Tag_Result	22
RET_VAL	Tag_ErrorCode	W#16#0000

プログラム制御演算

この章には下記に関する情報が記載されています：

IF: 条件実行

説明

「条件実行」命令は、条件に応じてプログラムの流れを分岐させます。 条件はブール値(TRUE または FALSE)の式です。 論理式または比較式は条件として記述することができます。

命令が実行されると、記述された式が評価されます。 式の値が TRUE の場合は条件を満たしていて、値が FALSE の場合は満たしていません。

構文

分岐のタイプに応じて、命令の以下の形式をプログラムできます。

- IF を使用した分岐:

```
SCL
IF <Condition> THEN <Instructions>
END_IF;
```

条件を満たす場合、THEN の後にプログラムされた命令が実行されます。 条件を満たさない場合、プログラムの実行は END_IF の後の次の命令に継続します。

- IF および ELSE を使用した分岐:

```
SCL
IF <Condition> THEN <Instructions1>
ELSE <Instructions0>;
END_IF;
```

条件を満たす場合、THEN の後にプログラムされた命令が実行されます。 条件を満たさない場合、ELSE の後にプログラムされた命令が実行されます。 その後、プログラムの実行は END_IF の後の次の命令に継続します。

- IF、ELSIF、および ELSE を使用した分岐:

```
SCL
IF <Condition1> THEN <Instructions1>
ELSIF <Condition2> THEN <Instruction2>
ELSE <Instructions0>;
END_IF;
```

最初の条件(<Condition1>)が満たされると、THEN の後の命令 (<Instructions1>)が実行されます。 命令の実行後、プログラムの実行は END_IF の後に継続します。

最初の条件を満たさない場合、次の条件(<Condition2>)がチェックされます。 次の条件(<Condition2>)が満たされると、THEN の後の命令 (<Instructions2>)が実行されます。命令の実行後、プログラムの実行は END_IF の後に継続します。

どの条件も満たされない場合、ELSE の後の命令(<Instructions0>)が実行され、END_IF の後にプログラムが実行されます。

IF 命令内には ELSIF および THEN の組み合わせをいくつでもネストすることができます。 ELSE 分岐のプログラミングはオプションです。

IF 命令の構文は以下の部分で構成されます。

パラメータ	データタイプ	メモリ領域	説明
<Condition>	BOOL	I、Q、M、D、L	評価される式
<Instructions>	-		条件が満たされる場合に実行される命令。 ELSE の後にプログラムされる命令は例外です。 ELSE の後の命令は、プログラムループ内のいずれの条件も満たさない場合に実行されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
IF "Tag_1" = 1
THEN "Tag_Value" := 10;
ELSIF "Tag_2" = 1
THEN "Tag_Value" := 20;
ELSIF "Tag_3" = 1
THEN "Tag_Value" := 30;
ELSE "Tag_Value" := 0;
END_IF;
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値			
Tag_1	1	0	0	0
Tag_2	0	1	0	0
Tag_3	0	0	1	0
Tag_Value	10	20	30	0

CASE: 多岐分岐の作成

説明

「多岐分岐の作成」命令は、数式の値に応じて複数の命令シーケンスの 1 つを実行します。

式の値は整数であることが必要です。 命令実行時、式の値が複数の定数値と比較されます。 式の値が定数値と一致する場合、この定数の直後にプログラムされた命令が実行されます。 以下の定数値が想定されます。

- 整数(たとえば 5)
- 整数の範囲(たとえば 15..20)
- 列挙した整数および範囲(たとえば 10, 11, 15..20)

構文

「多岐分岐の作成」命令には、以下の構文を使用します。

```
SCL
CASE <Expression> OF
<Constant1>: <Instructions1>
<Constant2>: <Instructions2>
<ConstantX>: <InstructionsX>; // X >=3
ELSE <Instructions0>;
END_CASE;
```

命令の構文は以下の部分で構成されます。

パラメータ	データタイプ	メモリ領域	説明
<Expression>	整数	I、Q、M、D、L	プログラムされた定数値と比較される値。
<constant>	整数	I、Q、M、D、L	命令シーケンス実行のための条件となる定数値。以下の定数値が想定されます。 • 整数(たとえば 5) • 整数の範囲(たとえば 15..20) • 列挙した整数および範囲(たとえば 10, 11, 15..20)
<Instruction>	-		式の値が定数値と一致する場合に実行されるすべての命令。 ELSE の後にプログラムされる命令は例外です。 これらの命令は値が一致しない場合に実行されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

式の値が最初の定数値(<Constant1>)と一致する場合、最初の定数の直後にプログラムされた命令(<Instructions1>)が実行されます。 プログラムの実行は、END_CASE の後に継続します。

式の値が最初の定数値(<Constant1>)と一致しない場合、この値は次にプログラムされた定数値と比較されます。 同様に、値が一致するまで CASE 命令が実行されます。 式の値がプログラムされた定数値のどれにも一致しない場合、ELSE の後にプログラムされた式(<Instructions0>)が実行されます。 ELSE は構文のオプション部分であり、省略可能です。

CASE 命令も命令ブロックを CASE に置き換えることでネスト可能です。 END_CASE は CASE 命令の終端を表します。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
CASE "Tag_Value" OF
     0 :
     "Tag_1" := 1;
     1,3,5 :
     "Tag_2" :=1;
     6..10 :
     "Tag_3" := 1;
     16,17,20..25 :
     "Tag_4" := 1;
ELSE "Tag_5" := 1;
END_CASE;
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値				
Tag_Value	0	1, 3 , 5	6, 7, 8, 9, 10	16,17, 20, 21, 22, 23, 24, 25	2
Tag_1	1	-	-	-	-
Tag_2	-	1	-	-	-
Tag_3	-	-	1	-	-
Tag_4	-	-	-	1	-
Tag_5	-	-	-	-	1
1: オペランドがシグナル状態「1」にセットされます。 -: オペランドのシグナル状態は変更されません。					

FOR: カウントループの実行

説明

「カウントループの実行」命令は、ループ変数が指定された値の範囲となるまでプログラムループを繰り返し実行します。

プログラムループもネスト可能です。 プログラムループ内では、他のループ変数を使用した追加のプログラムループをプログラムできます。

現在実行が継続しているプログラムループは、「ループ条件の再チェック」(CONTINUE)命令で終了できます。「ループを直ちに終了」(EXIT)命令は、ループ実行全体を終了します。 このトピックの詳細については、「関連項目」を参照してください。

FOR ステートメントの限界値

エンドレスに実行しない「安全」FOR ステートメントをプログラムするには、以下のルールおよび限界値を遵守してください。

ルール

```
FOR ii := Start TO End BY Step DO
```

If ...	... then	注記
start < end	end < (PMAX step)	実行タグ ii は、立ち上がり方向に実行します
start > end AND step < O	end > (NMAX step)	実行タグ ii は、立ち下がり方向に実行します

制限

さまざまな異なる限界値が以下の可能なデータタイプに適用されます。

データタイプ	PMAX	NMAX
SINT タイプの ii	127	-128
INT タイプの ii	32767	-32768
DINT タイプの ii	2147483647	-2147483648
LINT タイプの ii	9223372036854775807	-9223372036854775808

構文

「カウントループの実行」命令には、以下の構文を使用します。

SCL

```
FOR<Run_tag> := <Start_value> TO<End_value> BY<Increment> DO<Instructions>
END_FOR;
```

パラメータ

次の表に、「カウントループの実行」命令のパラメータを示します。

パラメータ	データタイプ		メモリ領域	説明
	S7-1200	S7-1500		
<Run tag>	SINT, INT, DINT	SINT, INT, DINT, LINT	I、Q、M、D、L	その値がループ実行で評価されるオペランド。 ループ変数のデータタイプにより、他のパラメータのデータタイプが決定します。
<Start value>	SINT, INT, DINT	SINT, INT, DINT, LINT	I、Q、M、D、L	その値がループ変数のループ実行の始点に割り当てられる式。
<End value>	SINT, INT, DINT	SINT, INT, DINT, LINT	I、Q、M、D、L	その値がプログラムループの最終実行を決定する式。 ループ変数の値は毎回ループの前にチェックされます。 • 終了値に未到達: DO 以下の命令が実行されます。 • 終了値に到達: FOR ループがもう 1 回だけ実行されます。 • 終了値を超過: FOR ループを終了します。 命令実行中の終了値の変更は許可されません。
<Increment>	SINT, INT, DINT	SINT, INT, DINT, LINT	I、Q、M、D、L	1 回のループごとにその値の分だけループ変数を増加(正のインクリメント)または減少(負のインクリメント)させる式。 インクリメントの指定はオプションです。 インクリメントが指定されていない場合は、ループ変数の値は 1 回のループごとに 1 増加します。 命令実行中のインクリメントの変更は許可されません。
<Instructions>	-	-		ループ変数の値が値の範囲内に存在する場合に、各ループにおいて実行される命令。 値の範囲は開始値および終了値で指定します。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
FOR i := 2 TO 8 BY 2
    DO "a_array[i] := "Tag_Value"*"b_array[i]";
END_FOR;
```

オペランド「Tag_Value」は、アレイタグ「b_array」の要素(2、4、6、8)で乗算されます。 結果はアレイタグ「a_array」の要素(2、4、6、8)内に読み取られます。

WHILE: 条件が有効ならば実行

説明

「条件が有効ならば実行」命令は、実行条件が満たされるまでプログラムループを繰り返し実行します。 条件はブール値(TRUE または FALSE)の式です。 論理式または比較式は条件として記述することができます。

命令が実行されると、記述された式が評価されます。 式の値が TRUE の場合は条件を満たしていて、値が FALSE の場合は満たしていません。

プログラムループもネスト可能です。 プログラムループ内では、他のループ変数を使用した追加のプログラムループをプログラムできます。

現在実行が継続しているプログラムループは、「ループ条件の再チェック」(CONTINUE)命令で終了できます。「ループを直ちに終了」(EXIT)命令は、ループ実行全体を終了します。 このトピックの詳細については、「関連項目」を参照してください。

構文

「条件が有効ならば実行」命令には、以下の構文を使用します。

```
SCL
WHILE <Condition> DO <Instructions>
END_WHILE;
```

WHILE 命令の構文は以下の部分で構成されます。

パラメータ	データタイプ	メモリ領域	説明
<Condition>	BOOL	I、Q、M、D、L	各ループの前に評価される式。
<Instructions>	-		条件が満たされる場合に実行される命令。 条件が満たされない場合、プログラム実行は END_WHILE の後に継続します。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
WHILE
      "Tag_Value1" <> "Tag_Value2"
      DO "Tag_Result"
      := "Tag_Input";
END_WHILE;
```

オペランド「Tag_Value1」および「Tag_Value2」の値が一致しなければ、オペランド「Tag_Input」の値がオペランド「Tag_Result」に割り当てられます。

REPEAT: 条件が有効でなければ実行

説明

「条件が有効でなければ実行」命令は、終了条件が満たされるまでプログラムループを繰り返し実行します。 条件はブール値(TRUE または FALSE)の式です。 論理式または比較式は条件として記述することができます。

命令が実行されると、記述された式が評価されます。 式の値が TRUE の場合は条件を満たしていて、値が FALSE の場合は満たしていません。

この命令は、終了条件を満たしている場合でも 1 回は実行されます。

プログラムループもネスト可能です。 プログラムループ内では、他のループ変数を使用した追加のプログラムループをプログラムできます。

現在実行が継続しているプログラムループは、「ループ条件の再チェック」(CONTINUE)命令で終了できます。「ループを直ちに終了」(EXIT)命令は、ループ実行全体を終了します。 このトピックの詳細については、「関連項目」を参照してください。

構文

「条件が有効でなければ実行」命令には、以下の構文を使用します。

```
SCL
REPEAT <Instructions>
UNTIL <Condition>END_REPEAT;
```

REPEAT 命令の構文は以下の部分で構成されます。

パラメータ	データタイプ	メモリ領域	説明
<Instructions>	-		プログラムされた条件が値「FALSE」の場合に実行される命令。 この命令は、終了条件を満たしている場合でも 1 回は実行されます。
<Condition>	BOOL	I、Q、M、D、L	各ループの後に評価される式。 式の値が「FALSE」の場合、プログラムループはもう 1 回実行されます。 式の値が「TRUE」の場合、プログラムループは END_REPEAT の後に継続します。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
REPEAT "Tag_Result"
        := "Tag_Value";
```

```
UNTIL "Tag_Error"
END_REPEAT;
```

オペランド「Tag_Error」の値がシグナル状態「0」ならば、オペランド「Tag_Value」の値がオペランド「Tag_Result」に割り当てられます。

CONTINUE: ループ条件の再チェック

説明

「ループ条件の再チェック」命令は、FOR、WHILE、または REPEAT ループの現在のプログラム実行を終了します。

この命令の実行後、プログラムループの実行条件が再度評価されます。 この命令は、命令を直接含んだプログラムループに影響を与えます。

構文

「ループ条件の再チェック」命令には、以下の構文を使用します。

```
SCL
CONTINUE;
```

例

次の例で、命令がどのように動作するかを示します。

```
SCL
FOR i
      := 1 TO 15 BY 2 DO
      IF (i < 5)THEN
      CONTINUE;
END_IF;
       "DB10".Test[i] := 1;
END_FOR;
```

有効なデータタイプの追加情報については、「関連項目」を参照してください。

条件 i < 5 を満たす場合、以降の値割り当て("DB10".Test[i] := 1)は実行されません。 実行変数(i)はインクリメント「2」で増加し、現在の値がプログラムされた値の範囲内にあるかチェックされます。 実行変数が値の範囲にある場合、IF 条件が再度評価されます。

条件 i < 5 を満たさない場合、以降の値割り当て("DB10".Test[i] := 1)が実行され、新しいループが開始します。 この場合も実行変数はインクリメント「2」で増加し、チェックされます。

EXIT: ループを直ちに終了

説明

「ループを直ちに終了」命令は、FOR、WHILE、または REPEAT ループの実行を条件に関わらず任意の場所でキャンセルします。 プログラムの実行は、ループの終端(END_FOR、END_WHILE、END_REPEAT)の後に継続します。

この命令は、命令を直接含んだプログラムループに影響を与えます。

構文

「ループを直ちに終了」命令には、以下の構文を使用します。

```
SCL
EXIT;
```

例

次の例で、命令がどのように動作するかを示します。

```
SCL
FOR i := 15 TO 1 BY -2 DO
IF (i < 5)
THEN EXIT;
END_IF;
"DB10".Test[i] := 1;
END_FOR;
```

有効なデータタイプの追加情報については、「関連項目」を参照してください。

条件 i < 5 を満たす場合、ループの実行はキャンセルされます。 プログラムの実行は、END_FOR の後に継続します。

条件 i < 5 を満たさない場合、以降の値割り当て("DB10".Test[i] := 1)が実行され、新しいループが開始します。 実行タグ(i)はインクリメント「-2」で減少し、現在の値がプログラムされた値の範囲内にあるかチェックされます。 (i)実行変数が値の範囲にある場合、IF 条件が再度評価されます。

GOTO: ジャンプ

説明

「ジャンプ」命令を使用し、ジャンプラベルでマークされた任意の場所からプログラムの実行を再開します。

ジャンプラベルおよび「ジャンプ」命令は同一ブロック内にあることが必要です。 ジャンプラベルの名前は、ブロック内で 1 回だけ割り当てることができます。 各ジャンプラベルを複数のジャンプ命令のターゲットにすることが可能です。

「外部」からのプログラムループ内へのジャンプは許容されませんが、ループからの「外部」へのジャンプは可能です。

構文

「ジャンプ」命令には、以下の構文を使用します。

```
SCL
GOTO <jump label>
...
<Jump label>: <Instructions>
```

GOTO 命令の構文は以下の部分で構成されます。

パラメータ	データタイプ	説明
<Jump label>	-	ジャンプ先のジャンプラベル
<Instructions>	-	ジャンプ後に実行される命令

例

次の例で、命令がどのように動作するかを示します。

```
SCL
CASE "Tag_Value" OF
1 : GOTO MyLABEL1;
2 : GOTO MyLABEL2;
3 : GOTO MyLABEL3;
ELSE GOTO MyLABEL4;
END_CASE;
MyLABEL1: "Tag_1" := 1;
MyLABEL2: "Tag_2" := 1;
MyLABEL3: "Tag_3" := 1;
MyLABEL4: "Tag_4" := 1;
```

オペランド「Tag_Value」の値に応じて、プログラムの実行は対応するジャンプラベルによって特定される場所から再開されます。 たとえば、オペランド「Tag_Value」の値が 2 の場合、プログラム実

行はジャンプラベル「MyLABEL2」から再開されます。 この場合、ジャンプラベル「MyLABEL1」で特定されるプログラム行はスキップされます。

RETURN: ブロック終了

説明

「ブロック終了」命令は、現在編集されているブロック内でのプログラム実行を終了し、呼び出し元ブロック内で継続します。

ブロックの終端ではこの命令を省略可能です。

構文

「ブロック終了」命令には、以下の構文を使用します。

```
SCL
RETURN;
```

例

次の例で、命令がどのように動作するかを示します。

```
SCL
IF "Tag_Error" <>0 THEN RETURN;
END_IF;
```

オペランド「Tag_Error」のシグナル状態がゼロではない場合、プログラムの実行は現在処理中のブロック内で終了します。

(*...*): コメントセクションを挿入

説明

「コメントセクションを挿入」命令を使用して、コメントセクションを追加できます。 括弧内のテキスト「(*...*)」はコメントとして処理されます。

構文

「コメントセクションを挿入」命令には、以下の構文を使用します。

SCL

```
(*...*)
```

例

次の例で、命令がどのように動作するかを示します。

SCL

```
(*これはコメントセクションです。*)
```

ランタイム制御

この章には下記に関する情報が記載されています：

- ENDIS_PW: パスワードの適正化の制限および有効化 (S7-1200, S7-1500)
- RE_TRIGR: サイクルタイムモニタのリスタート (S7-1200, S7-1500)
- STP: プログラム終了 (S7-1200, S7-1500)
- GET_ERROR: ローカルでエラーを取得 (S7-1200, S7-1500)
- GET_ERR_ID: ローカルでエラー ID を取得 (S7-1200, S7-1500)
- INIT_RD: すべての保持データの初期化 (S7-1500)
- WAIT: 遅延時間の設定 (S7-1500)
- ランタイム: プログラムランタイムの測定 (S7-1200, S7-1500)

ENDIS_PW: パスワードの適正化の制限および有効化

説明

「パスワード正当性の制限および有効化」命令を使用して、設定されたパスワードを CPU に対して正当化できるかどうかを指定します。 このため、正しいパスワードがわかっている場合でも、正規の接続を防止することができます。

命令呼び出し時に REQ パラメータのシグナル状態が「0」の場合、現在設定されている状態が、出力パラメータに表示されます。 入力パラメータに変更を加えても、変更は出力パラメータに転送されません。

命令呼び出し時に REQ パラメータのシグナル状態が「1」の場合、シグナル状態は、入力パラメータ(F_PWD、FULL_PWD、R_PWD、HMI_PWD)から取得されます。FALSE は、パスワードごとの正当化が許可されていないことを意味します。TRUE は、パスワードを使用できることを意味します。

パスワードの有効化または無効化を個別に許可または禁止することができます。 たとえば、フェールセーフパスワードを除く、すべてのパスワードを禁止することができます。 したがって、アクセスオプションを小さいユーザーグループに制限することができます。 出力パラメータ(F_PWD_ON、FULL_PWD_ON、R_PWD_ON、HMI_PWD_ON)は、REQ パラメータに関係なく、常にパスワード使用の現在のステータスを示します。

設定済みでないパスワードの場合、入力のシグナル状態が TRUE でなければならず、出力にシグナル状態 TRUE を返します。 フェールセーフパスワードは、F-CPU の場合のみ設定できるため、標準 CPU では、常にシグナル状態 TRUE と相互接続する必要があります。 命令がエラーを返す場合、この呼び出しには効果がなく、直前のロックがまだ有効です。

無効なパスワードは、以下の条件で再度有効にすることができます。

- CPU が工場出荷時設定にリセットされること。
- S7-1500 CPU のフロントパネルが、パスワードを再度有効にすることができる適切なメニューに移動することができるダイアログをサポートしていること。
- 「パスワードの適正化の制限および有効化」命令を呼び出す場合、目的のパスワードの入力パラメータのシグナル状態が「1」であること。
- 動作モードスイッチを STOP にセットします。 このスイッチの設定を RUN に戻すと直ちに、パスワードの適正化に対する制限が再確立されます。
- S7-1200 CPU への空きメモリカード(転送カードまたはプログラムカード)の差し込み。
- S7-1200 CPU では、電源オフから電源オンへの移行によって、保護が無効になります。 この場合、プログラム(たとえば、スタートアップ OB)で、再度、「パスワード正当性の制限および有効化」命令を呼び出す必要があります。

注記
「パスワード正当性の制限および有効化」命令は、HMI パスワードが有効にされていない場合、HMI システムからのアクセスをブロックします。

注記
既存の適正化された接続はアクセス権を保持し、「パスワードの適正化の制限および有効化」命令を使用してそれらを制限することはできません。

S7-1500 CPU の偶発的なロックアウトの防止

CPU のフロントパネルで設定を行うことができ、CPU は最新の設定を保存します。

偶発的なロックアウトを防止するには、S7-1500 CPU でモードセレクタを STOP に設定することによって保護を無効にすることができます。 モードセレクタを RUN に設定することによって、「パスワ

ードの適正化の制限および有効化」命令を再度呼び出したり、フロントパネルで追加の操作を実行したりせずに、保護が自動的に再度有効になります。

S7-1200 CPU の偶発的なロックアウトの防止

S7-1200 CPU には動作モードスイッチが存在しないため、電源オフから電源オンへの移行時の保護は無効です。 このため、ユーザープログラム内の特定のプログラムシーケンスによって偶発的なロックアウトを防止することをお奨めします。

これを行うには、サイクリック割り込み OB、またはメイン OB (OB 1)のタイマのいずれかを使用して、時間制御をプログラミングします。 これによって、電源オフから電源オンへの移行と、関連する保護の無効化の後、比較的迅速に、各 OB (たとえば、OB 1 または OB 35)内で「パスワード正当性の制限および有効化」命令を呼び出すオプションが可能になります。 スタートアップ OB (OB 100)でこの命令を呼び出して、この命令が無効で、パスワード正当化の制限が存在しない時間帯を比較的小さなものに保ちます。 この手順は、未許可のアクセスに対する可能な最大限の保護を提供します。

偶発的なロックアウトが発生した場合は、スタートアップ OB での呼び出しをスキップし(たとえば、入力パラメータを照会することによって)、ロックが再び有効になる前に、CPU へ接続確立する時間(たとえば、10 秒 ~ 1 分)を設定することができます。

ユーザープログラムコードにタイマが存在しない場合にロックアウトが発生した場合は、空の転送カードまたはプログラムカードを CPU に挿入します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 この後、ユーザープログラムを再び STEP 7 から CPU にダウンロードする必要があります。

S7-1200 CPU でパスワードが失われた場合の手順

パスワード保護された S7-1200 CPU のパスワードを紛失した場合は、空の転送カードまたはプログラムカードを使用して、パスワード保護されたプログラムを削除します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 その後、新しいユーザープログラムを STEP 7 Basic から CPU にロードすることができます。

警告

空の転送カードの挿入

ランタイム中に CPU に転送カードを挿入すると、CPU は STOP モードに切り替わります。 動作状態が不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期しない動作につながり、致命的または重大な人的傷害や物的損傷の原因になる恐れがあります。

転送カードを取り出すと、転送カードの内容が内部ロードメモリで使用可能になります。 この時点に、カードにプログラムが含まれていないことを確認してください。

警告

空のプログラムカードの挿入

ランタイム中に CPU にプログラムカードを挿入すると、CPU は STOP モードに移行します。 動作状態が不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期しない動作につながり、致命的または重大な人的傷害や物的損傷の原因になる恐れがあります。

プログラムカードが空であることを確認してください。 内部ロードメモリが空のプログラムカードにコピーされます。 直前に空であったプログラムカードを取り出すと、内部ロードメモリは空になります。

CPU を RUN モードに切り替える前に、転送カードまたはプログラムカードを取り出す必要があります。

動作モードへのパスワードの使用の影響

次の表に、「パスワードの適正化の制限および有効化」命令によるパスワードの使用が動作モードと対応するユーザーの操作に与える影響を示します。

操作	命令によるパスワードの保護
その後の基本状態 • STOP への動作モードの切り替え • メモリの手動リセット(PG、切り替え、MC (モーションコントロール)の変更) • 工場出荷時設定へのリセット	無効 (制限なし)
電源オン後の基本状態	• S7-1200-CPU: ロックは無効で、プログラム(たとえば、スタートアップ OB)で再びこの命令を呼び出す必要があります。 • S7-1500 CPU: 有効(電源オフの前にロックが有効になった場合). パスワードの不許可のオプションは保持型です。
動作モードの移行 RUN/STARTUP/HOLD -> STOP (命令、エラーまたは通信の終了によって)または STOP -> STARTUP/RUN/HOLD	有効 パスワードはまだ使用できません。

構文

「パスワード正当性の制限および有効化命令には、以下の構文を使用します。

```
SCL
ENDIS_PW(REQ := <Operand>,
         F_PWD := <Operand>,
         FULL_PWD := <Operand>,
         R_PWD := <Operand>,
         HMI_PWD := <Operand>,
         F_PWD_ON => <Operand>,
         FULL_PWD_ON => <Operand>,
         R_PWD_ON => <Operand>,
         HMI_PWD_ON => <Operand>)
```

パラメータ

次の表に、「パスワードの適正化の制限および有効化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	REQ パラメータのシグナル状態が「0」の場合は、現在設定されているパスワードのシグナル状態が照会されます。

F_PWD	Input	BOOL	I、Q、M、D、L	フェールセーフなどの読み取り/書き込みアクセス • F_PWD = "0": パスワードを許可しない • F_PWD = "1": パスワードを許可する
FULL_PWD	Input	BOOL	I、Q、M、D、L	読み取り/書き込みアクセス • FULL_PWD = "0": パスワードを許可しない • FULL_PWD = "1": パスワードを許可する
R_PWD	Input	BOOL	I、Q、M、D、L	読み取りアクセス • R_PWD = "0": パスワードを許可しない • R_PWD = "1": パスワードを許可する
HMI_PWD	Input	BOOL	I、Q、M、D、L	HMI アクセス • HMI_PWD = "0": パスワードを許可しない • HMI_PWD = "1": パスワードを許可する
F_PWD_ON	Output	BOOL	I、Q、M、D、L	フェールセーフなどの読み取り/書き込みアクセスステータス • F_PWD_ON = "0": パスワードを許可しない • F_PWD_ON = "1": パスワードを許可する
FULL_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取り/書き込みアクセスステータス • FULL_PWD_ON = "0": パスワードを許可しない • FULL_PWD_ON = "1": パスワードを許可する
R_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取りアクセスステータス • R_PWD_ON = "0": パスワードを許可しない • R_PWD_ON = "1": パスワードを許可する
HMI_PWD_ON	Output	BOOL	I、Q、M、D、L	HMI アクセスステータス • HMI_PWD_ON = "0": パスワードを許可しない • HMI_PWD_ON = "1": パスワードを許可する
ファンクション値 (RET_VAL)		WORD	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8090	「パスワードの適正化の制限および有効化」命令はサポートされていません
80D0	フェールセーフのパスワードが未設定です。 標準 CPU では、シグナル状態は TRUE であることが必要です。
80D1	読み取り/書き込みアクセスが未設定です
80D2	読み取りアクセスが未設定です
80D3	HMI アクセスが未設定です
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ENDIS_PW(REQ := 0,
                         F_PWD := 0,
                         FULL_PWD := 0,
                         R_PWD := 1,
                         HMI_PWD := 0,
                         F_PWD_ON => "Status_F_PWD",
                         FULL_PWD_ON => "Sta-
tus_FULL_PWD",
                         R_PWD_ON => "Status_R_PWD",
                         HMI_PWD_ON => "Sta-
tus_HMI_PWD");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	データタイプ	値
REQ	BOOL	0
F_PWD	BOOL	0
FULL_PWD	BOOL	0
R_PWD	BOOL	1
HMI_PWD	BOOL	0
Status_F_PWD	BOOL	0
Status_FULL_PWD	BOOL	0

Status_R_PWD	BOOL	1
Status_HMI_PWD	BOOL	0

REQ オペランドのシグナル状態が「0」であるため、この命令が実行されます。 R_PWD オペランドのシグナル状態は「1」です。つまり、割り当て済みのパスワードを入力すると、読み取りアクセスが許可されます。 R_PWD_ON ステータスオペランドのシグナル状態も「1」です。これにより、R_PWD オペランドが有効であることを知らせます。

RE_TRIGR: サイクルタイムモニタのリスタート

説明

「サイクルタイムモニタのリスタート」命令は、CPU の最大サイクルタイムを再起動します。 サイクルモニタリングタイムが、CPU の設定で設定した時間で再起動されます。

「サイクルタイムモニタのリスタート」命令は、すべてのブロック内で優先度に関わらず呼び出し可能です。

ハードウェア割り込み、診断割り込み、または周期割り込みなどの上位の優先度のブロック内でこの命令が呼び出される場合、この命令は実行されません。

「サイクルタイムモニタのリスタート」命令は、呼び出しの数に関係なく期間内に完全に実行されます (最大プログラムサイクルの 10 倍)。 時間が期限切れになると、プログラムサイクルは延長できなくなります。

構文

「サイクルタイムモニタのリスタート」命令には、以下の構文を使用します。

```
SCL
RE_TRIGR()
```

パラメータ

「サイクルタイムモニタのリスタート」命令にはパラメータがなく、エラー情報は提供されません。

STP: プログラム終了

説明

「プログラムの終了」命令を使用し、CPU を STOP モードに設定してプログラムの実行を終了します。 RUN から STOP への変更の影響は CPU の構成によって異なります。

構文

以下の構文が「プログラムの終了」命令に使用されます。

```
SCL
STP()
```

GET_ERROR: ローカルでエラーを取得

説明

「ローカルでエラーを取得」命令は、ブロック内のエラーの発生の照会に使用されます。 これは、通常、アクセスエラーによって生じます。 システムがブロックの処理中にエラーを報告した場合、命令の最後の実行の後、このエラーがブロックの実行中に発生した最初の該当エラーの場合に詳細情報が生成されます。

エラー情報は、「ErrorStruct」システムデータタイプのオペランド内にのみ保存可能です。 「ErrorStruct」システムデータタイプは、エラー情報が保存されている正確な構造体を指定します。 追加の命令を使用することで、この構造体とプログラムが適切な応答をするかを評価できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーに関するエラー情報が出力されます。

注記

<Operand>は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、このオペランドを「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでこのオペランドを宣言する。
- 命令を呼び出す前に、このオペランドを「0」にリセットする。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記

「ローカルでエラーを取得」命令は、ブロック内のローカルエラー処理を可能にします。 「ローカルでエラーを取得」がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

構文

「ローカルでエラーを取得」命令には、以下の構文を使用します。

SCL
```
GET_ERROR(<Operand>)
```

パラメータ

次の表に、「ローカルでエラーを取得」命令のパラメータを示します。

パラメータ	データタイプ	メモリ領域	説明
<Operand>	ErrorStruct	D、L	発生したエラーに関する情報

データタイプ「ErrorStruct」

次の表に、「ErrorStruct」データタイプの構造体を示します。

構造体コンポーネント		データタイプ	説明
ERROR_ID		WORD	エラー ID
FLAGS		BYTE	エラーがブロック呼び出し中に発生したのかどうかを示します。 16#01: ブロック呼び出し中のエラー 16#00: ブロック呼び出し中にエラーなし
REACTION		BYTE	既定の応答 0: 無視(書き込みエラー) 1: 代替値「0」で続行(読み出しエラー) 2: 命令をスキップ(システムエラー)
CODE_ADDRESS		CREF	ブロックのアドレスおよびタイプに関する情報
	BLOCK_TYPE	BYTE	エラーが発生したブロックのタイプ: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	プログラムブロックの番号
	OFFSET	UDINT	内部メモリへの参照
MODE		BYTE	オペランドのアドレスに関する情報
OPERAND_NUMBER		UINT	マシンコマンドのオペランド番号
POINTER_NUMBER_LOCATION		UINT	(A) 内部ポインタ
SLOT_NUMBER_SCOPE		UINT	(B) 内部メモリの記憶領域
DATA_ADDRESS		NREF	オペランドのアドレスに関する情報
	AREA	BYTE	(C) メモリ領域 L: 16#40～4E、86、87、8E、8F、C0～CE I: 16#81 Q: 16#82 M: 16#83 DB: 16#84、85、8A、8B データタイプ DINT の直接編集可能なタグの範囲違反: 16#04
	DB_NUMBER	UINT	(D) データブロックの番号
	OFFSET	UDINT	(E) オペランドの相対アドレス

構造体コンポーネント「ERROR_ID」

次の表に、構造体コンポーネント「ERROR_ID」で出力可能な値を示します。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、またはファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、またはファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー: 入力
2943	10563	書き込みエラー: 出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

例

次の例で、命令がどのように動作するかを示します。

```
SCL
LABEL: #TagOut := #Field[#index] * REAL#40.5;

GET_ERROR(#Error);

IF #Error.REACTION = 1 THEN
   GOTO LABEL;
   ;
END_IF;
```

#Field[#index]タグへのアクセスでエラーが発生しました。 #TagOut オペランドは、読み取り/アクセスエラーに関わらずシグナル状態「1」を返し、乗算は値「0.0」によって実行されます。 I このエラーシナリオが発生する場合は、エラーを取得するために、乗算の後に、「ローカルでエラーを取得」命令をプログラムすることを推奨します。 「ローカルでエラーを取得」命令によって供給されるエラー情報は、比較を使用して評価されます。 #Error.REACTION 構造コンポーネントの値が「1」の場合、読み取り/アクセスエラーが発生しており、プログラム実行はジャンプラベル LABEL で再び開始します。

GET_ERR_ID: ローカルでエラー ID を取得

説明

「ローカルでエラー ID を取得」命令は、ブロック内のエラーの発生の照会に使用されます。 これは、通常、アクセスエラーによって生じます。 この命令の最後の実行の後に、システムがブロック処理中にブロック実行に関するエラーを報告すると、この命令は発生した最初のエラーのエラー ID を出力します。

エラー ID は WORD データタイプのオペランド内にのみ保存可能です。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーのエラー ID が出力されます。

注記
<Operand>は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、このオペランドを「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでこのオペランドを宣言する。
- 命令を呼び出す前に、このオペランドを「0」にリセットする。

「ローカルでエラー ID を取得」命令の出力は、エラー情報が存在する場合のみ設定されます。 これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラー ID を取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラー ID を取得」命令は、ブロック内のローカルエラー処理を可能にします。「ローカルでエラー ID を取得」命令がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

構文

「ローカルでエラー ID を取得」命令には、以下の構文を使用します。

```
SCL
GET_ERR_ID()
```

パラメータ

次の表に、「ローカルでエラー ID を取得」命令のパラメータを示します。

パラメータ	データタイプ	メモリ領域	説明
ファンクション値	WORD	D、L	エラー ID

エラー ID

次の表に、出力として考えられる値を示します。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、またはファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、またはファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー: 入力
2943	10563	書き込みエラー: 出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

例

次の例で、命令がどのように動作するかを示します。

```
SCL
#TagOut := #Field[#index] * REAL#40.5;
```

```
#TagID := GET_ERR_ID();

IF #TagID = 16#2522 THEN
   MOVE_BLK(IN := #TagArrayIn[0],
            COUNT := 1,
            OUT => #TagArrayOut[1]);
END_IF;
```

#Field[#index]タグへのアクセスでエラーが発生しました。 #TagOut オペランドは、読み取り/アクセスエラーに関わらずシグナル状態「1」を返し、乗算は値「0.0」によって実行されます。 I このエラーシナリオが発生する場合は、エラーを取得するために、乗算の後に、「ローカルでエラー ID を取得」命令をプログラムすることを推奨します。 「ローカルでエラー ID を取得」命令によって供給されるエラー ID は、比較を使用して評価されます。 #TagID オペランドが ID 16#2522 を返す場合、読み取り/アクセスエラーが発生しており、#TagArrayIn[0]オペランドの値「100.0」が#TagArrayOut[1]オペランドに書き込まれます。

INIT_RD: すべての保持データの初期化

説明

「すべての保持データを初期化」命令を使用すると、すべてのデータブロック、ビットメモリ、および SIMATIC タイマとカウンタの保持データが同時にリセットされます。 この命令は、実行するとプログラムサイクル持続時間を超過するため、スタートアップ OB 内でのみ実行が可能です。

構文

「すべての保持データを初期化」命令には、以下の構文を使用します。

SCL
```
INIT_RD(<Operand>)
```

パラメータ

次の表に、「すべての保持データを初期化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	BOOL	I、Q、M、D、L	入力「REQ」のシグナル状態が「1」の場合、すべての保持データがリセットされます。
ファンクション値 (RET_VAL)		INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B5	この命令はスタートアップ OB 内にプログラムされていないため、実行できません。
一般エラー情報	関連項目:"GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := INIT_RD("Tag_REQ");
```

オペランド「Tag_REQ」のシグナル状態が「1」の場合、この命令が実行されます。 すべてのデータブロック、ビットメモリ、および SIMATIC タイマの保持データとカウンタがリセットされます。

WAIT: 遅延時間の設定

説明

「遅延時間の設定」命令は、指定された時間の間プログラム実行を一時停止します。 この命令の WT パラメータに、マイクロ秒で時間を指定します。

遅延時間は、-32768 から最大 32767 マイクロ秒 (µs)が設定可能です。設定可能な最短の遅延時間は個々の CPU に依存し、「遅延時間の設定」命令の実行時間に対応します。

この命令の実行には、さらに優先度の高いイベントが割り込む可能性があります。

「遅延時間の設定」命令はエラー情報を返しません。

構文

以下の構文が「遅延時間の設定」命令に使用されます。

```
SCL
WAIT(WT:= <Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
WT	Input	INT	I、Q、M、D、L、P	遅延時間(µs)

ランタイム: プログラムランタイムの測定

説明

「プログラムランタイムの測定」命令を使用して、プログラム全体、個々のブロック、またはコマンドシーケンスのランタイムを測定します。

プログラム全体のランタイムを測定する場合は、「プログラムランタイムの測定」命令を OB1 で呼び出します。 最初の呼び出しでランタイムの測定が開始され、2 回目の呼び出し後に出力 RET_VAL がプログラムのランタイムを返します。 測定されたランタイムには、たとえば上位レベルのイベントや通信による割り込みなど、プログラム実行中に発生する可能性のあるすべての CPU プロセスが含まれます。 「プログラムランタイムの測定」命令は CPU の内部カウンタを読み取り、値を IN-OUT パラメータに書き込みます。 この命令は、内部カウンタ周波数に基づいて現在のプログラムランタイムを計算し、出力 RET_VAL に書き込みます。

個々のブロック、または個々のコマンドシーケンスのランタイムを測定する場合、3 つの個別のネットワークが必要です。 プログラム内の個々のネットワークで、「プログラムランタイムの測定」命令を呼び出します。 命令のこの最初の呼び出しでランタイム測定の開始位置を設定します。 その後、次のネットワーク内で目的のプログラムブロック、またはコマンドシーケンスを呼び出します。 他のネットワーク内で、「プログラムランタイムの測定」命令の 2 回目の呼び出しを行い、命令の最初の呼び出しで行ったように同じメモリを IN/OUT パラメータに割り当てます。 3 番目のネットワークの「プログラムランタイムの測定」命令は内部 CPU カウンタを読み取り、内部カウンタ周波数に基づいてプログラムブロック、またはコマンドシーケンスの現在のランタイムを計算し、それを出力 RET_VAL に書き込みます。

以下は、ファームウェアバージョン V4.1 未満の S7-1200 が対象です。 「プログラムランタイムの測定」命令は、内部の高周波数カウンタを使用して時間を計算します。 カウンタがオーバーフローする場合(これは 1 分当たり 1 回発生する可能性があります)、命令は値<= 0.0 を返します。 これらのランタイム値は無視してください。

注記
コマンドシーケンスのランタイムは、正確に測定することはできません。これは、コマンドシーケンス内の命令のシーケンスが、プログラムを最適にコンパイルする過程で変更されるためです。

構文

以下の構文が「プログラムランタイムの測定」命令に使用されます。

```
SCL
RUNTIME(<Operand>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	InOut	LREAL	I、Q、M、D、L	ランタイム測定の開始位置を保存します。
ファンクション値		LREAL	I、Q、M、D、L	測定したランタイムを秒単位で返します

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がプログラムブロックのランタイム計算に基づいてどのように動作するかを示します。

```
SCL
"Tag_Result" := RUNTIME("Tag_Memory");

"Best_before_date_DB" ();

"Tag_Result" := RUNTIME("Tag_Memory");
```

この命令の最初の呼び出しでランタイムの測定が開始され、TagMemory オペランドでの命令の 2 回目の呼び出しの参照先としてバッファリングされます。

「Best_before_date」プログラムブロック FB1 が呼び出されます。

プログラムブロック FB1 が処理されると、命令の 2 回目の実行が行われます。 命令の 2 回目の呼び出しにより、プログラムブロックのランタイムが計算され、結果が出力 Tag_Result に書き込まれます。

ワード論理演算

この章には下記に関する情報が記載されています：

- DECO: デコード (S7-1200, S7-1500)
- ENCO: エンコード (S7-1200, S7-1500)
- SEL: 選択 (S7-1200, S7-1500)
- MUX: 多重化 (S7-1200, S7-1500)
- DEMUX: 逆多重化 (S7-1200, S7-1500)

DECO: デコード

説明

「デコード」命令は、入力値で指定されたビットを出力値にセットします。

「デコード」命令は、IN パラメータの値を読み取り、読み取った値とビット位置が一致する出力値にそのビットをセットします。 出力値の他のビットはゼロで埋められます。 IN パラメータの値が 31 よりも大きい場合は、モジュロ 32 命令が実行されます。

構文

「デコード」命令には、以下の構文を使用します。

SCL

```
DECO(IN := <Expression>)
DECO_WORD(IN := <Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	UINT	I、Q、M、D、L、P	設定される出力値のビット位置
_<Data type>		ビット列 既定値: DWORD	-	ファンクション値のデータタイプ: 1. 「_」を使用して、この命令のデータタイプを明示的に指定することができます。 2. データタイプを明示的に指定しないと、使用されるタグまたはタイプコード化された定数によって指定されます。 3. データタイプを明示的に指定せず、定義済みタグまたはタイプコード化された定数も指定しない場合は、既定のデータタイプが使用されます。
ファンクション値		ビット列	I、Q、M、D、L、P	現在の出力値

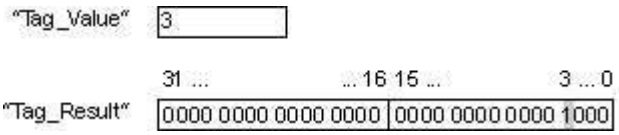
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := DECO(IN := "Tag_Value");
"Tag_Result2" := DECO_BYTE(IN := "Tag_Value2");
```

次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



この命令はオペランド「Tag_Value」の値から番号「3」を読み取り、3 番目のビットをオペランド「Tag_Result」の値にセットします。

ENCO: エンコード

説明

「エンコード」命令は、入力値の最小値ビットセットのビット番号を読み取り、これを結果として出力します。

構文

「エンコード」命令には、以下の構文を使用します。

```
SCL
ENCO(IN := <Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	ビット列	I、Q、M、D、L、P	入力値
ファンクション値		INT	I、Q、M、D、L、P	読み出される入力値のビットのビット番号。

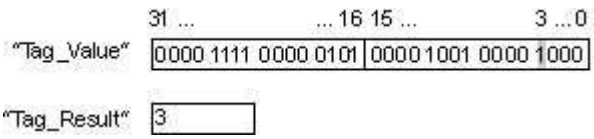
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ENCO(IN := "Tag_Value");
```

次の図に、命令が特定のオペランド値を使用してどのように動作するかを示します。



この命令はオペランド「Tag_Value」のビットにセットされた最小値を読み取り、オペランド「Tag_Result」にビット位置「3」を書き込みます。

SEL: 選択

説明

「選択」命令は、スイッチ(G パラメータ)に応じてパラメータ IN0 または IN1 のいずれかを選択し、その内容を結果として出力します。 G パラメータのシグナル状態が「0」の場合、IN0 パラメータの値は移動されます。 G パラメータのシグナル状態が「1」の場合、IN1 パラメータの値は移動され、ファンクション値として返されます。

この命令は、すべてのパラメータのタグが同じデータタイプクラスの場合にのみ実行されます。

構文

「選択」命令には、以下の構文を使用します。

```
SCL
SEL(G  := <Expression>,
    IN0  := <Expression>,
    IN1  := <Expression>)
```

命令の構文は以下の部分で構成されます。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
G	Input	BOOL	BOOL	I、Q、M、D、L	スイッチ
IN0	Input	2 進数、整数、浮動小数点数、タイマ、文字列、TOD、DATE、DT	2 進数、整数、浮動小数点数、タイマ、文字列、DATE、TOD、LTOD、DT、LDT	I、Q、M、D、L、P	最初の入力値
IN1	Input	2 進数、整数、浮動小数点数、タイマ、文字列、TOD、DATE、DT	2 進数、整数、浮動小数点数、タイマ、文字列、DATE、TOD、LTOD、DT、LDT	I、Q、M、D、L、P	2 番目の入力値
ファンクション値		2 進数、整数、浮動小数点数、タイマ、文字列、TOD、DATE、DT	2 進数、整数、浮動小数点数、タイマ、文字列、DATE、TOD、LTOD、DT、LDT	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := SEL(G := "Tag_Value",
                    IN0 := "Tag_0",
                    IN1 := "Tag_1");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_Value	0	1
Tag_0	W#16#0000	W#16#4C
Tag_1	W#16#FFFF	D#16#5E
Tag_Result	W#16#0000	D#16#5E

MUX: 多重化

説明

「多重化」命令は、選択された入力パラメータの値をコピーし、それを出力します。 K パラメータを使用して、値が移動される入力パラメータの番号を識別できます。 番号の割り当ては IN0 から開始され、新しい入力ごとに継続的にインクリメントされます。 最大 32 の入力を宣言できます。

入力には、数値データタイプおよび時間データタイプが許可されています。 パラメータが割り当てられたすべてのタグは、データタイプが同じであることが必要です。

次のいずれかの条件が満たされる場合、ファンクション値は無効です。

- 命令の実行中にエラーが発生していること。
- K パラメータの入力は、使用可能な入力の外側にあります。 INELSE 入力を使用しない場合、IN0 入力の値がこの場合ファンクション値に割り当てられ、ENO 許可出力が「0」にセットされます。

構文

「多重化」命令には、以下の構文を使用します。

```
SCL
MUX(K  := <Expression>,
    IN0  := <Expression>,
    IN1  := <Expression>,
    INELSE  := <Expression>)
```

パラメータ

次の表に、「多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
K	Input	整数	整数	I、Q、M、D、L、P	内容が転送されるパラメータを指定します。 • K = 0 ならば、パラメータ IN0 • K = 1 ならば、パラメータ IN1 など
IN0	Input	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、TOD、LTOD、DATE、タイマ、DT、LDT	I、Q、M、D、L、P	最初の入力値

IN1	Input	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、TOD、LTOD、DATE、タイマ、DT、LDT	I、Q、M、D、L、P	2 番目の入力値
INn	Input	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、TOD、LTOD、DATE、タイマ、DT、LDT	I、Q、M、D、L、P	オプションの入力値
INELSE	Input	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、TOD、LTOD、DATE、タイマ、DT、LDT	I、Q、M、D、L、P	K <> n の場合コピーする値を指定します。
ファンクション値		2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、TOD、LTOD、DATE、タイマ、DT、LDT	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := MUX(K := "Tag_Number",
                    IN0 := "Tag_1",
                    IN1 := "Tag_2",
                    INELSE := "Tag_3");
```

命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	**値**	
Tag_Number	1	4
Tag_1	DW#16#00000000	DW#16#00000000

Tag_2	DW#16#003E4A7D	DW#16#003E4A7D
Tag_3	DW#16#FFFF0000	DW#16#FFFF0000
Tag_Result	DW#16#003E4A7D	DW#16#FFFF0000

DEMUX: 逆多重化

説明

「逆多重化」命令は、入力パラメータ IN の値を選択された出力パラメータに転送します。 入力パラメータの選択は、パラメータ値 K とは別に行われます。 K パラメータは、入力パラメータ IN の値が転送される出力パラメータの数を指定します。 他の出力パラメータは変更されません。 番号の割り当ては OUT0 から開始され、新しい出力ごとに継続します。 最大 32 の出力パラメータを宣言できます。

K パラメータの値が出力パラメータの数より大きい場合、ENO 許可出力の値が「0」にセットされ、入力パラメータ IN の値が出力パラメータ OUTELSE に転送されます。

次のいずれかの条件が満たされる場合、ファンクション値は無効です。

- K パラメータの値が、使用可能な出力の数を超えていること。
- 命令の実行中にエラーが発生していること。

構文

「逆多重化」命令には、以下の構文を使用します。

```
SCL
DEMUX(K  := <Expression>,
      IN  := <Expression>,
      OUT0  := <Operand>,
      OUT1  := <Operand>,
      OUTELSE  := <Operand>)
```

パラメータ

次の表に、「逆多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
K	Input	整数	整数	I、Q、M、D、L、P	入力値(IN)がコピーされる出力を指定します。 • K = 0 ならば、パラメータ OUT0 • K = 1 ならば、パラメータ OUT1 など
IN	Input	2 進数、整数、浮動小数点数、	2 進数、整数、浮動小数	I、Q、M、D、L、P	入力値

		タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	点数、文字列、タイマ、TOD、LTOD、DATE、DT、LDT		
OUT0	Output	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、タイマ、TOD、LTOD、DATE、DT、LDT	I、Q、M、D、L、P	最初の出力
OUT1	Output	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、タイマ、TOD、LTOD、DATE、DT、LDT	I、Q、M、D、L、P	2 番目の出力
OUTn	Output	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、タイマ、TOD、LTOD、DATE、DT、LDT	I、Q、M、D、L、P	オプションの出力
OUTELSE	Output	2 進数、整数、浮動小数点数、タイマ、STRING、CHAR、WCHAR、TOD、DATE、DT	2 進数、整数、浮動小数点数、文字列、タイマ、TOD、LTOD、DATE、DT、LDT	I、Q、M、D、L、P	K > n の場合、入力 IN の値のコピー先となる出力。

使用可能なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
DEMUX(K := "Tag_Number",
      IN := "Tag_Value",
      OUT0 := "Tag_1",
      OUT1 := "Tag_2",
      OUTELSE := "Tag_3");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

ネットワーク実行前の「逆多重化」命令の入力値

パラメータ	オペランド	値	
K	Tag_Number	2	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#003E4A7D

ネットワーク実行後の「逆多重化」命令の出力値

パラメータ	オペランド	値	
OUT0	Tag_1	不変	不変
OUT1	Tag_2	DW#16#FFFFFFFF	不変
OUTELSE	Tag_3	不変	DW#16#003E4A7D

シフトとローテーション

この章には下記に関する情報が記載されています：

- SHR: 右へシフト (S7-1200, S7-1500)
- SHL: 左へシフト (S7-1200, S7-1500)
- ROR: 右へローテーション (S7-1200, S7-1500)
- ROL: 左へローテーション (S7-1200, S7-1500)

SHR: 右へシフト

説明

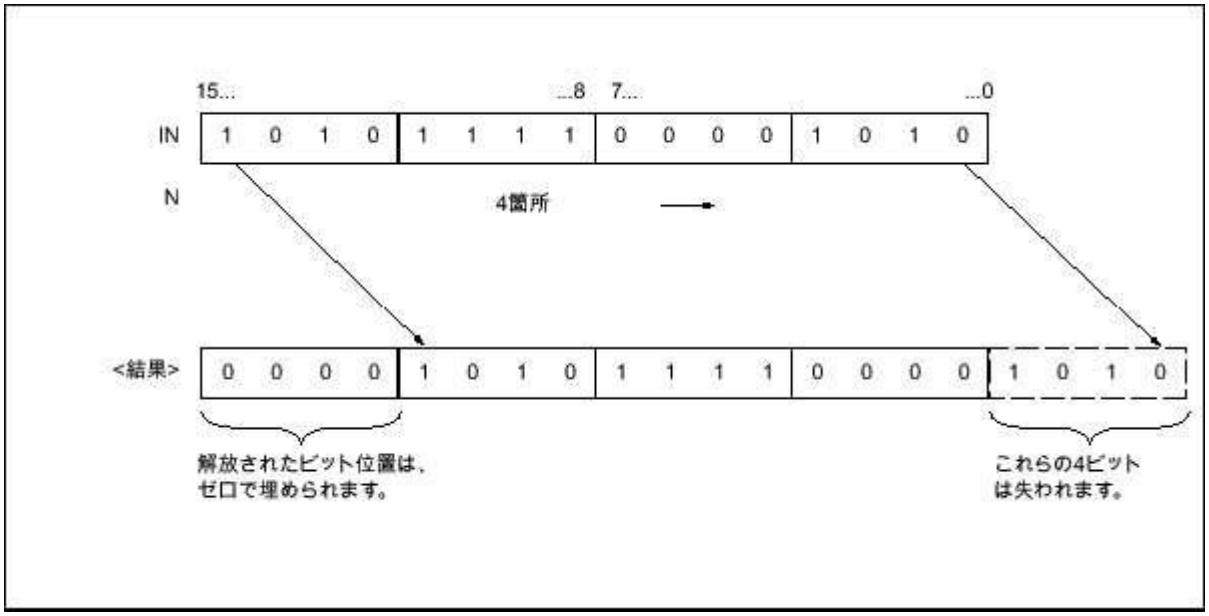
「右へシフト」命令は、IN パラメータの内容を 1 ビットずつ右へシフトし、ファンクション値として返します。 N パラメータは、指定した値をシフトするビット位置の数の指定に使用されます。

N パラメータの値が「0」の場合、IN パラメータの値が結果として与えられます。

N パラメータの値が使用可能なビット桁数よりも大きい場合、IN パラメータの値は使用可能なビット桁数だけ右にシフトされます。

シフトによって解放された左オペランド領域のビット位置は、ゼロで埋められます。

次の図に、整数データタイプのオペランドが、どのように 4 ビット位置右へシフトされるかを示します。



構文

「右へシフト」命令には、以下の構文を使用します。

```
SCL
SHR(IN := <Operand>,
    N := <Operand>)
```

パラメータ

次の表に、「右へシフト化」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L	シフトされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L	値(IN)がシフトされるビットの数
ファンクション値		ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := SHR(IN := "Tag_Value",
                    N := "Tag_Number");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	3
ファンクション値	Tag_Result	0000 0111 1111 010 1

オペランド「Tag_Value」の内容が、3 ビット位置右へシフトされます。 命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

SHL: 左へシフト

説明

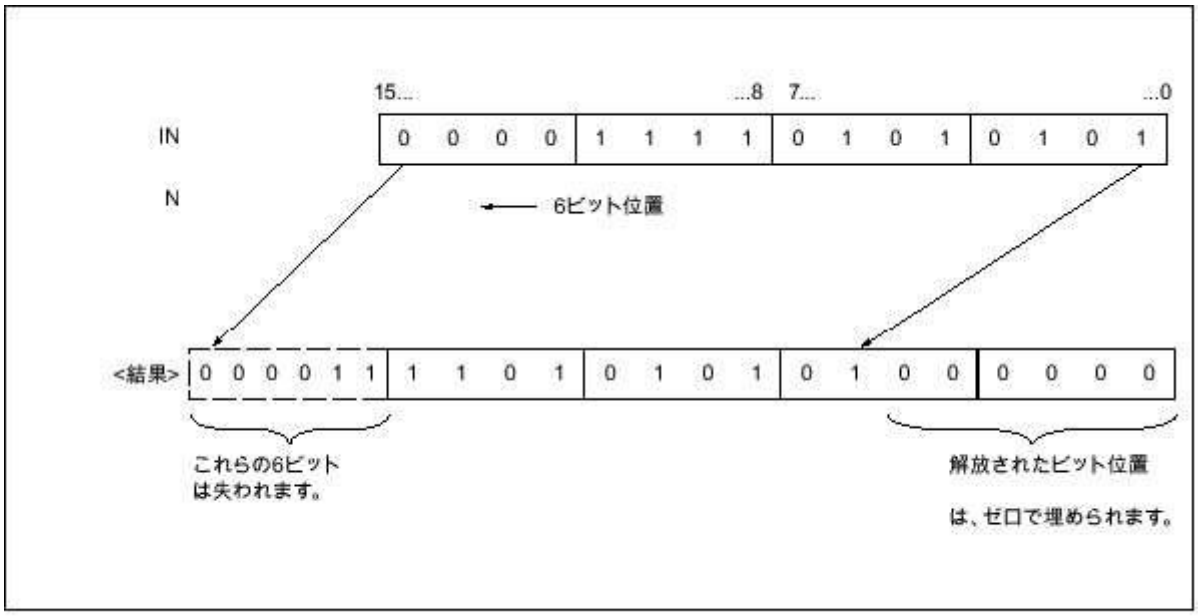
「左へシフト」命令は、IN パラメータの内容を 1 ビットずつ左へシフトし、ファンクション値として返します。 N パラメータは、指定した値をシフトするビット位置の数の指定に使用されます。

N パラメータの値が「0」の場合、IN パラメータの値が結果として与えられます。

N パラメータの値がビット位置の数よりも大きい場合、IN パラメータの値は、使用可能なビット位置の数だけ左へシフトされます。

結果では、シフトによって空いたビット位置がゼロで埋められます。

次の図に、WORD データタイプのオペランドの内容が、どのように 6 ビット位置左へシフトされるかを示します。



構文

「左へシフト」命令には、以下の構文を使用します。

```
SCL
SHL(IN := <Operand>,
    N := <Operand>)
```

パラメータ

次の表に、「左へシフト」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L	シフトされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L	値(IN)がシフトされるビットの数
ファンクション値		ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := SHL(IN := "Tag_Value",
                    N := "Tag_Number");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	0011 1111 1010 1111
N	Tag_Number	4
ファンクション値	Tag_Result	1111 1010 1111 0000

オペランド「Tag_Value」の値は、4 ビット位置左へシフトされます。 命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

ROR: 右へローテーション

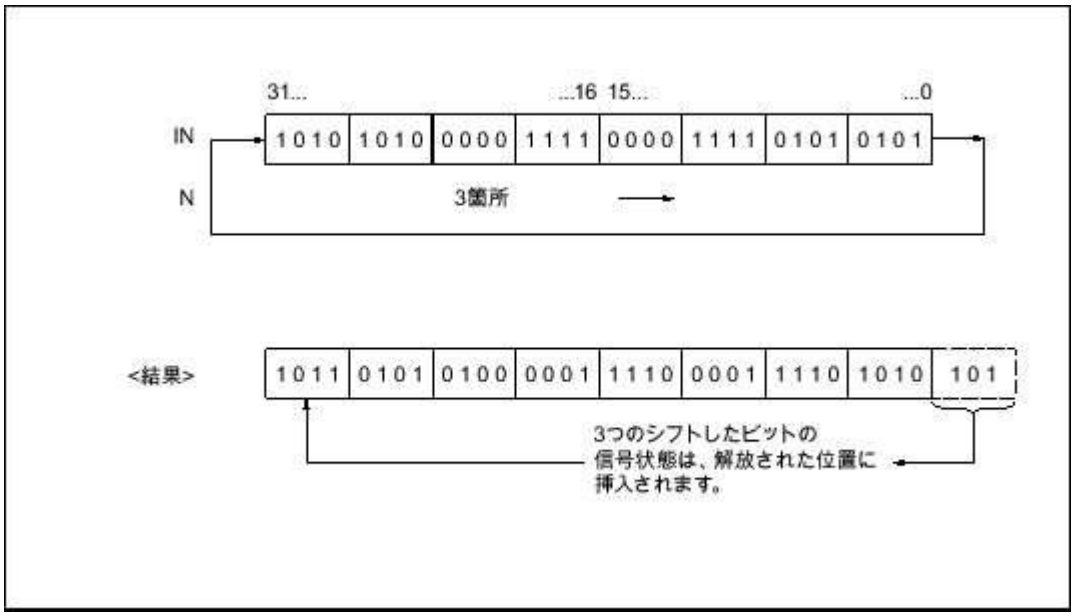
説明

「右へローテーション」命令は、IN パラメータの内容を 1 ビットずつ右へローテーションし、指定されたオペランドに結果を割り当てます。 N パラメータは、指定した値をローテーションするビット位置の数の指定に使用されます。 ローテーションによって解放されたビット位置は、押し出されたビット位置で埋められます。

N パラメータの値が「0」の場合、入力 IN の値が結果として与えられます。

使用可能なビット位置の数よりも N パラメータの値が大きい場合でも、IN 入力にあるオペランド値は、指定されたビット位置の数でローテーションされます。

次の図に、DWORD データタイプのオペランドの内容が、どのように 3 ビット位置右へローテーションされるかを示します。



構文

「右へローテーション」命令には、以下の構文を使用します。

```
SCL
ROR(IN := <Operand>,
    N := <Operand>)
```

パラメータ

次の表に、「右へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L	ローテーションされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L	値(IN)がローテーションされるビット位置の数
ファンクション値		ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ROR(IN := "Tag_Value",
                    N := "Tag_Number");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	0000 1111 1001 0101
N	Tag_Number	5
ファンクション値	Tag_Result	1010 1000 0111 1100

オペランド「Tag_Value」の内容は、5 ビット位置右へローテーションされます。 命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

ROL: 左へローテーション

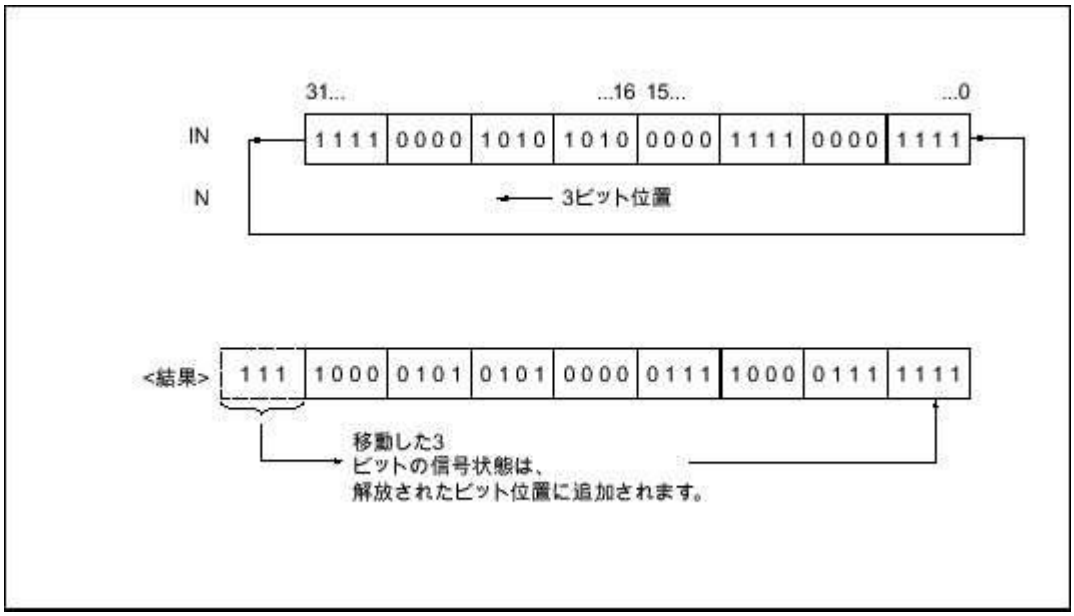
説明

「左へローテーション」命令は、IN パラメータの内容を 1 ビットずつ左へローテーションし、ファンクション値として返します。 N パラメータは、指定した値をローテーションするビット位置の数の指定に使用されます。 ローテーションによって解放されたビット位置は、押し出されたビット位置で埋められます。

N パラメータの値が「0」の場合、入力 IN の値が結果として与えられます。

使用可能なビット位置の数よりも N パラメータの値が大きい場合でも、IN 入力にあるオペランド値は、指定されたビット位置の数でローテーションされます。

次の図に、DWORD データタイプのオペランドの内容が、どのように 3 ビット位置左へローテーションされるかを示します。



構文

「左へローテーション」命令には、以下の構文を使用します。

```
SCL
ROL(IN := <Operand>,
    N := <Operand>)
```

パラメータ

次の表に、「左へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	ビット列、整数	ビット列、整数	I、Q、M、D、L	ローテーションされる値
N	Input	USINT, UINT, UDINT	USINT, UINT, UDINT, ULINT	I、Q、M、D、L	値(IN)がローテーションされるビット位置の数
ファンクション値		ビット列、整数	ビット列、整数	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := ROL(IN := "Tag_Value",
                    N := "Tag_Number");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	Tag_Value	1010 1000 1111 0110
N	Tag_Number	5
ファンクション値	Tag_Result	0001 1110 1101 0101

オペランド「Tag_Value」の内容は、左に 5 ビット位置ローテーションされます。 命令の結果は、ファンクション値としてオペランド「Tag_Result」内に返されます。

レガシー

この章には下記に関する情報が記載されています：

DRUM: PLC 実行

説明

「PLC の実装」命令を使用して、対応するステップの OUT_VAL パラメータのプログラム済みの値をプログラム済み出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)に割り当てます。 そのため、この命令が特定のステップに留まっている間は、このステップは S_MASK パラメータでプログラムされた許可マスクの条件を満たす必要があります。 このステップのイベントが真(TRUE)になっていて現在のステップのプログラムされた時間が経過した場合、または JOG パラメータの値が「0」から「1」に切り替わった場合、この命令は次のステップに進みます。 RESET パラメータのシグナル状態が「1」に切り替わると、この命令はリセットされます。 これによって、プリセットされたステップ(DSP)と現在のステップが等しくなります。

このステップで消費された時間の量は、プリセットタイムベース(DTBP)と各ステップのプリセットカウンタ値(S_PRESET)の積によって計算されます。 新しいステップの開始時に、この計算値が DCC パラメータにロードされます。このパラメータには、現在のステップの残り時間が含まれています。 たとえば、DTBP パラメータの値が「2」で、最初のステップのプリセット値が「100」(100 ミリ秒)の場合、DCC パラメータの値は「200」(200 ミリ秒)になります。

ステップは、時間値、イベント、またはその両方を使用してプログラム可能です。 イベントビットおよび時間値「0」を持つステップは、イベントビットのシグナル状態が「1」になると直ちに次のステップに進みます。 時間値のみを使用してプログラムされたステップは、時間を直ちに開始します。 「0」よりも大きなイベントビットおよび時間値を使用してプログラムされたステップは、イベントビットのシグナル状態が「1」になると時間を開始します。 イベントビットは、シグナル状態「1」で初期化されます。

プログラムされた最後のステップ(LST_STEP)の実行中にこのステップに割り当てられた時間が経過した場合、Q パラメータのシグナル状態は「1」にセットされ、それ以外の場合は「0」にセットされます。 Q パラメータがセットされると、この命令はリセットされるまでこのステップに留まります。

設定可能なマスク(S_MASK)で、出力ワード(OUT_WORD)の個々のビットを選択し、出力値(OUT_VAL)によって出力ビット(OUT1 ~ OUT16)をセットまたはリセットできます。 設定可能なマスクのビットのシグナル状態が「1」の場合、OUT_VAL 値は対応するビットをセットまたはリセットします。 設定可能なマスクビットのシグナル状態が「0」の場合、対応するビットは変更されません。 16 のステップすべてに設定可能なマスクのすべてのビットは、シグナル状態「1」で初期化されます。

OUT1 パラメータの出力ビットは、出力ワード(OUT_WORD)の最下位ビットに対応します。 OUT16 パラメータの出力ビットは、出力ワード(OUT_WORD)の最上位ビットに対応します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「PLC の実装」命令には、以下の構文を使用します。

SCL

```
<Instance>(RESET := <Operand>,
           JOG := <Operand>,
```

```
DRUM_EN := <Operand>,
LST_STEP := <Operand>,
EVENT1 - 16 := <Operand>,
OUT1 - 16 => <Operand>,
Q => <Operand>,
OUT_WORD => <Operand>,
ERR_CODE => <Operand>)
```

パラメータ

次の表に、「PLC 実装」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RESET	Input	BOOL	I、Q、M、D、L	シグナル状態「1」は、リセット条件を示します。
JOG	Input	BOOL	I、Q、M、D、L	シグナル状態が「0」から「1」に切り替わると、命令が次のステップに進みます。
DRUM_EN	Input	BOOL	I、Q、M、D、L	シグナル状態が「1」になると、PLC はイベントおよび時間条件に基づいて進むことができます。
LST_STEP	Input	BYTE	I、Q、M、D、L	プログラムされた最後のステップの番号。
EVENT(i), 1 ≤ i ≤ 16	Input	BOOL	I、Q、M、D、L	イベントビット(i); 初期シグナル状態は「1」。
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I、Q、M、D、L	出力ビット(j)
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
OUT_WORD	Output	WORD	I、Q、M、D、L、P	PLC が出力値を書き込むワードアドレス。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
JOG_HIS	Static	BOOL	I、Q、M、D、L	JOG パラメータの履歴ビット
EOD	Static	BOOL	I、Q、M、D、L	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
DSP	Static	BYTE	I、Q、M、D、L、P	PLC のプリセットされたステップ
DSC	Static	BYTE	I、Q、M、D、L、P	PLC の現在のステップ

DCC	Static	DWORD	I、Q、M、D、L、P	PLC のカウンタ現在値
DTBP	Static	WORD	I、Q、M、D、L、P	PLC のプリセットタイムベース
PrevTime	Static	TIME	I、Q、M、D、L	前のシステム時間
S_PRESET	Static	ARRAY[1..16] of WORD	I、Q、M、D、L	1 クロックパルス = 1 ミリ秒の場合の各ステップ[1 ~ 16]のプリセットされたカウンタ値。
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L	各ステップの出力値[1 ~ 16, 0 ~ 15]。
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L	各ステップの設定可能なマスク[1 ~ 16, 0 ~ 15]。 初期シグナル状態は「1」。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

ERR_CODE*	説明
W#16#0000	エラーは発生していません。
W#16#000B	LST_STEP パラメータの値が 1 未満であるか、または 16 を超えています。
W#16#000C	DSC パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
W#16#000D	DSP パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例で命令はステップ 1 からステップ 2 に進みます。出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)はステップ 2 に構成定義されたマスクおよび OUT_VAL パラメータの値によってセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

```
SCL
"DRUM_DB"(RESET := "Tag_Reset"
          JOG := "Tag_Input_Jog"
          DRUM_EN := "Tag_Input_DrumEN"
          LST_STEP := "Tag_Number_LastStep"
```

```
EVENT1 := "MyTag_Event_1"
EVENT2 := "MyTag_Event_2"
EVENT3 := "MyTag_Event_3"
EVENT4 := "MyTag_Event_4"
EVENT5 := "MyTag_Event_5"
EVENT6 := "MyTag_Event_6"
EVENT7 := "MyTag_Event_7"
EVENT8 := "MyTag_Event_8"
EVENT9 := "MyTag_Event_9"
EVENT10 := "MyTag_Event_10"
EVENT11 := "MyTag_Event_11"
EVENT12 := "MyTag_Event_12"
EVENT13 := "MyTag_Event_13"
EVENT14 := "MyTag_Event_14"
EVENT15 := "MyTag_Event_15"
EVENT16 := "MyTag_Event_16"
OUT1 => "MyTag_Output_1"
OUT2 => "MyTag_Output_2"
OUT3 => "MyTag_Output_3"
OUT4 => "MyTag_Output_4"
OUT5 => "MyTag_Output_5"
OUT6 => "MyTag_Output_6"
OUT7 => "MyTag_Output_7"
OUT8 => "MyTag_Output_8"
OUT9 => "MyTag_Output_9"
OUT10 => "MyTag_Output_10"
OUT11 => "MyTag_Output_11"
OUT12 => "MyTag_Output_12"
OUT13 => "MyTag_Output_13"
OUT14 => "MyTag_Output_14"
OUT15 => "MyTag_Output_15"
OUT16 => "MyTag_Output_16"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode");
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータの初期化には以下の値が使用されます。

パラメータ	オペランド	アドレス	値
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE

DRUM_EN	Tag_Input_DrumEN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

以下の値が命令の「DRUM_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE
OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE

OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE
S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

出力パラメータは、命令の実行前に以下の値にセットされます。

パラメータ	オペランド	アドレス	値
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE

OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	アドレス	値
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

命令の実行後、以下の値が命令のインスタンスデータブロック「DRUM_DB」で変更されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

DCAT: ディスクリート制御タイマアラーム

説明

「ディスクリート制御タイマアラーム」命令は、CMD パラメータが開くまたは閉じるためのコマンドを発行した時点からの時間を累積するために使用されます。 時間は、指定した時間内でプリセット時間(PT)を超えるか、またはデバイスの開閉に関する情報を受信するまで蓄積します(O_FB または C_FB)。 デバイスの開閉に関する情報を受信する前にプリセット済み時間が経過すると、対応するアラームが有効になります。 プリセット済み時間の前にコマンド入力のシグナル状態が切り替わると、この時間経過が再開します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「ディスクリート制御タイマアラーム」命令は入力条件に対して以下のように応答します。

- CMD パラメータのシグナル状態が「0」から「1」に切り替わると、パラメータ Q、CMD_HIS、ET(ET < PT の場合のみ)、OA および CA のシグナル状態が、以下のように影響を受けます。
  - Q および CMD_HIS パラメータが「1」にセットされます。
  - ET、OA および CA パラメータが「0」にリセットされます。
- パラメータ CMD のシグナル状態が「1」から「0」に切り替わると、パラメータ Q、ET (ET < PT の場合のみ)、OA、CA および CMD_HIS が「0」にリセットされます。
- CMD および CMD_HIS パラメータのシグナル状態が「1」で、O_FB パラメータが「0」にセットされると、命令が最後に実行されてからの時間差(ms)が ET パラメータの値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、OA パラメータのシグナル状態が「1」にセットされます。 ET パラメータの値が PT パラメータの値を超えない場合、OA パラメータがシグナル状態「0」にリセットされます。 CMD_HIS パラメータの値が CMD パラメータの値にセットされます。
- CMD、CMD_HIS および O_FB パラメータのシグナル状態が「1」にセットされていて C_FB パラメータの値が「0」の場合、OA パラメータがシグナル状態「0」にセットされます。 ET パラメータの値が PT パラメータの値にセットされます。 O_FB パラメータがシグナル状態「0」に切り替わると、命令の次回実行時にアラームがセットされます。 CMD_HIS パラメータの値が CMD パラメータの値にセットされます。
- CMD、CMD_HIS および C_FB パラメータの値が「0」の場合、命令が最後に実行されてからの時間差(ms)が ET パラメータの値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、CA パラメータのシグナル状態が「1」にセットされます。 PT パラメータの値を超えなければ、CA パラメータのシグナル状態は「0」です。 CMD_HIS パラメータの値が CMD パラメータの値にセットされます。
- CMD、CMD_HIS および O_FB パラメータのシグナル状態が「0」のときに C_FB パラメータが「1」にセットされると、CA パラメータが「0」にセットされます。 ET パラメータの値が PT パラメータの値にセットされます。 C_FB パラメータがシグナル状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 CMD_HIS パラメータの値が CMD パラメータの値にセットされます。
- O_FB および C_FB パラメータのシグナル状態が同時に「1」の場合、両方のアラーム出力のシグナル状態が「1」にセットされます。

「ディスクリート制御タイマアラーム」命令に工ラー情報がありません。

構文

「ディスクリート制御タイマアラーム」命令には、以下の構文を使用します。

```
SCL
<Instance>(CMD := <Operand>,
           O_FB := <Operand>,
           C_FB := <Operand>,
           Q => <Operand>,
           OA => <Operand>,
           CA => <Operand>)
```

パラメータ

次の表に、ブロックパラメータのタイプを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CMD	Input	BOOL	I、Q、M、D、L	シグナル状態「0」は、「閉じる」コマンドを示します。 シグナル状態「1」は、「開く」コマンドを示します。
O_FB	Input	BOOL	I、Q、M、D、L	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L	閉じるときにフィードバック入力
Q	Output	BOOL	I、Q、M、D、L	CMD パラメータのステータスを示します
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
ET	Static	DINT	D、L	現在の経過時間、1 カウント=1 ミリ秒。
PT	Static	DINT	D、L	プリセット時間値、1 カウント=1 ミリ秒。
PREV_TIME	Static	DWORD	D、L	前のシステム時間
CMD_HIS	Static	BOOL	D、L	CMD の履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

静的パラメータは、プログラムで命令を呼び出しているときには表示されません。 これらは、命令のインスタンスに保存されます。

例

次の例では、CMD パラメータが「0」から「1」に切り替わります。 命令の実行後、パラメータ Q が「1」にセットされ、2 つのアラーム出力 OA および CA のシグナル状態は「0」になります。 インスタンスデータブロックのパラメータ CMD_HIS がシグナル状態「1」にセットされ、パラメータ ET が「0」にリセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

```
SCL
"DCAT_DB"(CMD := "Tag_Input_CMD",
          O_FB := "Tag_Input_O_FB",
          C_FB := "Tag_Input_C_FB",
          Q => "Tag_Output_Q",
          OA => "Tag_Output_OA",
          CA => "Tag_Output_CA");
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令のインスタンスデータブロック「DCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令のインスタンスデータブロック「DCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

MCAT: モータ制御タイマアラーム

説明

「モータ制御タイマアラーム」命令を使用して、コマンド入力(開または閉)のいずれかのスイッチがオンにされる時点から、時間を累積します。 時間はプリセット済み時間が経過するまで、または関連のフィードバック入力が、デバイスが指定された時間内に要求された操作を実行したことを示す場合に蓄積されます。 フィードバックが受信される前にプリセット済み時間が経過すると、対応するアラームがトリガされます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「モータ制御タイマアラーム」命令の実行

次の表に、さまざまな入力条件に対する「モータ制御タイマアラーム」命令の応答を示します。

入力パラメータ								出力パラメータ								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	状態
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening (開き始める)
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open (開く)
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened (開いた)
>= PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm (アラームを開いている)
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing

															(閉じ始める)
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close (閉じる)
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed (閉じた)
>= PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm (アラームを閉じている)
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped (停止)

凡例:

INC	FB から ET への最後に処理されてからの時間差(ms)を追加します。
PT	PT は ET と同じ値にセットされます。
X	使用不可
<PT	ET < PT
>=PT	ET >= PT

入力パラメータ O_HIS および C_HIS のシグナル状態が両方とも「1」の場合、これらのパラメータのシグナル状態が直ちに「0」にセットされます。 この場合、上記の表の最後の行(X)が有効になります。 入力パラメータ O_HIS および C_HIS のシグナル状態が「1」であるかどうかをチェックすることができないため、この場合の出力パラメータは以下のようにセットされます。

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

構文

以下の構文が、「モータ制御タイマアラーム」命令に使用されます。

SCL

```
<Instance>(O_CMD := <Operand>,
           C_CMD := <Operand>,
           S_CMD := <Operand>,
           O_FB := <Operand>,
           C_FB := <Operand>,
           OO => <Operand>,
           CO => <Operand>,
```

```
OA => <Operand>,
CA => <Operand>,
Q => <Operand>)
```

パラメータ

次の表に、「モータ制御タイマアラーム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
O_CMD	Input	BOOL	I、Q、M、D、L	[開]コマンド入力
C_CMD	Input	BOOL	I、Q、M、D、L	[閉]コマンド入力
S_CMD	Input	BOOL	I、Q、M、D、L	[停止]コマンド入力
O_FB	Input	BOOL	I、Q、M、D、L	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L	閉じるときにフィードバック入力
OO	Output	BOOL	I、Q、M、D、L	[開]出力
CO	Output	BOOL	I、Q、M、D、L	[閉]出力
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「0」は、エラー条件を示します。
ET	Static	DINT	D、L	1 カウント= 1 ms のときの現在の経過時間
PT	Static	DINT	D、L	1 カウント= 1 ms のときのプリセット時間値
PREV_TIME	Static	DWORD	D、L	前のシステム時間
O_HIS	Static	BOOL	D、L	[開]履歴ビット
C_HIS	Static	BOOL	D、L	[閉]履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

静的パラメータは、プログラムで命令を呼び出しているときには表示されません。 これらは、命令のインスタンスに保存されます。

例

次の例で、命令がどのように動作するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。

```
SCL
"MCAT_DB"(O_CMD := "Tag_Iput_O_CMD",
          C_CMD := "Tag_Input_C_CMD",
          S_CMD := "Tag_Input_S_CMD",
```

```
O_FB := "Tag_Input_O_FB",
C_FB := "Tag_Input_C_FB",
OO => "Tag_OutputOpen",
CO => "Tag_OutputClosed",
OA => "Tag_Output_OA",
CA => "Tag_Output_CA",
Q => "Tag_Output_Q");
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

以下の値が命令のインスタンスデータブロック「MCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

以下の値が命令のインスタンスデータブロック「MCAT_DB」に保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

IMC: 入力ビットをマスクビットと比較

説明

「入力ビットをマスクビットと比較」命令を使用して、最大 16 のプログラムされた入力ビット(IN_BIT0 ~ IN_BIT15)のシグナル状態とマスクの対応するビットを比較します。 最大 16 のマスクされたステップをプログラム可能です。IN_BIT0 パラメータの値が、マスク CMP_VAL[x,0]の値と比較されます。「x」はステップ番号を示しています。 CMP_STEP パラメータで、比較に使用するマスクのステップ番号を指定します。 プログラムされた値は、すべて同じ方法で比較されます。 プログラムされていない入力ビットやマスクビットは、既定のシグナル状態である FALSE になります。

比較で一致が検出されると、OUT パラメータのシグナル状態が「1」にセットされます。 それ以外の場合 OUT パラメータが「0」にセットされます。

CMP_STEP パラメータの値が 15 よりも大きい場合、この命令は実行されません。 ERR_CODE パラメータでエラーメッセージが出力されます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「入力ビットをマスクビットと比較」命令には、以下の構文を使用します。

```
SCL
<Instance>(IN_BIT0 - 15 := <Operand>,
           CMP_STEP := <Operand>,
           OUT => <Operand>,
           ERR_CODE => <Operand>)
```

パラメータ

次の表に、「入力ビットをマスクビットと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN_BIT0	Input	BOOL	I、Q、M、D、L	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L	入力ビット 2 がマスクビット 2 と比較されます。
IN_BIT3	Input	BOOL	I、Q、M、D、L	入力ビット 3 がマスクビット 3 と比較されます。

IN_BIT4	Input	BOOL	I、Q、M、D、L	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L	入力ビット 15 がマスクビット 15 と比較されます。
CMP_STEP	Input	BYTE	I、Q、M、D、L、P	比較に使用されるマスクのステップ番号。
OUT	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、一致が検出されたことを示します。 シグナル状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

静的パラメータは、プログラムで命令を呼び出しているときには表示されません。 これらは、命令のインスタンスに保存されます。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000A	CMP_STEP パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

SMC: スキャンマトリックスの比較

説明

「スキャンマトリックスの比較」命令を使用して 16 のプログラム済み入力ビット(IN_BIT0～IN_BIT15)のシグナル状態を各ステップの比較マスクの対応するビットと比較します。 処理はステップ 1 から開始し、プログラムされた最後のステップ(LAST)まで、または一致が検出されるまで続行されます。IN_BIT0 パラメータの入力ビットが、マスク CMP_VAL[x,0]の値と比較されます。「x」はステップ番号を示しています。 プログラムされた値は、すべて同じ方法で比較されます。 一致が検出されると、OUT パラメータのシグナル状態が「1」にセットされ、一致するマスクのステップ番号が OUT_STEP パラメータに書き込まれます。 プログラムされていない入力ビットやマスクビットは、既定のシグナル状態である FALSE になります。 複数のステップに一致するマスクが存在する場合、最初に検出されたマスクのみが OUT_STEP パラメータに示されます。 一致が検出されないと、OUT パラメータのシグナル状態が「0」にセットされます。 この場合、OUT_STEP パラメータの値は LAST パラメータの値よりも「1」だけ大きくなります。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「スキャンマトリックスの比較」命令には、以下の構文を使用します。

```
SCL
<Instance>(IN_BIT0 - 15 := <Operand>,
           OUT => <Operand>,
           OUT_STEP => <Operand>,
           ERR_CODE => <Operand>)
```

構文

次の表に、「スキャンマトリックスの比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN_BIT0	Input	BOOL	I、Q、M、D、L	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L	入力ビット 2 がマスクビット 2 と比較されます。
IN_BIT3	Input	BOOL	I、Q、M、D、L	入力ビット 3 がマスクビット 3 と比較されます。

IN_BIT4	Input	BOOL	I、Q、M、D、L	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L	入力ビット 15 がマスクビット 15 と比較されます。
OUT	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、一致が検出されたことを示します。 シグナル状態「0」は、一致が検出されなかったことを示します。
OUT_STEP	Output	BYTE	I、Q、M、D、L、P	一致するマスクを持つステップの番号が含まれているか、または一致が検出されない場合は、LAST パラメータの値よりも「1」だけ大きいステップ番号が含まれています。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
LAST	Static	BYTE	I、Q、M、D、L、P	一致するマスクをスキャンする最後のステップのステップ番号を指定します。
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

静的パラメータは、プログラムで命令を呼び出しているときには表示されません。 これらは、命令のインスタンスに保存されます。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000E	LAST パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

LEAD_LAG: リード/ラグアルゴリズム

説明

「リード/ラグアルゴリズム」命令を使用して、アナログタグの信号を処理します。 GAIN パラメータのゲイン値はゼロよりも大きくなければなりません。「リード/ラグアルゴリズム」命令の結果は、以下の等式を使用して計算されます。

$$OUT = \left[\frac{LG\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_OUT + GAIN \left[\frac{LD\_TIME + SAMPLE\_T}{LG\_TIME + SAMPLE\_T}\right] IN - GAIN \left[\frac{LD\_TIME}{LG\_TIME + SAMPLE\_T}\right] * PREV\_IN$$

「リード/ラグアルゴリズム」命令は、処理を固定プログラムサイクルで行う場合のみ正しい結果が得られます。 LD_TIME、LG_TIME、および SAMPLE_T パラメータには同一のユニットを指定する必要があります。 LG_TIME > 4 + SAMPLE_T の場合、この命令は以下の関数を実行します。

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

GAIN パラメータの値がゼロ以下の場合、計算は実行されず、ERR_CODE パラメータにエラー情報が出力されます。

「リード/ラグアルゴリズム」命令をダイナミックフィードフォワード制御でループと共に補償器として使用できます。 この命令は、2 つの操作で構成されています。「Lead」演算は出力 OUT のフェーズをシフトして、出力が入力の前に来るようにします。 一方「Lag」演算は出力をシフトして、出力が入力の後になるようにします。「Lag」演算は積分に相当するため、ノイズサプレッサやローパスフィルタとして使用できます。「Lead」演算は微分と同等なため、ハイパスフィルタとして使用できます。 2 つの命令(Lead と Lag)をまとめて実行すると、低周波域では出力フェーズが入力フェーズよりも遅れ、高周波域では出力フェーズが入力フェーズに先行します。 つまり「リード/ラグアルゴリズム」命令は帯域フィルタとして使用できます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

構文

「リード/ラグアルゴリズム」命令には、以下の構文を使用します。

```
SCL
<Instance>(IN := <Operand>,
           SAMPLE_T := <Operand>,
           OUT => <Operand>,
           ERR_CODE => <Operand>)
```

次の表に、ブロックパラメータのタイプを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	REAL	I、Q、M、D、L、P	処理対象の現在のサンプル時間(サイクルタイム)の入力値。 定数も IN パラメータに指定することができます。
SAMPLE_T	Input	INT	I、Q、M、D、L、P	サンプル時間 定数も SAMPLE_T パラメータに指定することができます。
OUT	Output	REAL	I、Q、M、D、L	命令の結果
ERR_CODE	Output	WORD	I、Q、M、D、L	エラー情報
LD_TIME	Static	REAL	I、Q、M、D、L、P	サンプル時間としての同一ユニット内のリードタイム。
LG_TIME	Static	REAL	I、Q、M、D、L、P	サンプル時間としての同一ユニット内のラグタイム。
GAIN	Static	REAL	I、Q、M、D、L、P	% / %で表されるゲイン(定常状態時の出力の切り替えと入力の切り替えとの比率)。
PREV_IN	Static	REAL	I、Q、M、D、L、P	前の入力
PREV_OUT	Static	REAL	I、Q、M、D、L、P	前の出力

有効なデータタイプの追加情報については、「関連項目」を参照してください。

静的パラメータは、プログラムで命令を呼び出しているときには表示されません。 これらは、命令のインスタンスに保存されます。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
0009	GAIN パラメータの値がゼロ以下である。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。

```
SCL
"LEAD_LAG_DB"(IN := "Tag_Input",
              SAMPLE_T := "Tag_Input_SAMPLE_T",
              OUT => "Tag_Output_Result",
              ERR_CODE => "Tag_ErrorCode");
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

以下の値が命令のインスタンスデータブロック「LEAD_LAG_DB」に保存されます。

パラメータ	アドレス	値
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output_Result	2.0

以下の値が命令のインスタンスデータブロック「LEAD_LAD_DB」に保存されます。

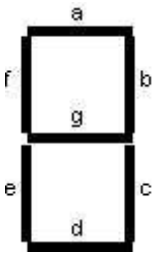
パラメータ	オペランド	値
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

SEG: 7 セグメント表示のビットパターンの作成

説明

「7 セグメント表示のビットパターンの作成」命令を使用して、指定されたソースワードの 4 つの各 16 進数(IN)を 7 セグメント表示の相当するビットパターンに変換します。 命令の結果は、OUT パラメータにダブルワードで出力されます。

16 進数と 7 セグメント(a、b、c、d、e、f、g)の割り当ての間には、以下の関係があります。

入力する数値 (バイナリ)	**セグメントの割り当て** **-gfedcba**	**表示** (16 進数)	**7 セグメント表示**
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

構文

「7 セグメント表示のビットパターンの作成」命令には、以下の構文を使用します。

```
SCL
SEG(IN := <Operand>,
    OUT => <Operand>)
```

パラメータ

次の表に、「7 セグメント表示のビットパターンの作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	WORD	I、Q、M、D、L、P	4 つの 16 進数のソースワード
OUT	Output	DWORD	I、Q、M、D、L、P	7 セグメント表示のビットパターン
ファンクション値		VOID		空のファンクション値

例

次の例で、命令がどのように動作するかを示します。

```
SCL
SEG(IN := "Tag_Input",
    OUT => "Tag_Output");
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値	
		16 進数	2 進数
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW16#065B4F66	00000110 01011011 01001111 01100110 表示: 1234

BCDCPL: 10 の補数の作成

説明

「10 の補数の作成」命令を使用して、オペランドで指定された 7 桁の BCD 数の 10 の補数を作成します。 この命令は、次の数式を使用して計算します。

10000000 (BCD として)

– 7 桁の BCD 数

---

10 の補数(BCD として)

構文

「10 の補数の作成」命令には、以下の構文を使用します。

```
SCL
BCDCPL(<Operand>)
```

パラメータ

次の表に、「10 の補数の作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	ビット列	I、Q、M、D、L、P	7 桁の BCD 数
ファンクション値		DWORD	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := BCDCPL("Tag_Input");
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

オペランド	値*
Tag_Input	DW#16#01234567
Tag_Result	DW#16#08765433
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

BITSUM: セットビット数カウント

説明

「セットビット数のカウント」命令を使用して、シグナル状態「1」にセットされるオペランドのビット数をカウントします。

構文

「セットビット数カウント」命令には、以下の構文を使用します。

```
SCL
BITSUM(<Operand>)
```

パラメータ

次の表に、「セットビット数カウント」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	DWORD	I、Q、M、D、L、P	セットビット数をカウントするオペランド
ファンクション値		INT	I、Q、M、D、L、P	命令の結果

例

次の例で、命令がどのように動作するかを示します。

```
SCL
"Tag_Result" := BITSUM("Tag_Input");
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

オペランド	値*
Tag_Input	DW#16#12345678
Tag_Result	W#16#000D (13 ビット)
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

GRAPH

この章には下記に関する情報が記載されています：

- グラフ構成 (S7-1500)
- GRAPH 操作 (S7-1500)
- LAD 命令比較演算 (S7-1500)
- FBD 命令比較演算 (S7-1500)

グラフ構成

この章には下記に関する情報が記載されています：

- ステップと移行 (S7-1500)
- ステップ (S7-1500)
- 移行 (S7-1500)
- シーケンス終了 (S7-1500)
- ステップにジャンプ (S7-1500)
- 代替分岐 (S7-1500)
- 同時分岐 (S7-1500)
- 分岐を閉じる (S7-1500)

ステップと移行

説明

「ステップと移行」構造体工レメントを使用して、PLC にステップと移行を同時に挿入することができます。

関連項目

ステップ

移行

ステップ

説明

ステップは、操作すると完成できる簡単なサブタスクに、複雑なオートメーションタスクを分割するために使用されます。 個々のステップは、プログラム実行時に指定された順序ですべてのステップを実行できるように、PLC にまとめられます。 各ステップには、一意の名前と番号を割り当てる必要があります。

実行が実際に行われるようにするには、次のいずれかの条件によってステップを有効にする必要があります。

- ステップが最初のステップとして定義されていること。
- 前のステップの移行が完了していること。
- イベント依存の操作によってステップが呼び出されていること。

すべての操作が実行されると、ステップは再度無効になります。

操作がプログラムされていないステップは、空のステップと呼ばれます。 このような空のステップは有効なステップと同様に応答するため、後続の移行は常に完了します。

移行

説明

移行は、ステップ間に配置されており、あるステップから次のステップに進むための条件(ステップイネーブル)を含んでいます。 移行のステップイネーブル条件が満たされると、次のステップが有効になり、その操作が実行されます。 LAD または FBD のいずれかで移行の条件をプログラムできます。

シーケンス終了

説明

「シーケンス終了」構造体工レメントを使用して、PLC または分岐を停止できます。 同時分岐では、移行はシーケンス終了に先行する必要があります。

注記
PLC のすべての分岐がシーケンス終了で完了した場合は、「INIT_SQ」パラメータまたは[テスト]タスクカードの[シーケンスコントロール]ペインにある[初期化]ボタンを使用することによって、PLC を再起動できます。

ステップにジャンプ

説明

ジャンプを使用して、GRAPH ファンクションブロック内の任意のステップでプログラムの実行を続行することができます。 PLC の周期的処理を有効にするために、メイン分岐または代替分岐の最後にジャンプを挿入できます。 ジャンプおよびジャンプ先は PLC 内で矢印として表されます。その場合、ジャンプ先に対して復帰移行が指定されます。

代替分岐

説明

代替分岐を使用して OR 分岐をプログラミングできます。 つまり、移行から始まる分岐をステップの後に挿入します。 どの移行条件が最初に満たされるかに応じて、対応する分岐が実行されます。 複数の移行条件が同時に満たされた場合、左端の移行が最上位の優先度を持ち、対応する分岐が実行されます。 代替分岐が移行で再度終了します。

PLC では最大 125 個の代替分岐をプログラミングできます。

同時分岐

説明

同時分岐を使用して AND 分岐をプログラミングできます。 つまり、1 つの移行を使用して複数のステップを有効にします。その後、それらの操作が実行されます。 同時分岐は常にステップから開始します。

同時分岐の後続の移行はメイン分岐に置かれます。この場合、各種の同時分岐をメイン分岐のさまざまなポイントに接続することができます。 移行で結合した分岐は、すべての分岐が完全に実行されたときにしか次のステップに進まないことに注意してください。

PLC では、最大 249 個の同時分岐をプログラミングできます。

分岐を閉じる

説明

[分岐を閉じる]エレメントを使用して、親分岐に対する同時分岐および代替分岐を閉じることができます。 ジャンプまたはシーケンス終了を用いて分岐を完了したくない場合は、この操作が必要です。 同時分岐では、ステップの後にしか「分岐を閉じる」を挿入できません。

GRAPH 操作

この章には下記に関する情報が記載されています：

タイマの動作

この章には下記に関する情報が記載されています：

- TP: パルスの生成 (S7-1500)
- TON: オンディレー生成 (S7-1500)
- TOF: オフディレー生成 (S7-1500)
- TONR: タイムアキュムレータ (S7-1500)

TP: パルスの生成

説明

「パルス生成」命令を使用して、プログラムされた持続時間の Q 出力を設定します。 この命令は、IN パラメータの論理演算結果(RLO)が「0」から「1」に切り替わるとき(信号立ち上がりエッジ)に開始されます。 命令が開始されると、プログラムされた時間 PT が開始します。 以降の入力信号の切り替えに関わらず、Q パラメータが時間 PT に対してセットされます。 信号の新しい立ち上がりエッジが検出されても、時間 PT が有効である限り、Q パラメータのシグナル状態は影響を受けません。

ET パラメータにセットされた現在の時間値を照会できます。 タイマ値は T#0s で開始し、時間 PT の値に達すると終了します。 時間 PT に達したときにパラメータ IN のシグナル状態が「0」の場合、パラメータ ET はリセットされます。

注記
時間がプログラム内でたとえばスキップされて呼び出されない場合は、ET パラメータは、時間の期限が切れるとすぐに定数値を返します。

「パルス生成」命令の各呼び出しは、命令データが保存される IEC タイマに割り当てる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TP_TIME、または TP_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TP_TIME、TP_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「パルス生成」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「パルス生成」命令には、以下の構文を使用します。

```
GRAPH
CALL TP ???, <IEC timer>
     (IN := <operand>
      PT := <operand>
      Q => <Operand>
      ET => <Operand>
      )
```

パラメータ

次の表に、「パルス生成」命令のパラメータを示します。

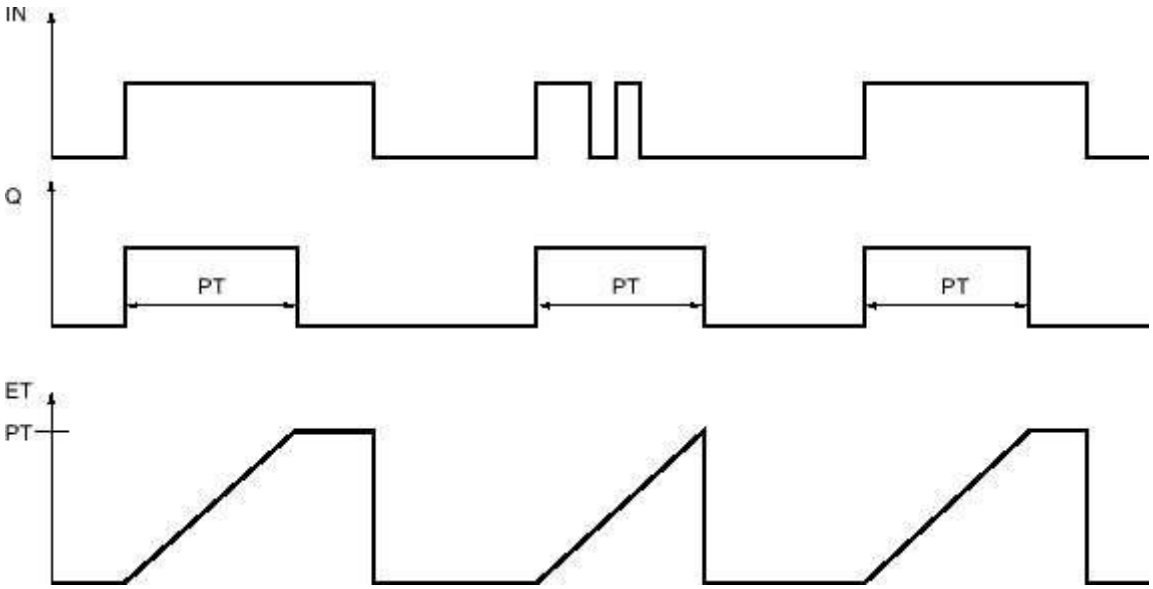
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	パルスの持続時間。 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	パルス出力
ET	Output	TIME, LTIME	I、Q、M、D、L、P	現在のタイマ値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスダイアグラム

次の図に、開始後の「パルス生成」命令のパルス図を示します。



例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL TP TIME, "IEC_TP_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
```

```
        ET => "Tag_ElapsedTIME"
        )
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わるときに、PT パラメータに対してプログラムされた時間が開始され、オペランド「Tag_Status」が「1」にセットされます。 現在の時間値は、オペランド「Tag_ElapsedTIME」内に保存されます。

TON: オンディレー生成

説明

「オンディレータイマ」命令を使用して、プログラムされた時間 PT によって Q 出力の設定を遅延させます。 この命令は、IN パラメータの論理演算結果(RLO)が「0」から「1」に切り替わるとき(信号立ち上がり工ッジ)に開始されます。 命令が開始されると、プログラムされた時間 PT が開始します。時間 PT が経過すると、Q パラメータのシグナル状態は「1」になります。 Q パラメータは、開始入力 IN が「1」の間はセットされた状態が続きます。開始入力のシグナル状態が「1」から「0」に切り替わると、Q パラメータがリセットされます。 開始入力で信号の新しい立ち上がり工ッジが検出されると、タイマファンクションが再開されます。

ET 出力で現在時間値を照会できます。 タイマ値は T#0s で開始し、時間 PT の値に達すると終了します。IN パラメータがシグナル状態「0」に切り替わると、ET パラメータは直ちにリセットされます。

注記
このタイマが、たとえば、プログラムでスキップされて、呼び出されない場合、ET 出力は時間の期限が切れるとすぐに定数値を返します。

「オンディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TON_TIME、または TON_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TON_TIME、TON_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「オンディレータイマ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「オンディレータイマ」命令には、以下の構文を使用します。

```
GRAPH
CALL TON ???, <IEC timer>
     (IN := <operand>
      PT := <operand>
      Q => <Operand>
      ET => <Operand>
      )
```

パラメータ

次の表に、「オンディレータイマ」命令のパラメータを示します。

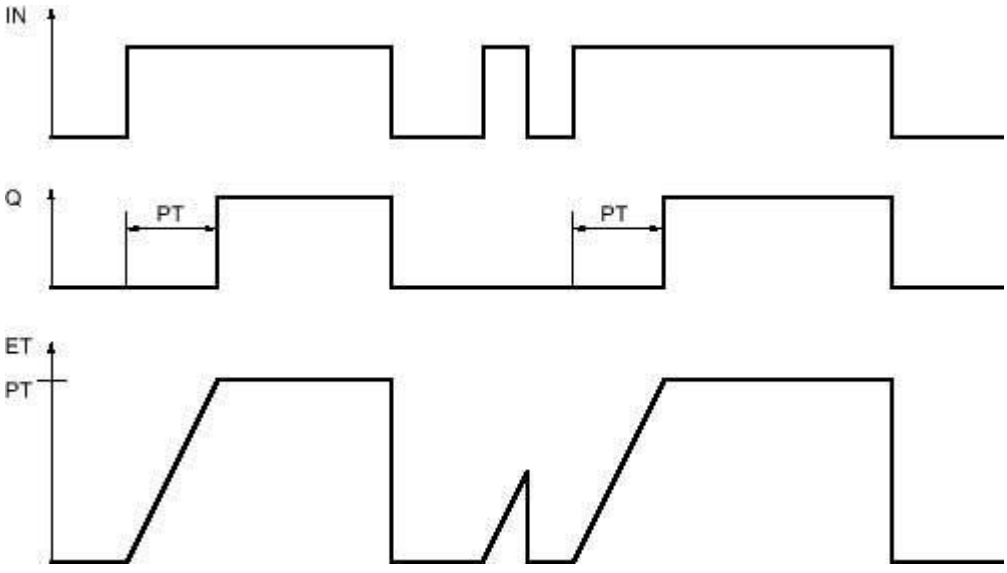
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	オンディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	時間 PT によって遅延されるシグナル状態
ET	Output	TIME, LTIME	I、Q、M、D、L、P	現在のタイマ値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスダイアグラム

次の図に、開始後の「オンディレータイマ」命令の動作を示します。



例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL TON TIME, "IEC_TON_DB"
     (IN := "Tag_Start"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
```

```
        ET => "Tag_ElapsedTIME"
        )
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わると、PT パラメータにプログラムされた時間が開始します。 時間の期限が切れると、「Tag_Status」オペランドがシグナル状態「1」にセットされます。 Tag_Start オペランドのシグナル状態が「1」である限り、Tag_Status オペランドは「1」にセットされた状態が継続します。 現在の時間値は、オペランド「Tag_ElapsedTime」内に保存されます。「Tag_Start」オペランドのシグナル状態が「1」から「0」に切り替わると、「Tag_Status」オペランドがリセットされます。

TOF: オフディレー生成

説明

「オフディレータイマ」命令を使用して、プログラムされた時間 PT によって Q 出力のリセットを遅延させます。 Q パラメータは、IN パラメータの論理演算結果(RLO)が「0」から「1」(信号の立ち上がりエッジ)に切り替わるときにセットされます。IN パラメータがシグナル状態「0」に戻ると、プログラムされた時間 PT が開始します。 時間 PT が継続している限り、Q パラメータはセットされたままです。 時間 PT が経過すると、Q パラメータはリセットされます。 時間 PT の期限が切れる前に、IN パラメータのシグナル状態が「1」に切り替わると、タイマはリセットされます。 Q パラメータのシグナル状態は「1」にセットされた状態が継続します。

現在の時間値は、ET パラメータ内で照会することができます。 タイマ値は T#0s で開始し、時間 PT の値に達すると終了します。 時間 PT が経過すると、ET パラメータは、IN パラメータが「1」に戻るまで、現在の値がセットされた状態が続きます。 時間 PT の期限が切れる前に IN 入力が「1」に切り替わると、ET 出力が値 T#0s にリセットされます。

注記
このタイマが、たとえば、プログラムでスキップされて、呼び出されない場合、ET 出力は時間の期限が切れるとすぐに定数値を返します。

「オフディレータイマ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TOF_TIME、または TOF_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TOF_TIME、TOF_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「オフディレータイマ」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「オフディレータイマ」命令には、以下の構文を使用します。

```
GRAPH
CALL TOF ???, <IEC timer>
     (IN := <operand>
      PT := <operand>
      Q => <Operand>
      ET => <Operand>
```

)

パラメータ

次の表に、「オフディレータイマ」命令のパラメータを示します。

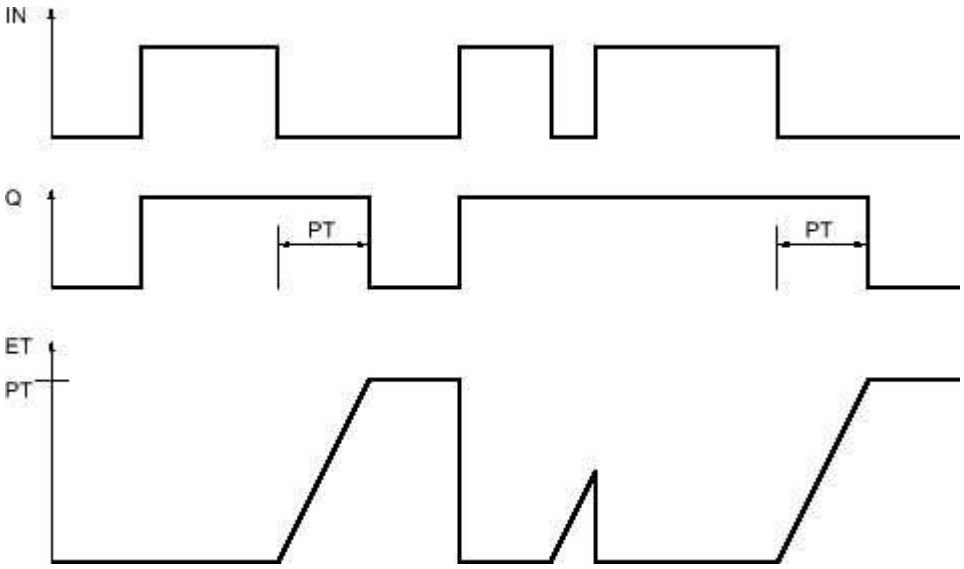
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	オフディレーの持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	時間 PT によって遅延されるシグナル状態
ET	Output	TIME, LTIME	I、Q、M、D、L、P	現在のタイマ値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスダイアグラム

次の図に、開始後の「オフディレータイマ」命令の動作を示します。



例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL TOF TIME, "IEC_TOF_DB"
     (IN := "Tag_Start"
```

```
PT := "Tag_PresetTIME"
Q => "Tag_Status"
ET => "Tag_ElapsedTIME"
)
```

「Tag_Start」オペランドのシグナル状態が「0」から「1」に切り替わると、「Tag_Status」オペランドがセットされます。オペランド「Tag_Start」のシグナル状態が「1」から「0」に切り替わると、PTパラメータにプログラムされた時間が開始します。 時間が継続中の場合は、オペランド「Tag_Status」はセットされた状態が続きます。時間が経過すると、オペランド「Tag_Status」はリセットされます。現在の時間値は、オペランド「Tag_ElapsedTIME」内に保存されます。

TONR: タイムアキュムレータ

説明

「タイムアキュムレータ」命令を使用して、PT パラメータによってセットされた時間内の時間値を累積します。IN 入力のシグナル状態が「0」から「1」(信号立ち上がりエッジ)に切り替わると命令が実行され、時間 PT が開始します。 時間 PT が経過している間、IN 入力のシグナル状態が「1」のときに記録された時間値が累積されます。 累積時間は ET 出力に書き込まれ、そこで照会できます。 持続時間 PT が経過すると、出力 Q のシグナル状態は「1」になります。IN パラメータのシグナル状態が「1」から「0」(信号立ち下がりエッジ)に切り替わった場合でも、Q パラメータは「1」にセットされた状態が継続します。

開始入力のシグナル状態に関係なく、R 入力が出力 ET と Q をリセットします。

「タイムアキュムレータ」命令の各呼び出しは、命令データが格納される IEC タイマに割り当てられる必要があります。 IEC タイマは、データタイプ IEC_TIMER、IEC_LTIMER、TONR_TIME、または TONR_LTIME の構造体であり、次のように宣言することが可能です。

- システムデータタイプ IEC_TIMER または IEC_LTIMER のデータブロックの宣言(「MyIEC_TIMER」など)
- ブロックの「Static」セクション内のタイプ TONR_TIME、TONR_LTIME、IEC_TIMER、または IEC_LTIMER のローカルタグとしての宣言(#MyIEC_TIMER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC タイマが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

命令データは、命令が呼び出されるとき、および出力 Q または ET がアクセスされるたびに更新されます。

「タイムアキュムレータ」命令を実行するには、その前に論理演算を行う必要があります。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「タイムアキュムレータ」命令には、以下の構文を使用します。

```
GRAPH
CALL TONR ???, <IEC timer>
     (IN := <operand>
      R := <operand>
      PT := <operand>
      Q => <Operand>
      ET => <Operand>
      )
```

パラメータ

次の表に、「タイムアキュムレータ」命令のパラメータを示します。

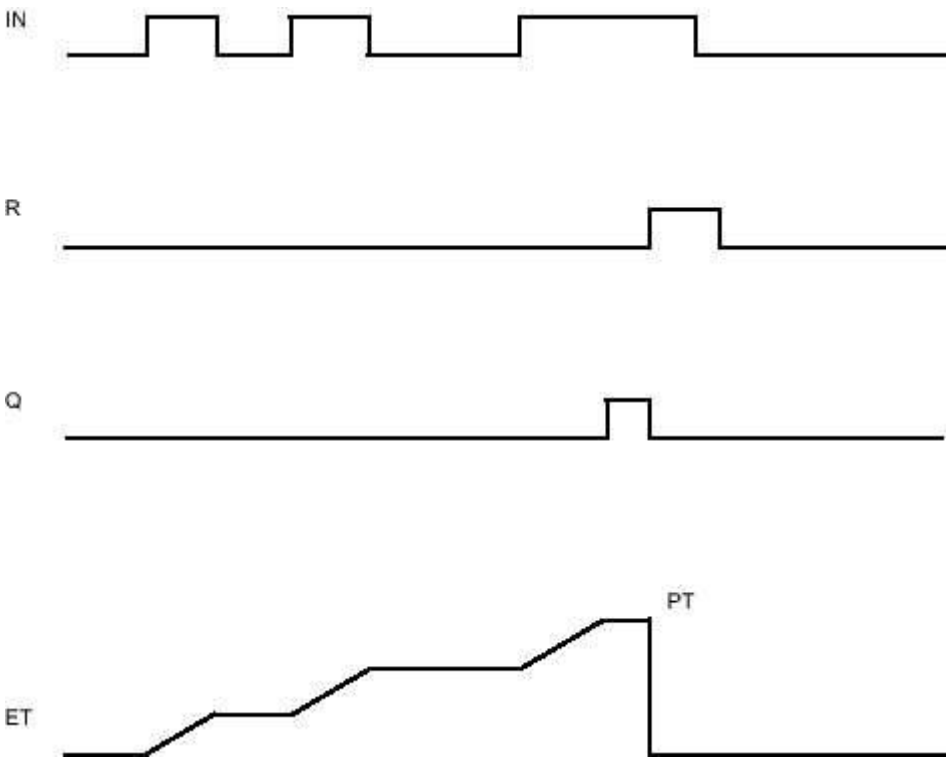
パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	BOOL	I、Q、M、D、L、P、または定数	開始入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	リセット入力
PT	Input	TIME, LTIME	I、Q、M、D、L、P、または定数	時間記録の最大持続時間 PT パラメータの値は正の値であることが必要です。
Q	Output	BOOL	I、Q、M、D、L、P	時間 PT の経過時にセットされる出力。
ET	Output	TIME, LTIME	I、Q、M、D、L、P	累積時間

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

パルスダイアグラム

次の図に、「タイムアキュムレータ」命令のパルスダイアグラムを示します。



例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL TONR TIME, "IEC_TONR_DB"
     (IN := "Tag_Start"
      R := "Tag_Reset"
      PT := "Tag_PresetTIME"
      Q => "Tag_Status"
      ET => "Tag_ElapsedTIME"
      )
```

オペランド「Tag_Start」のシグナル状態が「0」から「1」に切り替わると、PT パラメータにプログラムされた時間が開始します。 時間が経過している間、「Tag_Start」オペランドのシグナル状態が「1」のときに記録された時間値が累積されます。 累積時間は「Tag_ElapsedTIME」オペランド内に保存されます。 PT パラメータに指定されている時間値に達すると、「Tag_Status」オペランドがシグナル状態「1」にセットされます。現在の時間値は、オペランド「Tag_ElapsedTIME」内に保存されます。

カウンタ演算

この章には下記に関する情報が記載されています：

- CTU: カウントアップ (S7-1500)
- CTD: カウントダウン (S7-1500)
- CTUD: カウントアップ/カウントダウン (S7-1500)

CTU: カウントアップ

説明

CV パラメータの値のインクリメントには、「カウントアップ」命令を使用できます。CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わるとこの命令が実行され、CV パラメータのカウンタ現在値が 1 インクリメントされます。 立ち上がりエッジを検出するたびに、CV 出力において指定されたデータタイプの上限値に達するまで、カウンタ値がインクリメントされます。 上限値に達すると、CU パラメータのシグナル状態は命令に影響しなくなります。

Q パラメータのカウンタステータスを照会できます。 Q パラメータのシグナル状態は、PV パラメータによって決まります。 カウンタ現在値が PV パラメータの値以上の場合、Q パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、Q パラメータのシグナル状態は「0」です。

R パラメータのシグナル状態が「1」に切り替わると、CV パラメータの値はゼロにリセットされます。R パラメータのシグナル状態が「1」である限り、CU パラメータのシグナル状態はこの命令には影響を与えません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。 IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

システムデータタイプ IEC_<Counter> (共有 DB)のデータブロック

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

ローカルタグ

- CTU_SINT / CTU_USINT
- CTU_INT / CTU_UINT
- CTU_DINT / CTU_UDINT
- CTU_LINT / CTU_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTU_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyCTU_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。 インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントアップ」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「カウントアップ」命令には、以下の構文を使用します。

```
GRAPH
CALL CTU ???, <IEC counter>
     (CU := <operand>
      R := <operand>
      PV := <operand>
       Q => <Operand>
      CV => <Operand>
      )
```

パラメータ

次の表に、「カウントアップ」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CU	Input	BOOL	I、Q、M、D、L、または定数	カウント入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	リセット入力
PV	Input	整数	I、Q、M、D、L、P、または定数	Q 出力がセットされる値。
Q	Output	BOOL	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL CTU INT, "IEC_CTU_DB"
     (CU := "Tag_Start"
      R := "Tag_ResetCOUNTER"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

「Tag_Start」オペランドのシグナル状態が「0」から「1」に切り替わると、「カウントアップ」命令が実行され、「Tag_CounterValue」カウンタ現在値が 1 インクリメントされます。 それ以降、信号立ち上がりエッジが検出されるたびに、データタイプの上限値(INT = 32767)に達するまで、カウンタ値がインクリメントされます。

PV パラメータの値が「Tag_Status」出力を確定する限度として採用されます。 カウンタ現在値がオペランド「Tag_PresetValue」の値以上の場合、「Tag_Status」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_Status」出力のシグナル状態は「0」になります。

CTD: カウントダウン

説明

「カウントダウン」命令は、パラメータ CV のデクリメントに使用します。CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わるとこの命令が実行され、CV パラメータのカウンタ現在値が 1 デクリメントされます。 指定されたデータタイプの下限値に達するまで、信号の立ち上がりエッジが検出されるたびにカウンタがデクリメントされます。 下限値に達すると、CD パラメータのシグナル状態は命令に影響しなくなります。

Q パラメータのカウンタステータスを照会できます。 カウンタ現在値がゼロ以下の場合、Q パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、Q パラメータのシグナル状態は「0」です。

CV パラメータの値は、LD パラメータのシグナル状態が「1」に切り替わると PV パラメータの値にセットされます。LD パラメータのシグナル状態が「1」である限り、CD パラメータのシグナル状態はこの命令には影響を与えません。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

システムデータタイプ IEC_<Counter> (共有 DB)のデータブロック

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

ローカルタグ

- CTD_SINT / CTD_USINT
- CTD_INT / CTD_UINT
- CTD_DINT / CTD_UDINT
- CTD_LINT / CTD_ULINT
- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyCTD_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントダウン」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「カウントダウン」命令には、以下の構文を使用します。

```
GRAPH
CALL CTD ???, <IEC counter>
     (CD := <operand>
      LD := <operand>
      PV := <operand>
       Q => <Operand>
      CV => <Operand>
      )
```

パラメータ

次の表に、「カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CD	Input	BOOL	I、Q、M、D、L、または定数	カウント入力
LD	Input	BOOL	I、Q、M、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	CV 出力が LD = 1 にセットされる値。
Q	Output	BOOL	I、Q、M、D、L	カウンタステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL CTD INT, "IEC_CTD_DB"
     (CD := "Tag_Start"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
      Q => "Tag_Status"
      CV => "Tag_CounterValue"
      )
```

「Tag_Start」オペランドのシグナル状態が「0」から「1」に切り替わると、「カウントダウン」命令が実行され、「Tag_CounterValue」出力の値が 1 デクリメントされます。 以後のそれぞれの信号立ち上がりエッジで、指定したデータタイプの下限値(INT = -32768)に達するまでカウンタ値がデクリメントされます。

カウンタ現在値がゼロ以下である限り、「Tag_Status」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_Status」出力のシグナル状態は「0」になります。

CTUD: カウントアップ/カウントダウン

説明

「カウントアップ/カウントダウン」命令を使用して、CV パラメータの値をインクリメントおよびデクリメントできます。CU パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、CV パラメータのカウンタ現在値が 1 インクリメントされます。 CD パラメータのシグナル状態が「0」から「1」(信号の立ち上がりエッジ)に切り替わると、CV パラメータのカウンタ現在値が 1 デクリメントされます。 プログラムサイクル内の CU および CD 入力で信号の立ち上がりエッジが存在すると、CV パラメータのカウンタ現在値は変化しません。

カウンタ値は、CV パラメータで指定したデータタイプの上限値に達するまでインクリメントすることができます。 上限値に達すると、信号の立ち上がりエッジが発生しても、カウンタ値はインクリメントされなくなります。 カウンタ値は、指定されたデータタイプの下限値に達するとデクリメントされなくなります。

LD パラメータのシグナル状態が「1」に切り替わると、CV パラメータのカウンタ値が PV パラメータの値にセットされます。 LD パラメータのシグナル状態が「1」である限り、CU 入力および CD 入力のシグナル状態は命令に対する効力を持ちません。

R パラメータのシグナル状態が「1」に切り替わると、カウンタ値は 0 にセットされます。 R パラメータのシグナル状態が「1」である限り、CU、CD パラメータおよび LD パラメータのシグナル状態の変化は「カウントアップ/カウントダウン」命令に影響しません。

QU パラメータでアップカウンタのステータスを照会できます。 カウンタ現在値が PV パラメータの値以上の場合、QU パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、QU パラメータのシグナル状態は「0」です。

QD パラメータでダウンカウンタのステータスを照会できます。 カウンタ現在値がゼロ以下の場合、QD パラメータがシグナル状態「1」にセットされます。 それ以外のすべての場合、QD パラメータのシグナル状態は「0」です。

注記
カウントエラーのリスクを回避するために、プログラムでは 1 つの場所のカウンタのみを使用します。

「カウントアップ/カウントダウン」命令の各呼び出しは、命令データが格納される IEC カウンタに割り当てられる必要があります。IEC カウンタとは、次のいずれかのデータタイプを持つ構造体です。

システムデータタイプ IEC_<Counter> (共有 DB)のデータブロック

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

ローカルタグ

- CTUD_SINT / CTUD_USINT
- CTUD_INT / CTUD_UINT
- CTUD_DINT / CTUD_UDINT
- CTUD_LINT / CTUD_ULINT

- IEC_SCOUNTER / IEC_USCOUNTER
- IEC_COUNTER / IEC_UCOUNTER
- IEC_DCOUNTER / IEC_UDCOUNTER
- IEC_LCOUNTER / IEC_ULCOUNTER

IEC カウンタは、次のように宣言できます。

- システムデータタイプ IEC_<Counter>のデータブロックの宣言(「MyIEC_COUNTER」など)
- ブロックの「Static」セクションでの CTUD_<Data type>または IEC_<Counter>タイプのローカルタグとしての宣言(#MyCTUD_COUNTER など)

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここで IEC カウンタが自己のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

個別のデータブロック(シングルインスタンス)内にこの IEC カウンタをセットアップすると、このインスタンスデータブロックは、既定では「最適化したブロックアクセス」で作成され、個々のタグは保持型として定義されます。インスタンスデータブロックでの保持型の設定についての追加情報は、「関連項目」を参照してください。

「最適化したブロックアクセス」のファンクションブロック内で、この IEC カウンタをローカルタグ(マルチインスタンス)としてセットアップすると、このカウンタは、ブロックインターフェース内で、保持型として定義されます。

「カウントアップ/カウントダウン」命令の実行には、あらかじめ論理演算が必要です。 これは、ネットワーク内またはネットワークの終端に配置できます。

構文

「カウントアップ/カウントダウン」命令には、以下の構文を使用します。

```
GRAPH
CALL CTUD ???, <IEC counter>
     (CU := <operand>
      CD := <operand>
      R := <operand>
      LD := <operand>
      PV := <operand>
     QU => <Operand>
      QD => <Operand>
      CV => <Operand>
      )
```

パラメータ

次の表に、「カウントアップ/カウントダウン」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CU	Input	BOOL	I、Q、M、D、L、または定数	カウントアップ入力
CD	Input	BOOL	I、Q、M、D、L、または定数	カウントダウン入力
R	Input	BOOL	I、Q、M、D、L、P、または定数	リセット入力
LD	Input	BOOL	I、Q、M、D、L、P、または定数	ロード入力
PV	Input	整数	I、Q、M、D、L、P、または定数	QU 出力がセットされる値。 /CV 出力が LD = 1 にセットされる値。
QU	Output	BOOL	I、Q、M、D、L	アップカウンタのステータス
QD	Output	BOOL	I、Q、M、D、L	カウントダウンのステータス
CV	Output	整数、CHAR、WCHAR、DATE	I、Q、M、D、L、P	カウンタ現在値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL CTUD INT, "IEC_CTUD_DB"
     (CU := "Tag_StartCTU"
      CD := "Tag_StartCTD"
      R := "Tag_ResetCOUNTER"
      LD := "Tag_LoadPV"
      PV := "Tag_PresetValue"
     QU => "Tag_CounterStatusUP"
      QD => "Tag_CounterStatusDOWN"
      CV => "Tag_CounterValue"
      )
```

「Tag_StartCTU」または「Tag_StartCTD」入力のシグナル状態が「0」から「1」に切り替わった場合(信号立ち上がりエッジ)、「カウントアップ/カウントダウン」命令が実行されます。「Tag_StartCTU」入力で立ち上がりエッジが発生する場合、カウンタ現在値は 1 ずつインクリメントされ、「Tag_CounterValue」出力に保存されます。 「Tag_StartCTD」入力で立ち上がりエッジが発生する場合、カウンタ値は 1 ずつデクリメントされ、「Tag_CounterValue」出力に保存されます。 CU 入力で立ち上がりエッジが発生した場合、カウンタ値が上限値 32767 に達するまでインクリメントされます。CD 入力に立ち上がりエッジがある場合、カウンタ値が下限値 INT = -32768 に達するまでデクリメントされます。

カウンタ現在値が「Tag_PresetValue」入力の値以上である限り、「Tag_CounterStatusUP」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_CounterStatusUP」出力のシグナル状態は「0」になります。

カウンタ現在値がゼロ以下である限り、「Tag_CounterStatusDOWN」出力のシグナル状態は「1」になります。 それ以外のすべての場合、「Tag_CounterStatusDOWN」出力のシグナル状態は「0」になります。

四則演算

この章には下記に関する情報が記載されています：

NEG: 2 の補数の作成

説明

「2 の補数の作成」命令を使用して、オペランド値の符号を変更できます。 たとえば、値が正の場合は、この値と同値の負の数が出力されます。

OUT 出力の値は、命令の結果が OUT 出力で指定されたデータタイプに許可された範囲外である場合、無効です(整数についてのみ有効)。

構文

「2 の補数の作成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := NEG_???(<Operand>)
```

パラメータ

次の表に、「2 つの補数の作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P、または定数	入力値
<Result>	Output	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	入力値の 2 の補数

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := NEG_REAL("Tag_InValue")
```

この命令は、「Tag_InValue」オペランドの値の符号を変更し、結果を「Tag_OutValue」オペランドに出力します。

ABS: 絶対値の形成

説明

「絶対値の形成」命令を使用して、オペランド値の絶対値を計算します。

浮動小数点数に無効な値が含まれている場合、結果の値は無効です。

構文

「絶対値の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := ABS(<Operand>)
```

パラメータ

次の表に、「絶対値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P、または定数	入力値
<Result>	Output	SINT、INT、DINT、LINT、浮動小数点数	I、Q、M、D、L、P	入力値の絶対値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := ABS("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	-6.234
Tag_OutValue	6.234

この命令は、「Tag_InValue」オペランドの値の絶対値を計算し、結果を「Tag_OutValue」オペランドに出力します。

MIN: 最小値の取得

説明

「最小値の取得」命令を使用して、使用可能な IN1 および IN2 入力の値を比較し、最小値を OUT 出力に書き込みます。

OUT 出力の値は、次の条件の 1 つが満たされると無効です。

- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

構文

以下の構文が「最小値の取得」命令に使用されます。

```
GRAPH
CALL MIN ???
      (IN1 := <operand>
       IN2 := <operand>
        OUT => <Operand>
       )
```

パラメータ

次の表に、「最小値の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	2 番目の入力値
OUT	Output	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL MIN INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      OUT => "Tag_Minimum"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
OUT	Tag_Minimum	12222

この命令は、指定されたオペランドの値を比較し、最小値(「TagIn_Value1」)を「Tag_Minimum」出力にコピーします。

MAX: 最大値の取得

説明

「最大値の取得」命令を使用して、IN1 および IN2 入力の値を比較し、最大値を OUT 出力に書き込みます。

OUT 出力の値は、次の条件の 1 つが満たされると無効です。

- 命令の実行中にデータタイプの暗黙的な変換に失敗すること。
- 浮動小数点数の値が無効であること。

構文

以下の構文が「最大値の取得」命令に使用されます。

```
GRAPH
CALL MAX ???
      (IN1 := <operand>
       IN2 := <operand>
        OUT => <Operand>
       )
```

パラメータ

次の表に、「最大値の取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	最初の入力値
IN2	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	2 番目の入力値
OUT	Output	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL MAX INT
     (IN1 := "TagIn_Value1"
      IN2 := "TagIn_Value2"
      OUT => "Tag_Maximum"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN1	TagIn_Value1	12222
IN2	TagIn_Value2	14444
OUT	Tag_Maximum	14444

この命令は、指定されたオペランドの値を比較し、最大値(「TagIn_Value2」)を「Tag_Maximum」出力にコピーします。

LIMIT: 制限値の設定

説明

「制限値のセット」命令を使用して、IN 入力の値を MN および MX 入力の値に制限します。 IN 入力の値が条件 MN <= IN <= MX を満たす場合、この値は OUT 出力にコピーされます。 条件が満たされず、IN 入力値が下限値 MN を下回る場合、OUT 出力は MN 入力の値にセットされます。 上限値 MX を超える場合、OUT 出力は MX 入力の値にセットされます。

MN 入力の値が MX 入力の値よりも大きい場合、結果が未定義なります。 命令は、すべての入力のタグが同じデータタイプの場合にのみ実行されます。

OUT 出力の値は、次の条件の 1 つが満たされると無効です。

- 指定された複数のタグが同じデータタイプでないこと。
- オペランドの値が無効である。
- MN パラメータの値が MX パラメータの値よりも大きいこと。

構文

以下の構文が「制限値の設定」命令に使用されます。

```
GRAPH
CALL LIMIT ???
      (MN  := <operand>
       IN  := <operand>
       MX  := <operand>
        OUT => <Operand>
       )
```

パラメータ

次の表に、「制限値の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MN	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	下限値
IN	Input	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	入力値
MX	Input	整数、浮動小数点数、TIME、LTIME、TOD、	I、Q、M、D、L、P、または定数	上限値

		LTOD、DATE、LDT		
OUT	Output	整数、浮動小数点数、TIME、LTIME、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	結果
データタイプ TOD、LTOD、DATE、および LDT は、IEC テストが有効でない場合のみ使用できます。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL LIMIT_INT
     (MN := "Tag_LowLimit"
      IN := "Tag_InputValue"
      MX := "Tag_HighLimit"
      OUT => "Tag_Result"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MN	Tag_LowLimit	12000
IN	Tag_InputValue	8000
MX	Tag_HighLimit	16000
OUT	Tag_Result	12000

「Tag_InputValue」オペランドの値が「Tag_LowLimit」および「Tag_HighLimit」オペランドの値と比較されます。「Tag_InputValue」 オペランドの値が下限値未満であるため、「Tag_LowLimit」オペランドの値が「Tag_Result」出力にコピーされます。

SQR: 2 乗値の形成

説明

「2 乗値の形成」命令を使用して、オペランドの浮動小数点数の値を二乗し、結果を出力に書き込みます。

オペランドに無効な浮動小数点数が含まれている場合、結果の値は無効です。

構文

「2 乗値の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := SQR(<Operand>)
```

パラメータ

次の表に、「2 乗値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	入力値の 2 乗値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := SQR("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値
Tag_InValue	5.0
Tag_OutValue	25.0

この命令は、「Tag_InValue」オペランドの値を 2 乗し、結果を「Tag_OutValue」オペランドに出力します。

SQRT: 平方根の形成

説明

「平方根の形成」命令を使用して、入力値の平方根を計算し、結果を指定したオペランド内に保存します。この命令は、入力値がゼロよりも大きい場合に正の結果になります。 入力値がゼロ未満の場合、この命令は無効な浮動小数点数を返します。 オペランドの値が「0」の場合、結果も「0」になります。

構文

「平方根の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := SQRT(<Operand>)
```

パラメータ

次の表に、「平方根の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
<Result>	Output	浮動小数点数	I、Q、M、D、L	入力値の平方根

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := SQRT("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値
Tag_InValue	25.0
Tag_OutValue	5.0

この命令は、「Tag_InValue」オペランドの値の平方根を計算し、結果を「Tag_OutValue」オペランドに出力します。

LN: 自然対数の形成

説明

「自然対数の形成」命令を使用して、入力値の底 e (e=2.718282)に対する自然対数を計算します。この命令は、入力値がゼロよりも大きい場合に正の結果になります。 入力値がゼロ未満の場合、この命令は無効な浮動小数点数を返します。

構文

「自然対数の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := LN(<Operand>)
```

パラメータ

次の表に、「自然対数の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	入力値の自然対数

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := LN("Tag_InValue")
```

この命令は、「Tag_InValue」オペランドの値の自然対数を計算し、結果を「Tag_OutValue」オペランドに出力します。

EXP: 指数値の形成

説明

「指数値の形成」命令を使用して、底 e (e = 2,718282)およびオペランドの値から指数を計算します。

構文

「指数値の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := EXP(<Operand>)
```

パラメータ

次の表に、「指数値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	入力値
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	入力値の指数値

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := EXP("Tag_InValue")
```

この命令は、底 e および「Tag_InValue」オペランドの値から指数を計算し、結果(e^{IN})を「Tag_Out-Value」オペランドに出力します。

SIN: 正弦値の形成

説明

「正弦値の形成」命令を使用して、角度の正弦を計算します。 角度のサイズが、オペランドでラジアン単位で指定されます。

結果の値は、オペランドが有効な浮動小数点数でない場合、無効です。

構文

「正弦値の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := SIN(<Operand>)
```

パラメータ

次の表に、「正弦値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	ラジアン単位の角度のサイズ
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	指定された角度の正弦

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := SIN("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	1.0

この命令は、「Tag_InValue」オペランドで指定された角度の正弦を計算し、結果を「Tag_OutValue」オペランドに出力します。

COS: 余弦値の形成

説明

「余弦値の形成」命令を使用して、角度の余弦を計算します。 角度のサイズが、オペランドでラジアン単位で指定されます。

結果の値は、オペランドが有効な浮動小数点数でない場合、無効です。

構文

「余弦値の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := COS(<Operand>)
```

パラメータ

次の表に、「余弦値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	ラジアン単位の角度のサイズ
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	指定された角度の余弦

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := COS("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	+1.570796 (π/2)
Tag_OutValue	0

この命令は、「Tag_InValue」オペランドで指定された角度の余弦を計算し、結果を「Tag_OutValue」オペランドに出力します。

TAN: 正接値の形成

説明

「正接値の形成」命令を使用して、角度の正接を計算します。 角度のサイズが、オペランドでラジアン単位で指定されます。

結果の値は、オペランドが有効な浮動小数点数でない場合、無効です。

構文

「正接値の形成」命令には、次の構文を使用します。

```
GRAPH
<Result> := TAN(<Operand>)
```

パラメータ

次の表に、「正接値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	ラジアン単位の角度のサイズ
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	指定された角度の正接

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := TAN("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	+3.141593 (π)
Tag_OutValue	0

この命令は、「Tag_InValue」オペランドで指定された角度の正接を計算し、結果を「Tag_OutValue」オペランドに出力します。

ASIN: 逆正弦値の形成

説明

「逆正弦値の形成」命令を使用して、オペランドで指定された正弦値に対応する角度のサイズを計算します。 範囲-1 ~ +1 の有効な浮動小数点数のみをオペランドに指定できます。 計算された角度サイズは、-π/2 ~ +π/2 の値の範囲にすることができます。

次のいずれかの条件が満たされる場合、結果の値は無効です。

- オペランドの値が有効な浮動小数点数でないこと。
- オペランドの値が許容値の範囲(-1 ~ +1)外であること。

構文

「逆正弦値の形成」命令には、次の構文を使用します。

```
GRAPH
<Result> := ASIN(<Operand>)
```

パラメータ

次の表に、「逆正弦値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	正弦値
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := ASIN("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	1.0
Tag_OutValue	+1.570796 (π/2)

この命令は、「Tag_InValue」オペランドの正弦値に対応する角度のサイズを計算します。 命令の結果は、「Tag_OutValue」オペランドに格納されます。

ACOS: 逆余弦値の形成

説明

「逆余弦値の形成」命令を使用して、オペランドで指定された余弦値に対応する角度のサイズを計算します。 範囲-1 ~ +1 の有効な浮動小数点数のみをオペランドに指定できます。 計算された角度サイズは、0 ~ +π の値の範囲にすることができます。

次のいずれかの条件が満たされる場合、結果の値は無効です。

- オペランドの値が有効な浮動小数点数でないこと。
- オペランドの値が許容値の範囲(-1 ~ +1)外であること。

構文

「逆余弦値の形成」命令には、次の構文を使用します。

```
GRAPH
<Result> := ACOS(<Operand>)
```

パラメータ

次の表に、「逆余弦値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	余弦値
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := ACOS("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値
Tag_InValue	0
Tag_OutValue	+1.570796 (π/2)

この命令は、「Tag_InValue」オペランドの余弦値に対応する角度のサイズを計算します。 命令の結果は、「Tag_OutValue」オペランドに格納されます。

ATAN: 逆正接値の形成

説明

「逆正接値の形成」命令を使用して、オペランドで指定された正接値に対応する角度のサイズを計算します。 オペランドには、有効な浮動小数点数(または-NaN/+NaN)のみを指定することができます。 計算された角度サイズは、-π/2 ~ +π/2 の値の範囲にすることができます。

構文

「逆正接値の形成」命令には、以下の構文を使用します。

```
GRAPH
<Result> := ATAN(<Operand>)
```

パラメータ

次の表に、「逆正接値の形成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	正接値
<Result>	Output	浮動小数点数	I、Q、M、D、L、P	ラジアン単位の角度のサイズ

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := ATAN("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値
Tag_InValue	1.0
Tag_OutValue	+0.785398 (π/4)

この命令は、「Tag_InValue」オペランドの正接値に対応する角度のサイズを計算します。 命令の結果は、「Tag_OutValue」オペランドに格納されます。

FRAC: 小数部を返す

説明

「小数部を返す」命令を使用して、オペランドの値の小数位を求めることができます。 たとえば、オペランドに値 123.4567 が含まれている場合、結果は値 0.4567 を返します。

結果の値は、オペランドが有効な浮動小数点数でない場合、無効です。

構文

「小数部を返す」命令には、以下の構文を使用します。

```
GRAPH
CALL FRAC ???
      (IN := <operand>
       RET_VAL => <Operand>
       )
```

パラメータ

次の表に、「小数部を返す」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	浮動小数点数	I、Q、M、D、L、P、または定数	小数位を求める入力値。
RET_VAL	Output	浮動小数点数	I、Q、M、D、L、P	入力値の小数位

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL FRAC REAL
      (IN := "Tag_InValue"
       RET_VAL => "Tag_OutValue"
       )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	2.555
Tag_OutValue	0.555

この命令は、「Tag_InValue」オペランドの値の小数位を「Tag_OutValue」オペランドにコピーします。

ムーブ操作

この章には下記に関する情報が記載されています：

- Deserialize: 逆シリアル化 (S7-1500)
- Serialize: シリアル化 (S7-1500)
- MOVE_BLK: ブロックの移動 (S7-1500)
- MOVE_BLK_VARIANT: ブロックの移動 (S7-1500)
- UMOVE_BLK: 割り込みなしブロック転送 (S7-1500)
- FILL_BLK: フィル命令 (S7-1500)
- UFILL_BLK: 割り込みなしフィル命令 (S7-1500)
- SWAP:スワップ (S7-1500)
- 配列 DB (S7-1500)
- レガシー (S7-1500)

Deserialize: 逆シリアル化

説明

「逆シリアル化」命令を使用して、PLC データタイプ(UDT)のシーケンシャル表示を PLC データタイプに変換して戻し、その内容全体を表示します。

PLC データタイプのシーケンシャル表示を配置するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。標準メモリ領域の容量は 64 KB です。変換の前に、十分なメモリ領域が存在することを確認してください。

この命令では、変換された PLC データタイプの複数のシーケンシャル表示をそれらの元の状態に、順を追って変換し直すことができます。

PLC データタイプ(UDT)の単一のシーケンシャル表示のみを変換して元に戻す場合は、命令「TRCV: 通信接続経由のデータ受信」を直接に使用することもできます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

構文

「逆シリアル化」命令には、以下の構文を使用します。

```
GRAPH
CALL Deserialize VARIANT
     (SRC_ARRAY := <operand>
      DEST_VARIABLE => <Operand>
      POS := <operand>
      RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「逆シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_ARRAY	Input	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるグローバルデータブロック
DEST_VARIABLE	InOut	VARIANT	I、Q、M、L	返された変換済み PLC データタイプ(UDT)が格納されるタグ
POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS

				パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_ARRAY パラメータと DEST_VARIABLE パラメータのためのメモリ領域がオーバーラップしています。
8136	DEST_VARIABLE パラメータのデータブロックが標準アクセスによるブロックではありません。
8150	SRC_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8151	SRC_ARRAY パラメータでコード生成エラーが発生しました
8153	SRC_ARRAY パラメータのメモリの空き領域が不足しています。
8250	DEST_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8251	DEST_VARIABLE パラメータでコード生成エラーが発生しました
8254	DEST_VARIABLE パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL Deserialize VARIANT
     (SRC_ARRAY := "Buffer".Field
      DEST_VARIABLE => "Target".Client
      POS := #BufferPos
      RET_VAL => #Error
      )
```

この命令は、「Buffer」データブロックのカスタマデータのシーケンシャル表示を逆シリアル化し、「Target」データブロックに書き込みます。 変換済みカスタマデータで使用されるバイト数は、#BufferPos オペランドに格納されます。

Serialize: シリアル化

説明

「シリアル化」命令を使用して、データタイプの構造体の部分を失うことなく、複数の PLC データタイプ(UDT)を 1 つのシーケンシャル表示に変換します。

この命令を使用して、ユーザプログラムの複数の構造体データ項目を一時的にバッファ(グローバルデータブロックを推奨)に保存し、別の CPU に送信します。 変換済み PLC データタイプを格納するメモリ領域には、バイトデータタイプの配列が存在し、メモリ領域は標準アクセスで宣言する必要があります。標準メモリ領域の容量は 64 KB です。 変換の前に、十分なメモリ領域が存在することを確認してください。

POS パラメータのオペランドには、変換済み PLC データタイプによって使用されるバイト数に関する情報が収納されています。

単一の PLC データタイプ(UDT)を送信する場合、直接に命令「TSEND: 通信接続経由のデータ送信」を呼び出すことができます。

注記

S7-1500 シリーズの CPU への適用事項

ブロックプロパティ「最適化したブロックアクセス」が付加されているブロックでは、ビットの長さは 1 バイトです。

構文

「シリアル化」命令には、以下の構文を使用します。

```
GRAPH
CALL Serialize VARIANT
     (SRC_VARIABLE := <operand>
      DEST_ARRAY => <Operand>
      POS := <operand>
      RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「シリアル化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC_VARIABLE	Input	VARIANT	I、Q、M、L	シーケンシャル表示に変換される PLC データタイプ (UDT)。
DEST_ARRAY	InOut	VARIANT	I、Q、M、L	生成されたデータストリームが格納されるデータブロック。

POS	InOut	DINT	I、Q、M、D、L	変換済み PLC データタイプが使用するバイト数。 POS パラメータは、ゼロベースで計算されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B0	SRC_VARIABLE パラメータと DEST_ARRAY パラメータのためのメモリ領域がオーバーラップしています。
8150	SRC_VARIABLE パラメータの VARIANT データタイプに値が含まれていません。
8152	SRC_VARIABLE パラメータでコード生成エラーが発生しました
8236	DEST_ARRAY パラメータのデータブロックが標準アクセスによるブロックではありません。
8250	DEST_ARRAY パラメータの VARIANT データタイプに値が含まれていません。
8252	DEST_ARRAY パラメータでコード生成エラーが発生しました
8253	DEST_ARRAY パラメータのメモリの空き領域が不足しています。
8254	DEST_ARRAY パラメータの無効なデータタイプ
8382	POS パラメータの値が、配列の限界値外です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL Serialize VARIANT
     (SRC_VARIABLE := "Source".Client
      DEST_ARRAY => "Buffer".Field
      POS := #BufferPos
      RET_VAL => #Error
      )
```

この命令は、「Source」データブロックのカスタマデータをシリアル化し、シーケンシャル表示で「Buffer」データブロックに書き込みます。 シーケンシャル表示によって占有されたバイト数は、#BufferPos オペランドに格納されます。

MOVE_BLK: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動します。 宛先領域にコピーするエレメントの数は、COUNT パラメータで指定できます。移動されるエレメントの幅は、IN パラメータのエレメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

出力 OUT の値は、入力 IN または出力 OUT で使用可能にされたよりも多くのデータがコピーされた場合、無効です。

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
GRAPH
CALL MOVE_BLK ???_???
      (IN := <operand>
       COUNT := <operand>
       OUT => <Operand>
       )
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	コピーされるソース領域の最初のエレメント
COUNT	Input	USINT、UINT、UDINT、ULINT	I、Q、M、D、L、P、または定数	ソース領域から宛先領域にコピーするエレメントの数。
OUT [1)]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント
[1)] 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL MOVE_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。

MOVE_BLK_VARIANT: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動します。配列全体または配列のエレメントを同じデータタイプの別の配列にコピーすることができます。ソース配列と宛先配列のサイズ(エレメントの数)は異なる場合があります。配列内の複数エレメントまたは単一エレメントをコピーすることができます。

ブロックの作成時にこの命令を使用する場合、VARIANT ごとにソースおよび宛先が転送されるため、配列はまだ既知である必要はありません。

SRC_INDEX および DEST_INDEX パラメータでのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

コピーされるデータが使用可能なデータよりも多い場合、この命令は実行されません。

注記

BOOL データタイプと接続している VARIANT

VARIANT データタイプのパラメータ(ソース領域または宛先領域)と BOOL データタイプのタグまたは ARRAY of BOOL を相互接続する場合、以下のオプションがあります。

1. シンボリックにアドレス指定することができます。

   例:SRC パラメータ:"Data_block".myArray
2. 任意のポインタによってアドレス指定することができます。ただし、領域で指定された長さは 8 で割り切れるようにする必要があります。割り切れない場合、この命令は実行されません。

   例:SRC パラメータ:P#DB123.DBX456.0 BOOL 1000

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
GRAPH
CALL MOVE_BLK_VARIANT
     (SRC := <operand>
      COUNT := <operand>
      SRC_INDEX := <operand>
      DEST_INDEX := <operand>
      DEST => <Operand>
      RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRC	Input	VARIANT (配列または個々の配列エレメントをポイントする)、<Data_type>の配列	I、Q、M、L	コピー元のソースブロック
COUNT	Input	UDINT	I、Q、M、D、L、または定数	コピーされるエレメント数 SRC が DEST パラメータまたは ARRAY パラメータで指定されていない場合は、COUNT パラメータに値「1」を割り当てます。
SRC_INDEX	Input	DINT	I、Q、M、D、L、または定数	• SRC_INDEX パラメータは、ゼロベースで計算されます。SRC がパラメータ ARRAY で指定されると、パラメータ SRC_INDEX の整数はコピー元のソース領域内の最初のエレメントを指定します。宣言された配列の限界値には関係ありません。 • SRC がパラメータ Array で指定されないか、または配列の 1 つの単一エレメントのみが指定された場合、パラメータ SRC_INDEX に値「0」を割り当てます。
DEST_INDEX	Input	DINT	I、Q、M、D、L、または定数	• DEST_INDEX パラメータは、ゼロベースで計算されます。DEST がパラメータ ARRAY で指定されると、パラメータ DEST_INDEX の整数はコピー先の宛先領域内の最初のエレメントを指定します。宣言された配列の限界値には関係ありません。 • DEST が ARRAY パラメータで指定されていない場合は、DEST_INDEX パラメータに値「0」を割り当てます。
DEST	Output [1)]	VARIANT	I、Q、M、L	ソースブロックの内容がコピーされる宛先領域。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

1) データがタグに流れるため、DEST パラメータは Output として宣言されます。ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	データタイプが一致していません
8151	SRC パラメータにアクセスできません。
8152	SRC パラメータのオペランドが入力されていません。
8153	SRC パラメータでコード生成エラーが発生しました
8154	SRC パラメータのオペランドのデータタイプが BOOL です。
8281	COUNT パラメータの値が無効です
8382	SRC_INDEX パラメータの値が VARIANT の限界値外です。
8383	SRC_INDEX パラメータの値が配列の上限値外です
8482	DEST_INDEX パラメータの値が VARIANT の限界値外です。
8483	DEST_INDEX パラメータの値が配列の上限値外です
8534	DEST パラメータが書き込み保護されています
8551	DEST パラメータにアクセスできません。
8552	DEST パラメータのオペランドが入力されていません。
8553	DEST パラメータでコード生成エラーが発生しました
8554	DEST パラメータのオペランドのデータタイプが BOOL です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL MOVE_BLK_VARIANT
     (SRC := #SrcField
      COUNT := "Tag_Count"
      SRC_INDEX := "Tag_Src_Index"
      DEST_INDEX := "Tag_Dest_Index"
      DEST => #DestField
      RET_VAL := "Tag_Result"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	ブロックインターフェースでの宣言	オペランド	値
SRC	Input	#SrcField	ローカルオペランド #SrcField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。("MOVE_UDT"の ARRAY[0..10]
COUNT	Input	Tag_Count	2
SRC_INDEX	Input	Tag_Src_Index	3
DEST_INDEX	Input	Tag_Dest_Index	3
DEST	InOut	#DestField	ローカルオペランド #DestField は、ブロックのプログラミング時にはまだ不明であった PLC データタイプを使用します。("MOVE_UDT"の ARRAY[10..20]

UDT の配列の 4 番目の工レメントから始めて、2 つの工レメントが、ソース領域から宛先領域に移動されます。コピーが、UDT の配列の 4 番目のエレメントから始めて、挿入されます。

UMOVE_BLK: 割り込みなしブロック転送

説明

「割り込み不可能なブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動します。 この命令に割り込みをかけることはできません。 宛先領域にコピーする工レメントの数は、COUNT パラメータで指定できます。移動される工レメントの幅は、IN パラメータの工レメントの幅で定義されます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記

割り込みなしブロック転送

他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。このため、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

出力 OUT の値は、入力 IN または出力 OUT で使用可能にされたよりも多くのデータがコピーされた場合、無効です。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

構文

以下の構文が「割り込みなしブロック転送」命令に使用されます。

```
GRAPH
CALL UMOVE_BLK ???_???
     (IN := <operand>
      COUNT := <operand>
      OUT => <Operand>
      )
```

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1)]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	コピーされるソース領域の最初の工レメント
COUNT	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	ソース領域から宛先領域にコピーする工レメントの数。

OUT [1]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	ソース領域の内容がコピーされる宛先領域の最初のエレメント
[1] 指定されたデータタイプは、ARRAY 構造体のエレメントとしてしか使用できません。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL UMOVE_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、3 番目のエレメントから開始して、#a_array タグから 3 つの INT エレメントを選択し、2 番目のエレメントで開始して、その内容を#b_array 出力タグにコピーします。 他のオペレーティングシステムのアクティビティによって、コピー操作に割り込みをかけることはできません。

FILL_BLK: フィル命令

説明

「フィル」命令を使用して、IN パラメータの値でメモリ領域(宛先領域)を埋めます。 宛先領域は、OUT パラメータで指定されたアドレスから開始して埋められます。 コピー操作の繰り返し回数は、COUNT パラメータの値で指定されます。 命令が実行されると、IN パラメータの値が選択され、COUNT パラメータの値で指定された回数だけコピー先の領域にコピーされます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

出力 OUT の値は、入力 IN または出力 OUT で使用可能にされたよりも多くのデータがコピーされた場合、無効です。

構文

「フィル」命令には、以下の構文を使用します。

```
GRAPH
CALL FILL_BLK ???_???
      (IN := <operand>
       COUNT := <operand>
       OUT => <Operand>
       )
```

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT、UINT、UDINT、ULINT	I、Q、M、D、L、P、または定数	コピー操作の繰り返し回数
OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	D、L	フィル命令が開始する宛先領域のアドレス
[1] 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。 [2] 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL FILL_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つのエレメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つのエレメントで構成されます。

この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出力タグに 3 回コピーします。

UFILL_BLK: 割り込みなしフィル命令

説明

「割り込み不可フィル」命令を使用して、IN パラメータの値でメモリ領域(宛先領域)を埋めます。 この命令に割り込みをかけることはできません。 宛先領域は、OUT パラメータで指定されたアドレスから開始して埋められます。 コピー操作の繰り返し回数は、COUNT パラメータの値で指定されます。 命令が実行されると、IN パラメータの値が選択され、COUNT パラメータの値で指定された回数だけコピー先の領域にコピーされます。

この命令は、ソース領域と宛先領域が同じデータタイプの場合にのみ実行できます。

注記
他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。このため、「割り込み不可能なフィル」命令の実行中には CPU の割り込み応答時間が長くなります。

出力 OUT の値は、入力 IN または出力 OUT で使用可能にされたよりも多くのデータがコピーされた場合、無効です。

「割り込み不可フィル」命令を使用して最大 16 KB 移動することができます。 この場合、CPU 固有の制限に注意してください。

構文

「割り込み不可能なフィル」命令には、以下の構文を使用します。

```
GRAPH
CALL UFILL_BLK ???_???
     (IN := <operand>
      COUNT := <operand>
      OUT => <Operand>
      )
```

パラメータ

次の表に、「割り込み不可能なフィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN [1]	Input	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、DATE、CHAR、WCHAR	I、Q、M、D、L、P、または定数	宛先領域を埋めるために使用する工レメント
COUNT	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、P、または定数	コピー操作の繰り返し回数
OUT [2]	Output	2 進数、整数、浮動小数点数、タイマ、TOD、LTOD、	D、L	フィル命令が開始する宛先領域のアドレス

		DATE、CHAR、WCHAR		

1) 指定されたデータタイプは、ARRAY 構造体の工レメントとして使用することもできます。

2) 指定されたデータタイプは、ARRAY 構造体の工レメントとしてしか使用できません。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL UFILL_BLK INT_UINT
     (IN := #a_array[2]
      COUNT := "Tag_Count"
      OUT => #b_array[1]
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
IN	a_array[2]	a_array オペランドのデータタイプは ARRAY [0..5] of INT です。これは、INT データタイプの 6 つの工レメントで構成されます。
COUNT	Tag_Count	3
OUT	b_array[1]	b_array オペランドのデータタイプは ARRAY [0..6] of INT です。これは、INT データタイプの 7 つの工レメントで構成されます。

この命令は、#a_array タグの 3 番目のエレメント(#a_array[2])を#b_array 出カタグに 3 回コピーします。 他のオペレーティングシステムのアクティビティによって、コピー操作に割り込みをかけることはできません。

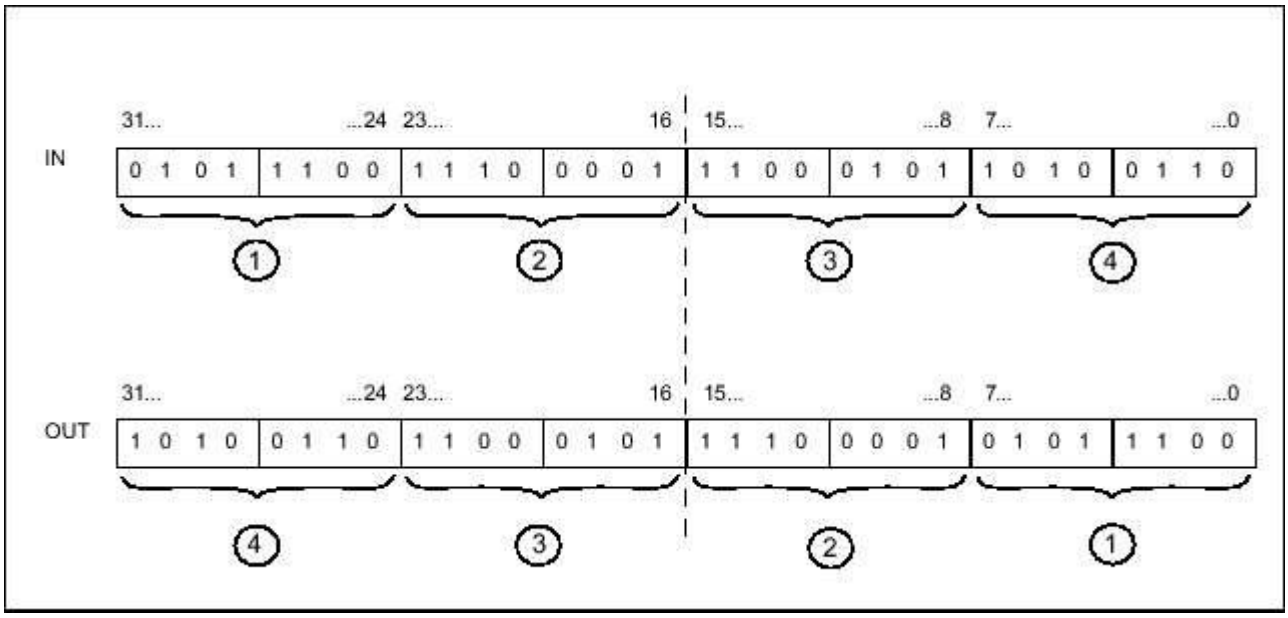
SWAP:スワップ

説明

「スワップ」命令を使用して、オペランドのバイトの順序を変更します。

使用されるデータタイプに応じて、アキュムレータ 1 のすべてのバイトまたはアキュムレータ 1 の右のワードのバイトのみをスワップすることができます。

次の図に、データタイプ DWORD のオペランドのバイトをスワップする方法を示します。



アキュムレータ 1 の右ワードのバイトの順序

データタイプ WORD の場合、アキュムレータ 1 の右ワードのバイトをスワップします。

次の表に、命令の実行前および実行後のアキュムレータ 1 の内容を示します。

ステータス	アキュムレータ 1 のバイト			
実行前	A	B	C	D
実行後	A	B	D	C

命令の結果はアキュムレータ 1 の右ワードに保存されます。アキュムレータ 1 の左ワードのバイトは命令に影響されず、変わりません。

構文

「スワップ」命令には、以下の構文を使用します。

```
GRAPH
<Result> := SWAP(<Operand>)
```

パラメータ

次の表に、「スワップ」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	WORD, DWORD, LWORD	I、Q、M、D、L、または定数	バイトがスワップされるオペランド
<Result>	Output	WORD, DWORD, LWORD	I、Q、M、D、L	結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"Tag_OutValue" := SWAP("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値			
Tag_InValue	0000	1111	0000	1111
Tag_OutValue	0000	1111	1111	0000

アキュムレータ 1 のすべてのバイトをスワップ

データタイプ DWORD の場合、アキュムレータ 1 のすべてのバイトをスワップします。

次の表に、命令の実行前および実行後のアキュムレータ 1 の内容を示します。

ステータス	アキュムレータ 1 のバイト			
実行前	A	B	C	D
実行後	D	C	B	A

命令の結果は、アキュムレータ 1 に保存されます。アキュムレータ 2 の内容は、変わりません。

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値							
Tag_InValue	1111	0000	0000	1111	0000	0000	1111	1111
Tag_OutValue	1111	1111	0000	0000	0000	1111	1111	0000

「Tag_InValue」オペランドのバイトがスワップされ、「Tag_OutValue」オペランドに格納されます。

配列 DB

この章には下記に関する情報が記載されています：

- ReadFromArrayDB: 配列データブロックからの読み出し (S7-1500)
- WriteToArrayDB: 配列データブロックへの書き込み (S7-1500)
- ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し (S7-1500)
- WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み (S7-1500)

ReadFromArrayDB: 配列データブロックからの読み出し

説明

「配列データブロックからの読み出し」命令を使用し、配列データブロックからデータを読み出して、宛先領域に書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

構文

「ARRAY データブロックからの読み出し」命令には、以下の構文を使用します。

```
GRAPH
CALL ReadFromArrayDB
     (DB := <operand>
      INDEX := <operand>
      VALUE => <Operand>
      RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	Output [1)]	VARIANT	I、Q、M、L	読み出されて出力される値
RET_VAL	Output	INT	I、Q、M、D、L、P	命令の結果
1) データがタグに流れるため、VALUE パラメータは Output として宣言されます。 ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード*	説明

(W#16#...)	
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、書き込み保護されているか、またはロードメモリ内に存在します。
8135	配列データブロックに無効な値が含まれます。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8450	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8452	コード生成エラー
8453	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL ReadFromArrayDB
     (DB := "ArrayDB"
      INDEX := "ArrayDB".THIS[2]
      VALUE => "TargetField[10]".Data2[1]
      RET_VAL => "TagResult"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"TargetField[10]".Data2[1]	「TargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[10 to 20]です。

3 番目のエレメントは「ArrayDB」から読み出され、"TargetField[10]".Data2[1]オペランドに書き込まれます。

WriteToArrayDB: 配列データブロックへの書き込み

説明

「配列データブロックへの書き込み」命令を使用して、配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。 配列の エレメントは、PLC データタイプまたはその他の基本データタイプにすることができます。 配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

構文

「配列データブロックへの書き込み」命令には、以下の構文を使用します。

```
GRAPH
CALL WriteToArrayDB
      (DB := <operand>
       INDEX := <operand>
       VALUE := <Operand>)
       RET_VAL => <Operand>
       )
```

パラメータ

次の表に、「配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内のエレメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値
RET_VAL	Output	INT	I、Q、M、D、L、P	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
80B5	コピーが割り込まれました。
8132	データブロックが存在しないか、小さすぎるか、またはロードメモリ内に存在します。
8134	データブロックが書き込み保護されています。
8135	データブロックが配列データブロックではありません。
8154	データブロックのデータタイプが不正です。
8282	INDEX パラメータの値が、配列の限界値外です。
8350	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8352	コード生成エラー
8353	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL WriteToArrayDB
     (DB := "ArrayDB"
      INDEX := "ArrayDB".THIS[2]
      VALUE := "SourceField[1]".Data1[6]
      RET_VAL => "TagResult"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceField[1]".Data1[6]	「SourceField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。

「SourceField」オペランドから、2 番目のエレメントの「Data1[6]」エレメントが「ArrayDB」に書き込まれます。 3 番目のエレメントは、「ArrayDB」に書き込まれます。

ReadFromArrayDBL: ロードメモリの配列データブロックからの読み出し

説明

「ロードメモリの配列データブロックからの読み出し」命令を使用して、ロードメモリの配列データブロックからデータを読み出します。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。配列の工レメントは、PLC データタイプまたはその他の基本データタイプにすることができます。配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 この命令は、BUSY パラメータで信号立ち下がりエッジが検出された場合に終了します。1 プログラムサイクルの間、DONE パラメータのシグナル状態が「1」になり、このサイクル内に、読み取られた値が VALUE パラメータに出力されます。 他のすべてのプログラムサイクルでは、VALUE パラメータの値は変更されません。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

構文

「ロードメモリの ARRAY データブロックからの読み出し」命令には、以下の構文を使用します。

```
GRAPH
CALL ReadFromArrayDBL DB_ANY, "インスタンス DB"
      (REQ := <operand>
       DB := <operand>
       INDEX := <operand>
       VALUE := <operand>
       BUSY => <Operand>
       DONE => <Operand>
       ERROR => <Operand>
       )
```

パラメータ

次の表に、「ロードメモリの配列データブロックからの読み出し」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」:配列 DB の読み出しから開始
DB	Input	DB_ANY	I、Q、M、D、L	読み出されるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	読み出されるエレメント
VALUE	InOut	VARIANT	I、Q、M、L	読み出されて出力される値 TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ読み出し中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。	

命令「READ_DBL:ロードメモリのデータブロックからの読み出し」および「WRIT_DBL:データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL ReadFromArrayDBL DB_ANY, "ReadFromArrayDBL_DB"
     (REQ := "TagReg"
      DB := "ArrayDB"
      INDEX := "ArrayDB".THIS[2]
      VALUE := "SourceTargetField[1]".Data1[6]
      BUSY => "TagBusy"
      DONE => "TagDone"
      ERROR => "TagError"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceTargetField[1]".Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagReq」オペランドで立ち上がりエッジが検出されると、「ロードメモリの ARRAY データブロックからの読み出し」命令が実行されます。 3 番目のエレメントは「ArrayDB」から読み出され、"SourceTargetField[1]".Data1[6]オペランドに出力されます。 「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値はもう変更されません。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み

説明

「ロードメモリの配列データブロックへの書き込み」命令を使用して、ロードメモリの配列データブロックにデータを書き込みます。

配列データブロックは、1 つの[データタイプ]配列のみで構成されるデータブロックです。配列の工レメントは、PLC データタイプまたはその他の基本データタイプにすることができます。配列でのカウントは、配列の後の宣言に関係なく、常に下限値「0」から開始します。

配列データブロックに「ロードメモリにのみ保存」ブロック属性が指定されている場合は、ロードメモリにのみ保存されます。

この命令は、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 BUSY パラメータのシグナル状態が「1」になります。 BUSY パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、VALUE パラメータの値がデータブロックに書き込まれます。 DONE1 パラメータで信号立ち下がりエッジが検出された場合、命令は終了し、パラメータの値がデータブロックに書き込まれます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。このトピックの詳細については、「関連項目」を参照してください。

注記
配列データブロックは、「最適化済み」ブロックプロパティを使用して作成する必要があります。

構文

「ロードメモリの ARRAY データブロックへの書き込み」命令には、以下の構文を使用します。

```
GRAPH
CALL WriteToArrayDBL DB_ANY, "インスタンス DB"
      (REQ := <operand>
       DB := <operand>
       INDEX := <operand>
       VALUE := <operand>
       BUSY => <Operand>
       DONE => <Operand>
       ERROR => <Operand>
       )
```

パラメータ

次の表に、「ロードメモリの配列データブロックへの書き込み」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =「1」: 配列 DB への書き込みを開始
DB	Input	DB_ANY	I、Q、M、D、L	データが書き込まれるデータブロック
INDEX	Input	DINT	I、Q、M、D、L、P、または定数	データが書き込まれる DB 内のエレメント
VALUE	Input	VARIANT	I、Q、M、L	書き込まれる値 TEMP セクションのローカル定数またはタグは、使用する必要がありません。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY =「1」: 配列 DB はまだ書き込み中です
DONE	Output	BOOL	I、Q、M、D、L	DONE =「1」: 命令が正常に実行されました
ERROR	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、ERROR パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERROR パラメータ

次の表に、ERROR パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
80B4	配列データブロックに保存されたエレメントのデータタイプが、VARIANT で転送されるエレメントのデータタイプと一致しません。
8230	データブロックが番号が不正です。
8231	データブロックが存在しません。
8232	データブロックが小さすぎるか、またはロードメモリ内に存在しません。
8234	データブロックが書き込み保護されています。
8235	データブロックが配列 DB ではありません。
8254	データブロックのデータタイプが不正です。
8382	INDEX パラメータの値が、配列の限界値外です。
8750	VALUE パラメータのデータタイプ VARIANT が、値「0」を提供します。
8751	コード生成エラー
8752	コード生成エラー
8753	VALUE パラメータのサイズが、配列データブロックのエレメントの長さと一致しません。

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。

命令「READ_DBL:ロードメモリのデータブロックからの読み出し」および「WRIT_DBL:データブロックへのロードメモリの書き込み」によってトリガされるエラーコードの説明については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL WriteToArrayDBL DB_ANY, "WriteToArrayDBL_DB"
     (REQ := "TagReg"
      DB := "ArrayDB"
      INDEX := "ArrayDB".THIS[2]
      VALUE := "SourceTargetField[1]".Data1[6]
      BUSY => "TagBusy"
      DONE => "TagDone"
      ERROR => "TagError"
      )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
REQ	TagReq	BOOL
DB	ArrayDB	「ArrayDB」オペランドのデータタイプは、INT の配列[0 to 10]です。
INDEX	"ArrayDB".THIS[2]	「ArrayDB」の 3 番目のエレメント
VALUE	"SourceTargetField[1]".Data1[6]	「SourceTargetField」オペランドのデータタイプは、「MOVE_UDT」の配列[0 to 10]です。
BUSY	TagBusy	BOOL
DONE	TagDone	BOOL

「TagReq」オペランドで立ち上がりエッジが検出されると、「ロードメモリの ARRAY データブロックへの書き込み」命令が実行されます。 「TagBusy」オペランドで信号立ち下がりエッジが検出された場合、すぐに命令は終了し、VALUE パラメータの値が「ArrayDB」の 3 番目のエレメントに書き込まれます。 この命令が処理された後、「TagDone」オペランドのシグナル状態は「1」になります。

レガシー

この章には下記に関する情報が記載されています：

- BLKMOV: ブロックの移動 (S7-1500)
- UBLKMOV: 割り込みなしブロック転送 (S7-1500)
- FILL: フィル命令 (S7-1500)

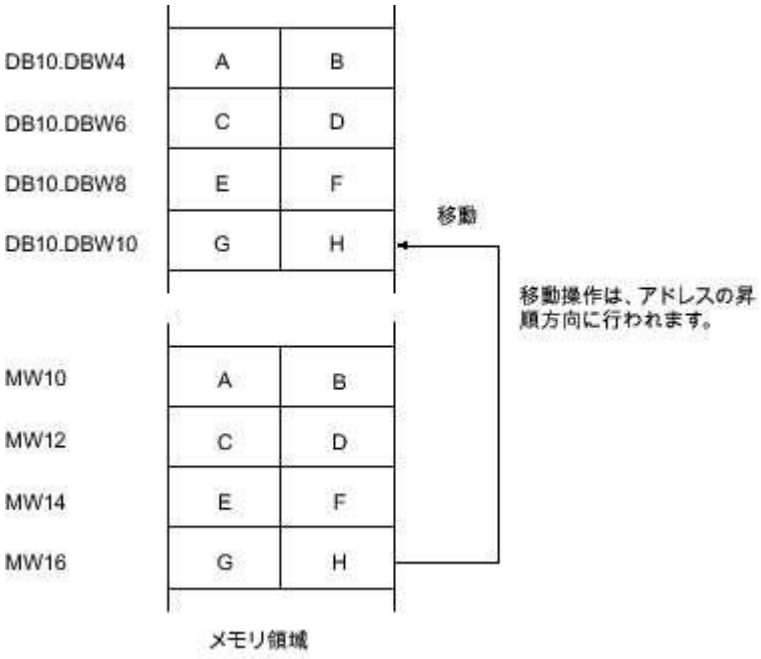
BLKMOV: ブロックの移動

説明

「ブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動します。ムーブ操作は、アドレスの昇順方向に行われます。VARIANT を使用して、ソース領域と宛先領域を定義します。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

次の図に、ムーブ操作の原理を示します。



ソース領域および宛先領域の整合性

「ブロックムーブ」命令の実行中にソースデータが変更されないことを確認してください。変更があった場合、宛先データの整合性は保証できません。

割り込み機能

ネストレベルに対する制限はありません。

メモリ領域

「ブロックの移動」命令を使用して、次のメモリ領域を移動します。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力

- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。ソース領域と宛先領域の長さが異なる場合、小さな領域の長さのみが移動します。

ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

文字列の移動のルール

STRING データタイプの移動元および移動先の領域の移動にも「ブロックの移動」命令を使用します。ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報も、宛先領域に書き込まれます。 ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。

文字列の最大長および実際の長さに関する情報を移動するには、SRCBLK および DSTBLK パラメータにバイト単位で領域を指定します。

構文

以下の構文が「ブロックの移動」命令に使用されます。

```
GRAPH
CALL BLKMOV VARIANT
     (SRCBLK := <operand>
      RET_VAL => <Operand>
      DSTBLK => <Operand>
      )
```

パラメータ

次の表に、「ブロックの移動」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output	VARIANT [1)]	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。

1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみで使用できます。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL BLKMOV VARIANT
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。 ムーブ操作中にエラーが発生した場合、「Tag_ErrorCode」タグにエラーコードが出力されます。

UBLKMOV: 割り込みなしブロック転送

説明

「割り込み不可能なブロックの移動」命令を使用して、メモリ領域(ソース領域)の内容を別のメモリ領域(宛先領域)に移動します。ムーブ操作は、アドレスの昇順方向に行われます。VARIANT を使用して、ソース領域と宛先領域を定義します。

他のオペレーティングシステムのアクティビティによって、ムーブ操作に割り込みをかけることはできません。結果として、「割り込みなしブロック転送」命令の実行中には CPU の割り込み応答時間が長くなります。

注記
この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

メモリ領域

「割り込み不可能なブロックムーブ」命令を使用して、次のメモリ領域を移動します。

- データブロックの領域
- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

「割り込みなしブロック転送」命令の実行中は、ソース領域および宛先領域が重複してはなりません。ソース領域が宛先領域よりも小さい場合、ソース領域全体が宛先領域に書き込まれます。宛先領域の残りのバイトは変更されません。

宛先領域がソース領域よりも小さい場合、宛先領域全体が書き込まれます。 ソース領域の残りのバイトは無視されます。

仮パラメータとして定義されたソース領域または宛先領域が、SRCBLK または DSTBLK パラメータで指定された宛先領域またはソース領域よりも小さい場合、データは転送されません。

BOOL データタイプのブロックを移動する場合、タグを絶対アドレス指定する必要があり、かつ領域の指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行できません。

「割り込み不可能なブロックムーブ」命令を使用して最大 16 KB を移動することができます。 この場合、CPU 固有の制限に注意してください。

文字列の移動のルール

STRING データタイプの移動元および移動先の領域の移動にも「割り込み不可能なブロックムーブ」命令を使用します。ソース領域が STRING データタイプのみの場合、文字列に実際に含まれる文字が移動されます。 実際の長さおよび最大長に関する情報は、宛先領域には書き込まれません。ソース領域と宛先領域の両方が STRING データタイプの場合、宛先領域内の文字列の現在の長さが、実際に移動する文字数にセットされます。 STRING データタイプのブロックを移動する場合、ブロックの長さとして「1」を指定します。

構文

以下の構文が「割り込みなしブロック転送」命令に使用されます。

```
GRAPH
CALL UBLKMOV VARIANT
     (SRCBLK := <operand>
      RET_VAL => <Operand>
      DSTBLK => <Operand>
      )
```

パラメータ

次の表に、「割り込みなしブロック転送」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SRCBLK	Input	VARIANT	I、Q、M、L、P	移動元のメモリ領域(ソース領域)を指定します。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。
DSTBLK	Output 1)	VARIANT	I、Q、M、L、P	ブロックの移動先のメモリ領域(宛先領域)を指定します。
1) データがタグに流れるため、DSTBLK パラメータは Output として宣言されます。ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8091	ソース領域または宛先領域は、ロードメモリのみで使用できます。
8152	WSTRING、WCHAR および BOOL データタイプは SRCBLK パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは DSTBLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL UBLKMOV VARIANT
     (SRCBLK := P#M100.0 BYTE 10
      RET_VAL => "Tag_ErrorCode"
      DSTBLK => P#DB1.DBX0.0 BYTE 10
      )
```

この命令は、MB100 から開始して 10 バイトをコピーし、DB1 に書き込みます。 ムーブ操作中にエラーが発生した場合、「Tag_ErrorCode」タグにエラーコードが出力されます。

FILL: フィル命令

説明

「フィル」命令を使用して、メモリ領域(宛先領域)を別のメモリ領域(ソース領域)の内容によって埋めます。 「フィル」命令は、宛先領域がすべて書き込まれるまでソース領域の内容を宛先領域に移動します。 ムーブ操作は、アドレスの昇順方向に行われます。

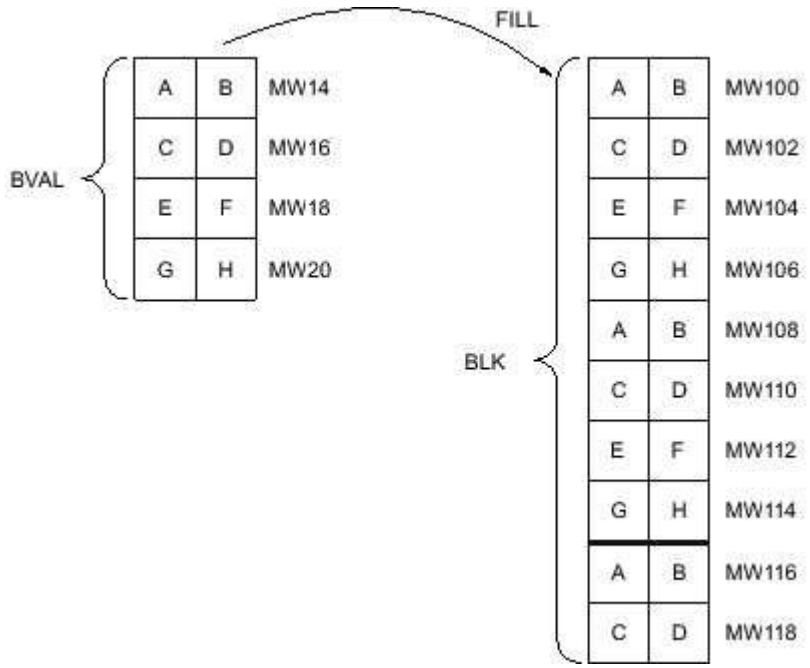
VARIANT を使用して、ソース領域と宛先領域を定義します。

注記

この命令のタグは、「最適化したブロックアクセス」属性が設定されていないデータブロックでのみ使用可能です。保持性設定「IDB に設定」が宣言されているタグの場合、「最適化されたアクセス」のタグも使用可能です。

「最適化したブロックアクセス属性を持つブロックの場合、命令「FILL_BLK: フィル」命令を使用できます。

次の図に、ムーブ操作の原理を示します。



例: MW100～MW118 の範囲の内容は、メモリワード MW14～MW20 の内容で事前割り当てされます。

ソース領域および宛先領域の整合性

「フィル」命令の実行中にソースデータが変更されないことを確認してください。変更されると、宛先データの整合性を保証できなくなります。

メモリ領域

「フィル」命令を使用して、以下のメモリ領域を移動します。

- データブロックの領域

- ビットメモリ
- プロセスイメージ入力
- プロセスイメージ出力

移動全般の規則

ソース領域と宛先領域は重複しないようにします。 事前割り当てする宛先領域が BVAL 入力パラメータの長さの整数倍でない場合でも、宛先領域は最後のバイトまで書き込まれます。

事前割り当てする宛先領域がソース領域よりも小さい場合は、ソース領域に含まれているデータのうち、宛先領域に書き込める量のみがコピーされます。

実際に存在する宛先領域またはソース領域がソース領域または宛先領域に割り当てられたメモリ領域よりも小さい場合(BVAL、BLK パラメータ)、データは転送されません。

ANY ポインタ(ソースまたは宛先)がデータタイプ BOOL の場合、これを絶対アドレス指定する必要があり、かつ指定された長さは 8 で割り切れる必要があります。割り切れないと、この命令は実行されません。

宛先領域が STRING データタイプの場合、この命令は管理情報を含む文字列全体を書き込みます。

構造体移動のルール

構造体を入力パラメータとして転送するとき、構造体の長さが常に偶数バイト数になっているように注意する必要があります。 構造体を奇数バイトによって宣言する場合は、追加の 1 バイトが必要になります。

構文

「フィル」命令には、以下の構文を使用します。

```
GRAPH
CALL FILL VARIANT
     (BVAL := <operand>
      RET_VAL => <Operand>
      BLK => <Operand>
      )
```

パラメータ

次の表に、「フィル」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
BVAL	Input	VARIANT	I、Q、M、L、P	その内容に内容が BLK パラメータの宛先領域を埋めるのに使用されるメモリ領域(ソース領域)の指定。
RET_VAL	Output	INT	I、Q、M、D、L、P	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

BLK	Output [1)]	VARIANT	I、Q、M、L、P	ソース領域の内容で埋められるメモリ領域の指定。
1) データがタグに流れるため、BLK パラメータは Output として宣言されます。ただし、タグ自体はブロックインターフェースで InOut として宣言される必要があります。				

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8092	ソース領域または宛先領域は、ロードメモリのみで使用できます。
8152	WSTRING、WCHAR および BOOL データタイプは BVAL パラメータでサポートされていません。
8352	WSTRING、WCHAR および BOOL データタイプは BLK パラメータでサポートされていません。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL FILL VARIANT
     (BVAL := P#M14.0 WORD 4
      RET_VAL => "Tag_ErrorCode"
      BLK => P#M100.0 WORD 10
      )
```

この命令は、ソース領域 MW14～MW20 をコピーし、BVAL パラメータのメモリ領域に含まれる 4 ワードの内容で宛先領域 MW100～MW118 を埋めます。

変換操作

この章には下記に関する情報が記載されています：

- CONVERT: 変換データ (S7-1500)
- ROUND: 数値の四捨五入 (S7-1500)
- CEIL: 浮動小数点数から次に大きい整数を生成 (S7-1500)
- FLOOR: 浮動小数点数から次に小さい整数を生成 (S7-1500)
- TRUNC: 値の切り捨て (S7-1500)
- SCALE_X: スケール (S7-1500)
- NORM_X: 正規化 (S7-1500)
- レガシー (S7-1500)

CONVERT: 変換データ

説明

「値の変換」命令を使用して、オペランドの内容を読み出し、割り当てられたデータタイプに従って変換します。

実行可能な変換に関する情報は、「関連項目」の「明示的な変換」のセクションを参照してください。

結果の値は、オーバーフローなどのエラーが処理中に発生した場合、無効です。

構文

「値の変換」命令には、以下の構文を使用します。

```
GRAPH
<Result> := ???_TO_???(<Operand>)
```

パラメータ

次の表に、「値の変換」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	2 進数、 整数、浮動小数点数、CHAR、WCHAR、BCD16、BCD32	I、Q、M、D、L、P、または定数	変換する値
<Result>	Output	2 進数、 整数、浮動小数点数、CHAR、WCHAR、BCD16、BCD32	I、Q、M、D、L、P	変換の結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := INT_TO_DINT("Tag_InValue")
```

「Tag_InValue」オペランドの内容が読み出され、整数(32 ビット)に変換されます。 結果が「Tag_Out-Value」オペランドに格納されます。

ROUND: 数値の四捨五入

説明

「数値の四捨五入」命令を使用して、オペランドの値を近似の整数に四捨五入します。 この命令は値を浮動小数点数として解釈し、浮動小数点数の値を隣の整数に変換します。 入力値が偶数と奇数のちょうど間にある場合、偶数が選択されます。

結果の値は、オーバーフローなどのエラーが処理中に発生した場合、無効です。

構文

「数値の四捨五入」命令には、以下の構文を使用します。

```
GRAPH
<Result> := ROUND(<Operand>)
```

パラメータ

次の表に、「数値の四捨五入」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	四捨五入される入力値
<Result>	Output	整数、浮動小数点数	I、Q、M、D、L、P	丸めの結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := ROUND("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	2	-2

「Tag_InValue」オペランドの浮動小数点数は、最も近い偶数の整数に四捨五入され、「Tag_OutValue」オペランドに出力されます。

CEIL: 浮動小数点数から次に大きい整数を生成

説明

「浮動小数点数から次に大きい整数を生成」命令を使用し、オペランドの値を次に大きい整数に四捨五入します。 この命令は、値を浮動小数点数として解釈し、この数を次に大きい整数に変換します。 結果は、入力値以上になることがあります。

結果の値は、オーバーフローなどのエラーが処理中に発生した場合、無効です。

構文

「浮動小数点数から次に大きい整数を生成」命令には、次の構文を使用します。

```
GRAPH
<Result> := CEIL(<Operand>)
```

パラメータ

次の表に、「浮動小数点数から次に大きい整数を生成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	浮動小数点数としての入力値
<Result>	Output	整数、浮動小数点数	I、Q、M、D、L、P	次に大きい整数の場合の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := CEIL("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	1	0

「Tag_InValue」オペランドの浮動小数点数は、次に大きい整数に四捨五入され、「Tag_OutValue」オペランドに出力されます。

FLOOR: 浮動小数点数から次に小さい整数を生成

説明

「浮動小数点数から次に小さい整数を生成」命令を使用し、オペランドの値を次に小さい整数に変換します。 この命令は、値を浮動小数点数として解釈し、この数を次に小さい整数に変換します。 結果は、入力値以下になることがあります。

結果の値は、オーバーフローなどのエラーが処理中に発生した場合、無効です。

構文

「浮動小数点数から次に小さい整数を生成」命令には、次の構文を使用します。

```
GRAPH
<Result> := FLOOR(<Operand>)
```

パラメータ

次の表に、「浮動小数点数から次に小さい整数を生成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、P、または定数	浮動小数点数としての入力値
<Result>	Output	整数、浮動小数点数	I、Q、M、D、L、P	次に小さい整数の場合の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := FLOOR("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_InValue	0.50000000	-0.50000000
Tag_OutValue	0	-1

「Tag_InValue」オペランドの浮動小数点数は、次に小さい整数に四捨五入され、「Tag_OutValue」オペランドに出力されます。

TRUNC: 値の切り捨て

説明

「数値を切り捨てる」命令を使用して、オペランドの値から整数を形成することができます。 値は、浮動小数点数として解釈されます。 この命令は、浮動小数点数の整数部のみを選択し、それを結果として小数位なしで出力します。

結果の値は、オーバーフローなどのエラーが処理中に発生した場合、無効です。

構文

「数値を切り捨てる」命令には、以下の構文を使用します。

```
GRAPH
<Result> := TRUNC(<Operand>)
```

パラメータ

次の表に、「数値を切り捨てる」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	Input	浮動小数点数	I、Q、M、D、L、または定数	浮動小数点数としての入力値
<Result>	Output	整数、浮動小数点数	I、Q、M、D、L	浮動小数点数の整数部分の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := TRUNC("Tag_InValue")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値	
Tag_InValue	1.50000000	-1.50000000
Tag_OutValue	1	-1

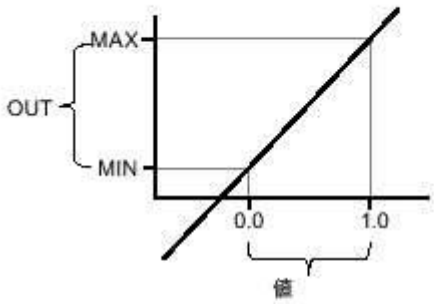
「Tag_InValue」オペランドの浮動小数点数の整数部が、整数に変換され、「Tag_OutValue」オペランドに出力されます。

SCALE_X: スケール

説明

「スケール」命令を使用して、VALUE パラメータの値を指定した値の範囲にマッピングしてスケーリングします。 命令が実行されると、VALUE 入力の浮動小数点値が、MIN および MAX パラメータによって定義された値の範囲にスケーリングされます。 スケーリングの結果は整数になり、OUT 出力に保存されます。

次の図に、値をどのようにスケーリングできるかを示します。



「スケール」命令は、以下の等式で機能します。

OUT = [VALUE * (MAX – MIN)] + MIN

次のいずれかの条件が満たされる場合、結果の値は無効です。

- MIN パラメータの値が、MAX パラメータの値以上であること。
- 指定した浮動小数点数の値が、IEEE-754 に従い正規化された数の範囲外であること。
- オーバーフローが発生していること。
- VALUE パラメータの値が NaN (Not a Number = 無効な算術演算の結果)であること。

注記
アナログ値の変換についての詳細については、それぞれのマニュアルを参照してください。

構文

「スケール」命令には、以下の構文を使用します。

```
GRAPH
CALL SCALE_X ???_???
      (MIN := <operand>
       VALUE := <operand>
       MAX := <operand>
       RET_VAL => <Operand>
       )
```

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MIN	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE	Input	浮動小数点数	I、Q、M、D、L、または定数	スケーリングされる値 定数を入力する場合は、それを宣言する必要があります。
MAX	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の上限値
RET_VAL	Output	整数、浮動小数点数	I、Q、M、D、L	スケーリングの結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL SCALE_X INT_REAL
      (MIN := "Tag_Minimum"
       VALUE := "Tag_Value"
       MAX := "Tag_Maximum"
       RET_VAL => "Tag_OutputValue"
       )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_Minimum	10
VALUE	Tag_Value	0.5
MAX	Tag_Maximum	30
RET_VAL	Tag_ReturnValue	20

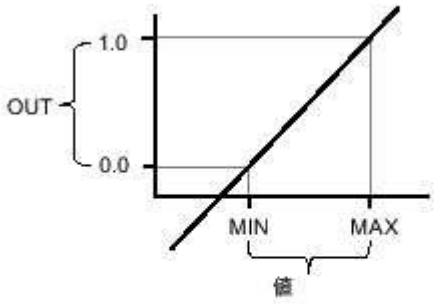
「Tag_Value」パラメータの値が、「Tag_Minimum」および「Tag_Maximum」パラメータの値によって定義された値の範囲にスケーリングされます。 結果が「Tag_ReturnValue」パラメータに保存されます。

NORM_X: 正規化

説明

「正規化」命令を使用して、VALUE パラメータの値をリニアスケーリングにマッピングして正規化します。 MIN および MAX パラメータを使用し、スケーリングに適用する値の範囲の限界を定義します。 正規化される値のこの値の範囲内の位置に応じて結果が計算され、OUT パラメータに浮動小数点数として保存されます。 正規化される値が MIN パラメータの値に等しい場合、OUT パラメータの値は「0.0」になります。 正規化される値が MAX パラメータの値に等しい場合、OUT パラメータの値は「1.0」になります。

次の図に、値が正規化される例を示します。



「正規化」命令は、以下の等式で機能します。

OUT = (VALUE – MIN) / (MAX – MIN)

次のいずれかの条件が満たされる場合、結果の値は無効です。

- MIN パラメータの値が、MAX パラメータの値以上であること。
- 指定した浮動小数点数の値が、IEEE-754 に従い正規化された数の範囲外であること。
- オーバーフローが発生していること。
- VALUE パラメータの値が NaN (Not a Number = 無効な算術演算の結果)であること。

注記
アナログ値の変換についての詳細については、それぞれのマニュアルを参照してください。

構文

「正規化」命令には、以下の構文を使用します。

```
GRAPH
CALL NORM_X ???_???
      (MIN := <operand>
       VALUE := <operand>
       MAX := <operand>
       RET_VAL => <Operand>
       )
```

パラメータ

次の表に、「正規化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MIN [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の下限値
VALUE [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	正規化する値
MAX [1)]	Input	整数、浮動小数点数	I、Q、M、D、L、または定数	値の範囲の上限値
RET_VAL	Output	浮動小数点数	I、Q、M、D、L	正規化の結果
[1)] これらの 3 つのパラメータで定数を使用する場合は、そのいずれかを宣言するのみで済みます。				

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

定数の宣言の詳細については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL NORM_X INT_REAL
      (MIN := "Tag_Minimum"
       VALUE := "Tag_Value"
       MAX := "Tag_Maximum"
       RET_VAL => "Tag_OutputValue"
       )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
MIN	Tag_Minimum	10
VALUE	Tag_Value	20
MAX	Tag_Maximum	30

RET_VAL	Tag_ReturnValue	0.5

「Tag_Value」パラメータの値が、「Tag_Minimum」および「Tag_Maximum」パラメータを使用して定義された値の範囲に割り当てられます。 「Tag_Value」パラメータのタグ値が、定義された値の範囲に従って正規化されます。 結果が、浮動小数点数として「Tag_ReturnValue」パラメータに保存されます。

レガシー

この章には下記に関する情報が記載されています：

- SCALE: スケール (S7-1500)
- UNSCALE: スケール解除 (S7-1500)

SCALE: スケール

説明

[スケール]命令を使用して、IN パラメータの整数を下限値と上限値の間の物理単位でスケーリングできる浮動小数点数に変換します。LO_LIM および HI_LIM パラメータを使用し、入力値をスケーリングする範囲の下限値と上限値を指定します。 命令の結果は、OUT パラメータに出力されます。

「スケール」命令は、以下の等式で機能します。

OUT = [((FLOAT (IN) – K1)/(K2–K1)) * (HI_LIM–LO_LIM)] + LO_LIM

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。 BIPOLAR パラメータには、以下のシグナル状態が可能です。

- シグナル状態「1」: IN パラメータの値はバイポーラで-27 648 ~ 27 648 の値の範囲にあると仮定されています。この場合、定数「K1」の値は「-27 648.0」に、定数「K2」の値は「+27 648.0」になります。
- シグナル状態「0」: IN パラメータの値はユニポーラで 0 ~ 27 648 の値の範囲にあると仮定されています。この場合、定数「K1」の値は「0.0」に、定数「K2」の値は「+27 648.0」になります。

IN パラメータの値が定数値「K2」よりも大きい場合、命令の結果が上限値(HI_LIM)にセットされ、エラーが出力されます。

IN パラメータの値が定数値「K1」未満の場合、命令の結果が下限値(LO_LIM)の値にセットされ、エラーが出力されます。

指定された下限値が上限値よりも大きい(LO_LIM >HI_LIM)場合、結果は入力値に反比例してスケーリングされます。

構文

「スケール」命令には、以下の構文を使用します。

```
GRAPH
CALL SCALE
     (IN := <Operand>
      HI_LIM := <Operand>
      LO_LIM := <Operand>
       BIPOLAR := <Operand>
      RET_VAL => <Operand>
      OUT => <Operand>
      )
```

パラメータ

次の表に、「スケール」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	INT	I、Q、M、D、L、P、または定数	スケーリングする入力値。

HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
RET_VAL	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT	Output	REAL	I、Q、M、D、L、P	命令の結果

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	**説明**
0000	エラーは発生していません。
0008	IN パラメータの値が定数値「K2」を超えているか、または定数値「K1」未満になっています。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
CALL SCALE
     (IN := "Tag_InputValue"
      HI_LIM := "Tag_HighLimit"
      LO_LIM := "Tag_LowLimit"
      BIPOLAR := "Tag_Bipolar"
      RET_VAL => "Tag_ErrorCode"
      OUT => "Tag_OutputValue"
      )
```

次の表に、実行前の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

次の表に、実行後の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	22
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	50.03978588

「Tag_InputValue」オペランドは、変換されてスケーリングされる値を示します。 上限値と下限値が、「Tag_HighLimit」および「Tag_LowLimit」オペランドによって定義されます。 オペランド「Tag_Bipolar」 = TRUE を使用して、IN パラメータの値がバイポーラとして解釈されるように指定します。 命令の結果は、「Tag_OutputValue」オペランドに出力されます。

UNSCALE: スケール解除

説明

「スケール解除」命令を使用して、IN パラメータの浮動小数点数を下限値と上限値間の物理単位にスケーリング解除し、それらを整数に変換します。LO_LIM および HI_LIM パラメータを使用し、入力値をスケーリング解除する範囲の下限値と上限値を指定します。 命令の結果は、OUT パラメータに出力されます。

「スケール解除」命令は、以下の等式で機能します。

OUT = [((IN–LO_LIM)/(HI_LIM–LO_LIM)) * (K2–K1)] + K1

定数「K1」および「K2」の値は、BIPOLAR パラメータのシグナル状態で決定されます。BIPOLAR パラメータでは、以下のシグナル状態が可能です。

- シグナル状態「1」:IN パラメータの値はバイポーラで-27 648～27 648 の値の範囲にあると仮定されています。この場合に定数「K1」の値は「-27 648.0」に定数「K2」の値は「+27 648.0」になります。
- シグナル状態「0」:IN パラメータの値はユニポーラで、0～27 648 の範囲の値にあると仮定されています。この場合に定数「K1」の値は「0.0」、定数「K2」の値は「+27 648.0」になります。

IN パラメータの値が定数値「HI_LIM」よりも大きい場合、この命令の結果が定数値(K2)にセットされ、エラーが出力されます。

IN パラメータの値が下限値の定数値「LO_LIM」未満の場合、この命令の結果が定数値(K1)にセットされ、エラーが出力されます。

構文

以下の構文が「スケール解除」命令に使用されます。

```
GRAPH
CALL UNSCALE
      (IN := <Operand>
       HI_LIM := <Operand>
       LO_LIM := <Operand>
        BIPOLAR := <Operand>
       RET_VAL => <Operand>
       OUT => <Operand>
       )
```

パラメータ

次の表に、「スケール解除」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	REAL	I、Q、M、D、L、P、または定数	整数値にスケーリング解除する入力値。

HI_LIM	Input	REAL	I、Q、M、D、L、P、または定数	上限値
LO_LIM	Input	REAL	I、Q、M、D、L、P、または定数	下限値
BIPOLAR	Input	BOOL	I、Q、M、D、L、または定数	IN パラメータの値をバイポーラとして解釈するか、またはユニポーラとして解釈するかを示します。 以下のパラメータ値が想定されます。 1: バイポーラ 0: ユニポーラ
RET_VAL	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT	Output	INT	I、Q、M、D、L、P	命令の結果

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
0008	IN パラメータの値が上限値(HI_LIM)を超えているか、または下限値(LO_LIM)未満になっています。
一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
CALL UNSCALE
     (IN := "Tag_InputValue"
      HI_LIM := "Tag_HighLimit"
      LO_LIM := "Tag_LowLimit"
      BIPOLAR := "Tag_Bipolar"
      RET_VAL => "Tag_ErrorCode"
      OUT => "Tag_OutputValue"
      )
```

次の表に、実行前の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	0.0

次の表に、実行後の各種オペランドの値を示します。

パラメータ	オペランド	値
IN	Tag_InputValue	50.03978588
HI_LIM	Tag_HighLimit	100.0
LO_LIM	Tag_LowLimit	0.0
BIPOLAR	Tag_Bipolar	1
RET_VAL	Tag_ErrorCode	W#16#0000
OUT	Tag_OutputValue	22

「Tag_InputValue」オペランドは、変換されてスケーリング解除される値を示します。 上限値と下限値が、「Tag_HighLimit」および「Tag_LowLimit」オペランドによって定義されます。 オペランド「Tag_Bipolar」 = TRUE を使用して、IN パラメータの値がバイポーラとして解釈されるように指定します。 命令の結果は、「Tag_OutputValue」オペランドに出力されます。

プログラム制御演算

この章には下記に関する情報が記載されています：

- ランタイム制御 (S7-1500)

ランタイム制御

この章には下記に関する情報が記載されています：

- ENDIS_PW: パスワードの適正化の制限および有効化 (S7-1500)
- RE_TRIGR: サイクルタイムモニタのリスタート (S7-1500)
- STP: プログラム終了 (S7-1500)
- GET_ERROR: ローカルでエラーを取得 (S7-1500)
- GET_ERR_ID: ローカルでエラー ID を取得 (S7-1500)
- INIT_RD: すべての保持データの初期化 (S7-1500)
- WAIT: 遅延時間の設定 (S7-1500)

ENDIS_PW: パスワードの適正化の制限および有効化

説明

「パスワード正当性の制限および有効化」命令を使用して、設定されたパスワードを CPU に対して正当化できるかどうかを指定します。 このため、正しいパスワードがわかっている場合でも、正規の接続を防止することができます。

命令呼び出し時に REQ パラメータの信号状態が「0」の場合、現在設定されている状態が、出力パラメータに表示されます。 入力パラメータに変更を加えても、変更は出力パラメータに転送されません。

命令呼び出し時に REQ パラメータの信号状態が「1」の場合、信号状態は、入力パラメータ(F_PWD、FULL_PWD、R_PWD、HMI_PWD)から取得されます。FALSE は、パスワードごとの正当化が許可されていないことを意味します。TRUE は、パスワードを使用できることを意味します。

パスワードの有効化または無効化を個別に許可または禁止することができます。 たとえば、フェールセーフパスワードを除く、すべてのパスワードを禁止することができます。 したがって、アクセスオプションを小さいユーザーグループに制限することができます。 出力パラメータ(F_PWD_ON、FULL_PWD_ON、R_PWD_ON、HMI_PWD_ON)は、REQ パラメータに関係なく、常にパスワード使用の現在のステータスを示します。

設定済みでないパスワードの場合、入力の信号状態が TRUE でなければならず、出力に信号状態 TRUE を返します。 フェールセーフパスワードは、F-CPU の場合のみ設定できるため、標準 CPU では、常に信号状態 TRUE と相互接続する必要があります。 命令がエラーを返す場合、この呼び出しには効果がなく、直前のロックがまだ有効です。

無効なパスワードは、以下の条件で再度有効にすることができます。

- CPU が工場出荷時設定にリセットされること。
- S7-1500 CPU のフロントパネルが、パスワードを再度有効にすることができる適切なメニューに移動することができるダイアログをサポートしていること。
- 「パスワードの適正化の制限および有効化」命令を呼び出す場合、目的のパスワードの入力パラメータの信号状態が「1」であること。
- モードセレクタを STOP に設定すること。 このスイッチの設定を RUN に戻すと直ちに、パスワードの適正化に対する制限が再確立されます。
- S7-1200 CPU への空きメモリカード(転送モジュールまたはプログラムカード)の差し込み。
- S7-1200 CPU では、電源オフから電源オンへの移行によって、保護が無効になります。 この場合、プログラム(たとえば、スタートアップ OB)で、再度、「パスワード正当性の制限および有効化」命令を呼び出す必要があります。

注記
「パスワード正当性の制限および有効化」命令は、HMI パスワードが有効にされていない場合、HMI システムからのアクセスをブロックします。

注記
既存の適正化された接続はアクセス権を保持し、「パスワードの適正化の制限および有効化」命令を使用してそれらを制限することはできません。

S7-1500 CPU での意図しないロックアウトの防止

CPU のフロントパネルで設定を行うことができ、CPU は最新の設定を保存します。

偶発的なロックアウトを防止するには、S7-1500 CPU でモードセレクタを STOP に設定することによって保護を無効にすることができます。 モードセレクタを RUN に設定することによって、「パスワ

ードの適正化の制限および有効化」命令を再度呼び出したり、フロントパネルで追加の操作を行う必要なしに、保護が自動的に再度有効にされます。

S7-1200 CPU での意図しないロックアウトの防止

S7-1200 CPU にはモードスイッチが存在しないため、電源オフから電源オンへの移行時の保護は無効です。 このため、ユーザープログラム内の特定のプログラムシーケンスによって偶発的なロックアウトを防止することをお奨めします。

これを行うには、サイクリック割り込み OB、または、メイン OB (OB 1)のタイマのどちらかを使用して、時間制御をプログラミングします。 これによって、電源オフから電源オンへの移行と、関連する保護の無効化の後、比較的迅速に、各 OB (たとえば、OB 1 または OB 35)内で「パスワード正当性の制限および有効化」命令を呼び出すオプションが可能になります。 スタートアップ OB (OB 100)でこの命令を呼び出して、この命令が無効で、パスワード正当化の制限が存在しない時間ウィンドウを比較的小さなものに保ちます。 この手順は、未許可のアクセスに対する可能な最大限の保護を提供します。

偶発的なロックアウトが発生した場合は、スタートアップ OB での呼び出しをスキップし(たとえば、入力パラメータを照会することによって)、ロックが再び有効になる前に、CPU へ接続確立する時間(たとえば、10 秒 ~ 1 分)を設定することができます。

ユーザープログラムコードにタイマが存在しない場合にロックアウトが発生した場合は、空の転送カードまたはプログラムカードを CPU に挿入します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 この後、ユーザープログラムを再び STEP 7 から CPU にダウンロードする必要があります。

S7-1200 CPU でパスワードが失われた場合の手順

パスワード保護された S7-1200 CPU のパスワードを紛失した場合は、空の転送カードまたはプログラムカードを使用して、パスワード保護されたプログラムを削除します。 空の転送カードまたはプログラムカードは、CPU の内部ロードメモリを削除します。 その後、新しいユーザープログラムを STEP 7 Basic から CPU にロードすることができます。

警告

空の転送カードの挿入

ランタイム中に CPU にトランスファーカードを挿入すると、CPU は STOP モードに切り替わります。 動作ステータスが不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。 これは、オートメーションシステムの予期せぬ動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

転送カードを取り出すと、転送カードの内容が内部ロードメモリで使用可能になります。 この時点に、カードにプログラムが含まれていないことを確認してください。

警告

空のプログラムカードの挿入

ランタイム中に CPU にプログラムカードを挿入すると、CPU は STOP モードに移行します。動作ステータスが不安定な場合、コントローラで障害が発生し、制御対象デバイスの制御されない動作の原因になる場合があります。これは、オートメーションシステムの予期せぬ動作につながり、致命的または重大な人的傷害や物的損害の原因になる恐れがあります。

プログラムカードが空であることを確認してください。 内部ロードメモリが空のプログラムカードにコピーされます。 直前に空であったプログラムカードを取り出すと、内部ロードメモリは空になります。

CPU を RUN モードに切り替える前に、転送カードまたはプログラムカードを取り出す必要があります。

動作モードへのパスワードの使用の影響

次の表に、「パスワードの適正化の制限および有効化」命令によるパスワードの使用が動作モードと対応するユーザーの操作に与える影響を示します。

操作	命令によるパスワードの保護
その後の基本状態 • STOP への動作モードの切り替え • メモリの手動リセット(PG、切り替え、MC (モーションコントロール)の変更) • 工場出荷時設定へのリセット	無効 (制限なし)
電源オン後の基本状態	• S7-1200-CPU: ロックは無効で、プログラム(たとえば、スタートアップ OB)で再びこの命令を呼び出す必要があります。 • S7-1500 CPU: 有効(電源オフの前にロックが有効になった場合). パスワードの不許可のオプションは保持型です。
動作モードの移行 RUN/STARTUP/HOLD -> STOP (命令、エラーまたは通信の終了によって)または STOP -> STARTUP/RUN/HOLD	有効 パスワードはまだ使用できません。

構文

「パスワード正当性の制限および有効化命令には、以下の構文を使用します。

```
GRAPH
CALL ENDIS_PW
     (REQ := <Operand>
      F_PWD := <Operand>
      FULL_PWD := <Operand>
      R_PWD := <Operand>
      HMI_PWD := <Operand>
      F_PWD_ON => <Operand>
      FULL_PWD_ON => <Operand>
      R_PWD_ON => <Operand>
      HMI_PWD_ON => <Operand>
      RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「パスワードの適正化の制限および有効化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ パラメータの信号状態が「0」の場合は、現在設定されているパスワードの信号状態が照会されます。
F_PWD	Input	BOOL	I、Q、M、D、L、または定数	フェールセーフなどの読み取り/書き込みアクセス • F_PWD = "0": パスワードを許可しない • F_PWD = "1": パスワードを許可する
FULL_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取り/書き込みアクセス • FULL_PWD = "0": パスワードを許可しない • FULL_PWD = "1": パスワードを許可する
R_PWD	Input	BOOL	I、Q、M、D、L、または定数	読み取りアクセス • R_PWD = "0": パスワードを許可しない • R_PWD = "1": パスワードを許可する
HMI_PWD	Input	BOOL	I、Q、M、D、L、または定数	HMI アクセス • HMI_PWD = "0": パスワードを許可しない • HMI_PWD = "1": パスワードを許可する
F_PWD_ON	Output	BOOL	I、Q、M、D、L	フェールセーフなどの読み取り/書き込みアクセス • F_PWD_ON = "0": パスワードを許可しない • F_PWD_ON = "1": パスワードを許可する
FULL_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取り/書き込みアクセスステータス • FULL_PWD_ON = "0": パスワードを許可しない • FULL_PWD_ON = "1": パスワードを許可する
R_PWD_ON	Output	BOOL	I、Q、M、D、L	読み取りアクセスステータス • R_PWD_ON = "0": パスワードを許可しない • R_PWD_ON = "1": パスワードを許可する
HMI_PWD_ON	Output	BOOL	I、Q、M、D、L	HMI アクセスステータス • HMI_PWD_ON = "0": パスワードを許可しない • HMI_PWD_ON = "1": パスワードを許可する
RET_VAL	Output	WORD	I、Q、M、D、L	エラー情報

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

8090	「パスワードの適正化の制限および有効化」命令はサポートされていません
80D0	フェールセーフのパスワードが未設定です。 標準 CPU では、信号状態は TRUE であることが必要です。
80D1	読み取り/書き込みアクセスが未設定です
80D2	読み取りアクセスが未設定です
80D3	HMI アクセスが未設定です
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

RE_TRIGR: サイクルタイムモニタのリスタート

説明

「サイクルタイムモニタのリスタート」命令を使用して、CPU の最大サイクルタイムを再起動することができます。 最大サイクルタイムが、CPU の設定で設定した期間中に最初からやり直されます。

「サイクルタイムモニタのリスタート」命令は、すべてのブロック内で優先度に関わらず呼び出し可能です。

ハードウェア割り込み、診断割り込み、または周期割り込みなどの上位の優先度のブロック内でこの命令が呼び出される場合、この命令は実行されません。

「サイクルタイムモニタのリスタート」命令は、呼び出しの数に関係なく期間内に完全に実行されます (最大プログラムサイクルの 10 倍)。 時間が期限切れになると、プログラムサイクルは延長できなくなります。

構文

「サイクルタイムモニタのリスタート」命令には、以下の構文を使用します。

```
GRAPH
CALL RE_TRIGR ()
```

パラメータ

「サイクルタイムモニタのリスタート」命令にはパラメータがなく、エラー情報は提供されません。

STP: プログラム終了

説明

「プログラムの終了」命令を使用し、CPU を STOP モードに設定してプログラムの実行を終了します。RUN から STOP への変更の影響は CPU の構成によって異なります。

構文

以下の構文が「プログラムの終了」命令に使用されます。

```
GRAPH
CALL STP ()
```

パラメータ

「プログラムの終了」命令にはパラメータがなく、エラー情報は提供されません。

GET_ERROR: ローカルでエラーを取得

説明

「ローカルでエラーを取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 システムがブロックの処理中にエラーを報告した場合、命令の最後の実行の後、このエラーがブロックの実行中に発生した最初の該当エラーの場合、詳細情報が、OUT 出力のオペランドに格納されます。

OUT 出力では、「ErrorStruct」システムデータタイプのオペランドのみを指定できます。 「ErrorStruct」システムデータタイプは、エラー情報が保存されている正確な構造体を指定します。 追加の命令を使用することで、この構造体とプログラムが適切な応答をするかを評価できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーに関するエラー情報が出力されます。

注記
OUT 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラーを取得」命令は、ブロック内のローカルエラー処理を可能にします。 「ローカルでエラーを取得」がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

構文

「ローカルでエラーを取得」命令には、以下の構文を使用します。

```
GRAPH
CALL GET_ERROR
     (OUT => <Operand>
      )
```

パラメータ

次の表に、「ローカルでエラーを取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OUT	Output	ErrorStruct	D、L	エラー情報

データタイプ「ErrorStruct」

次の表に、「ErrorStruct」データタイプの構造体を示します。

構造体コンポーネント		データタイプ	説明
ERROR_ID		WORD	エラー ID
FLAGS		BYTE	エラーがブロック呼び出し中に発生したのかどうかを示します。 16#01: ブロック呼び出し中のエラー 16#00: ブロック呼び出し中にエラーなし
REACTION		BYTE	デフォルトの応答 0: 無視(書き込みエラー) 1: 代替値「0」で続行(読み出しエラー) 2: 命令をスキップ(システムエラー)
CODE_ADDRESS		CREF	ブロックのアドレスおよびタイプに関する情報
	BLOCK_TYPE	BYTE	エラーが発生したブロックのタイプ: 1: OB 2: FC 3: FB
	CB_NUMBER	UINT	プログラムブロックの番号
	OFFSET	UDINT	内部メモリへの参照
MODE		BYTE	オペランドのアドレスに関する情報
OPERAND_NUMBER		UINT	マシンコマンドのオペランド番号
POINTER_NUMBER_LOCATION		UINT	(A) 内部ポインタ
SLOT_NUMBER_SCOPE		UINT	(B) 内部メモリの記憶領域
DATA_ADDRESS		NREF	オペランドのアドレスに関する情報
	AREA	BYTE	(C) メモリ領域 L: 16#40～4E、86、87、8E、8F、C0～CE I:16#81 Q: 16#82 M: 16#83 DB: 16#84、85、8A、8B データタイプ DINT の直接編集可能なタグの範囲違反: 16#04
	DB_NUMBER	UINT	(D) データブロックの番号

	OFFSET	UDINT	(E) オペランドの相対アドレス

構造体コンポーネント「ERROR_ID」

次の表に、構造体コンポーネント「ERROR_ID」で出力可能な値を示します。

ID* (16 進数)	ID* (10 進)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2534	9524	ブロック番号エラー FC
2535	9525	ブロック番号エラー FB
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、ファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、ファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー:入力
2943	10563	書き込みエラー:出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

GET_ERR_ID: ローカルでエラー ID を取得

説明

「ローカルでエラー ID を取得」命令は、ブロック内のエラーの発生のクエリに使用されます。 これは、通常、アクセスエラーによって生じます。 この命令の最後の実行の後に、システムがブロック処理中にブロック実行に関するエラーを報告すると、タグで発生した最初のエラーのエラー ID が RET_VAL 出力に格納されます。

RET_VAL 出力では、WORD データタイプのオペランドのみを指定できます。 ブロックで複数のエラーが発生した場合、発生した最初のエラーが修復された後にのみ、命令で発生した次のエラーのエラー ID が出力されます。

注記
RET_VAL 出力は、エラー情報が存在する場合のみ変更されます。 エラーを処理した後、出力を「0」にセットし直す場合、以下のオプションがあります。

- ブロックインターフェースの「一時」セクションでタグを宣言する。
- 命令を呼び出す前に、タグを「0」にリセットする。

「ローカルでエラー ID を取得」命令の出力は、エラー情報が存在する場合のみ設定されます。これらの条件の 1 つが満たされない場合、残りのプログラム実行は「ローカルでエラー ID を取得」命令の影響を受けません。

この命令を他のトラブルシューティングオプションと組み合わせて実装する方法の例は、「関連項目」を参照してください。

注記
「ローカルでエラー ID を取得」命令は、ブロック内のローカルエラー処理を可能にします。「ローカルでエラー ID を取得」命令がブロックのプログラムコードに挿入されている場合、エラー発生時に、あらかじめ定義されたすべてのシステム応答が無視されます。

構文

「ローカルでエラー ID を取得」命令には、以下の構文を使用します。

```
GRAPH
CALL GET_ERR_ID
     (RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「ローカルでエラー ID を取得」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Output	WORD	I、Q、M、D、L	エラー ID

RET_VAL パラメータ

次の表に、RET_VAL パラメータで出力できる値を示します。

エラーコード* (16 進数)	エラーコード* (10 進数)	説明
0	0	エラーは発生していません。
2503	9475	無効なポインタ
2520	9504	無効な STRING
2522	9506	読み取りエラー: オペランドが有効な範囲外
2523	9507	書き込みエラー: オペランドが有効な範囲外
2524	9508	読み取りエラー: 無効なオペランド
2525	9509	書き込みエラー: 無効なオペランド
2528	9512	読み取りエラー: データのアラインメント
2529	9513	書き込みエラー: データのアラインメント
252C	9516	無効なポインタ
2530	9520	書き込みエラー: データブロック
2533	9523	無効なポインタが使用されました
2534	9524	ブロック番号エラー FC
2535	9525	ブロック番号エラー FB
2538	9528	アクセスエラー: DB が存在しない
2539	9529	アクセスエラー: 不適切な DB が使用されました
253A	9530	グローバルデータブロックが存在しません。
253C	9532	情報が誤っているか、ファンクションが存在しません。
253D	9533	システムファンクションが存在しません。
253E	9534	情報が誤っているか、ファンクションブロックが存在しません。
253F	9535	システムブロックが存在しません。
2550	9552	アクセスエラー: DB が存在しない
2551	9553	アクセスエラー: 不適切な DB が使用されました
2575	9589	プログラムのネスティング深さのエラー
2576	9590	ローカルデータ配布のエラー
2577	9591	ブロックプロパティ「レジスタ経由のパラメータ渡し」が選択されていません。
25A0	9632	TP の内部エラー
25A1	9633	タグが書き込み保護されています
25A2	9634	タグの数値が不正です
2942	10562	読み取りエラー:入力
2943	10563	書き込みエラー:出力
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。		

INIT_RD: すべての保持データの初期化

説明

「すべての保持データを初期化」命令を使用して、すべてのデータブロック、ビットメモリ、および SIMATIC タイマとカウンタの保持データを同時にリセットします。 この命令は、実行するとプログラムサイクル持続時間を超過するため、スタートアップ OB 内でのみ実行が可能です。

構文

「すべての保持データを初期化」命令には、以下の構文を使用します。

```
GRAPH
CALL INIT_RD
     (REQ := <Operand>
      RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「すべての保持データを初期化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	入力「REQ」のシグナル状態が「1」の場合、すべての保持データがリセットされます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報: 命令の実行中にエラーが発生した場合、RET_VAL パラメータにエラーコードが出力されます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
80B5	この命令はスタートアップ OB 内にプログラムされていないため、実行できません。

一般エラー情報	関連項目: "GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラミング工ディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
CALL INIT_RD
     (REQ := "Tag_REQ"
      RET_VAL => "Tag_RET_VAL"
      )
```

「Tag_REQ」オペランドのシグナル状態が「1」の場合、この命令が実行されます。 すべてのデータブロック、ビットメモリ、および SIMATIC タイマの保持データとカウンタがリセットされます。

WAIT: 遅延時間の設定

説明

「遅延時間の設定」命令を使用して、指定された時間の間プログラム実行を一時停止できます。 WT パラメータにマイクロ秒単位で時間を指定します。

遅延時間は、-32768 から最大 32767 マイクロ秒 (µs)が設定可能です。設定可能な最短の遅延時間は個々の CPU に依存し、「遅延時間の設定」命令の実行時間に対応します。

この命令の実行には、さらに優先度の高いイベントが割り込む可能性があります。

「遅延時間の設定」命令はエラー情報を返しません。

構文

以下の構文が「遅延時間の設定」命令に使用されます。

```
GRAPH
CALL WAIT
      (WT := <operand>
      )
```

パラメータ

次の表に、「遅延時間の設定」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
WT	Input	INT	I、Q、M、D、L、P、または定数	遅延時間(µs)

ワード論理演算

この章には下記に関する情報が記載されています：

- NOT (INV):反転 (S7-1500)
- DECO: デコード (S7-1500)
- ENCO: エンコード (S7-1500)
- SEL: 選択 (S7-1500)
- MUX: 多重化 (S7-1500)
- DEMUX: 逆多重化 (S7-1500)

NOT (INV):反転

説明

命令「反転」を使用して、オペランドのビットのシグナル状態を反転します。この命令が実行されると、オペランドの値が排他的 OR で 16 進数マスク(16 ビットの数では W#16#FFFF、または 32 ビットの数では DW#16#FFFF FFFF)にリンクされます。 これによって、この後、結果として出力される個々のビットのシグナル状態が反転されます。

構文

「反転」命令には、以下の構文を使用します。

```
GRAPH
<Result> := NOT(<Operand>)
```

パラメータ

次の表に、「反転」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand>	入力	ビット列、整数	I、Q、M、D、L、P、または定数	入力値
<Result>	出力	ビット列、整数	I、Q、M、D、L、P	オペランド値の 1 の補数

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
"TagOut_Value" := NOT("TagIn_Value")
```

次の表に、命令が特定のオペランド値を使用してどのように機能するかを示します。

オペランド	値
TagIn_Value	W#16#000F
TagOut_Value	W#16#FFF0

この命令は、「TagIn_Value」オペランドの個々のビットのシグナル状態を反転し、その結果を「TagOut_Value」オペランドに書き込みます。

DECO: デコード

説明

「デコード」命令を使用して、入力値で指定されたビットを出力値にセットします。

「デコード」命令は、IN パラメータの値を読み取り、読み取った値とビット位置が一致する OUT パラメータにそのビットをセットします。 出力値の他のビットはゼロで埋められます。 IN パラメータの値が 31 よりも大きい場合、モジュロ 32 演算が実行されます。

構文

「デコード」命令には、以下の構文を使用します。

```
GRAPH
CALL DECO UINT_???
      (IN := <operand>
       OUT => <Operand>
      )
```

パラメータ

次の表に、「デコード」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	UINT	I、Q、M、D、L、P、または定数	設定される出力値のビット位置。
OUT	Output	ビット列	I、Q、M、D、L、P	出力値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

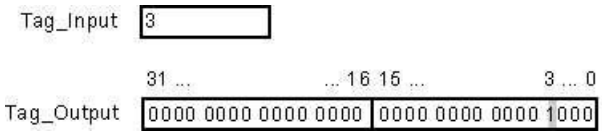
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL DECO UINT_DWORD
      (IN := "Tag_Input"
       OUT => "Tag_Output"
      )
```

次の図に、命令が特定の値を使用してどのように動作するかを示します。



この命令は、入力の「Tag_Input」オペランドの値からビット番号「3」を読み取り、3 番目のビットを出力の「Tag_Output」オペランドの値にセットします。

ENCO: エンコード

説明

「エンコード」命令を使用して、入力値の最下位ビットのビット番号を読み出し、OUT パラメータに出力します。

「エンコード」命令は IN パラメータの値の最下位ビットを選択し、そのビット番号を OUT パラメータのオペランドに書き込みます。 IN パラメータに値 DW#16#00000001 または DW#16#00000000 が含まれる場合、値「0」が OUT パラメータに出力されます。

構文

「エンコード」命令には、以下の構文を使用します。

```
GRAPH
CALL ENCO ???
      (IN := <operand>
       OUT => <Operand>
      )
```

パラメータ

次の表に、「エンコード」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	ビット列	I、Q、M、D、L、P、または定数	入力値
OUT	Output	INT	I、Q、M、D、L、P	出力値

命令のデータタイプを[???]ドロップダウンリストから選択できます。

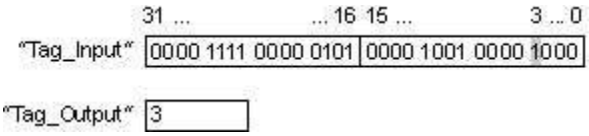
有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
CALL ENCO DWORD
      (IN := "Tag_Input"
       OUT => "Tag_Output"
      )
```

次の図に、命令が特定の値を使用してどのように動作するかを示します。



この命令は「Tag_Input」タグの最下位ビットを選択し、そのビット位置「3」を「Tag_Output」タグに書き込みます。

SEL: 選択

説明

「選択」命令を使用して、スイッチ(G 入力)に応じて、2 つの入力 IN0 または IN1 のいずれかを選択し、その内容を OUT 出力に移動します。G 入力のシグナル状態が「0」の場合、IN0 入力の値が移動されます。 G 入力のシグナル状態が「1」の場合、IN1 入力の値が OUT 出力に移動されます。

命令は、すべてのパラメータのタグが同じデータタイプの場合のみ実行されます。

構文

「選択」命令には、以下の構文を使用します。

```
GRAPH
CALL SEL ???
     (G  := <operand>
      IN0  := <operand>
      IN1  := <operand>
      OUT  => <Operand>
     )
```

パラメータ

次の表に、「選択」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
G	Input	BOOL	I、Q、M、D、L、または定数	スイッチ
IN0	Input	ビット列、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	最初の入力値
IN1	Input	ビット列、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P、または定数	2 番目の入力値
OUT	Output	ビット列、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、P	結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL SEL CHAR
     (G := "Tag_Input_G"
      IN0 := "Tag_Input0"
      IN1 := "Tag_Input1"
      OUT => "Tag_Output"
     )
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
G	Tag_Input_G	1
IN0	Tag_Input0	W#16#0000
IN1	Tag_Input1	W#16#FFFF
OUT	Tag_Output	W#16#FFFF

「Tag_Input_G」入力のシグナル状態に基づいて、「Tag_Input0」または「Tag_Input1」入力の値が選択され、「Tag_Output」出力に移動されます。

MUX: 多重化

説明

「多重化」命令を使用して、選択した入力の内容を RET_VAL 出力にコピーします。 選択可能な入力の数は最大 32 まで拡張できます。番号付けは IN0 で開始し、各新規入力によって続きます。K パラメータを使用し、内容が RET_VAL 出力にコピーされる入力を定義します。 K パラメータの値が使用可能な入力の数よりも大きい場合、INELSE パラメータの内容が RET_VAL 出力にコピーされます。

「多重化」命令は、すべての入力と RET_VAL 出力のタグが同じデータタイプの場合にのみ実行できます。 K パラメータは、指定できるのが整数のみであるため、例外です。

RET_VAL パラメータの値は、次の条件の 1 つが満たされると無効です。

- K パラメータの入力は、使用可能な入力の外側にあります。 この応答は、入力 INELSE が使用されるかどうかに関係ありません。 出力 RET_VAL の値は変更されないままになり、許可出力 ENO は「0」にセットされます。
- 命令の実行中にエラーが発生していること。

構文

「多重化」命令には、以下の構文を使用します。

```
GRAPH
CALL MUX ???_???
      (K  := <operand>
       IN0  := <operand>
       IN1  := <operand>
       INn  := <operand>
       INELSE  := <operand>
       RET_VAL => <Operand>
      )
```

パラメータ

次の表に、「多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
K	Input	整数	I、Q、M、D、L、または定数	内容がコピーされる入力を指定します。
IN0	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、または定数	最初の入力値
IN1	Input	2 進数、整数、浮動小数点数、タイ	I、Q、M、D、L、または定数	2 番目の入力値

		マ、CHAR、WCHAR、TOD、LTOD、DATE、LDT		
INn	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、または定数	オプションの入力値
INELSE	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、または定数	K > n の場合コピーする値を指定します。
RET_VAL	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L	値がコピーされる出力。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL MUX SINT_SINT
      (K := "Tag_Number"
       IN0 := "Tag_Value_1"
       IN1 := "Tag_Value_2"
       INELSE := "Tag_Value_3"
       RET_VAL => "Tag_Result"
       )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

パラメータ	オペランド	値
K	Tag_Number	1
IN0	Tag_Value_1	DW#16#00000000
IN1	Tag_Value_2	DW#16#3E4A7D
INELSE	Tag_Value_3	DW#16#FFFF0000
RET_VAL	Tag_Result	DW#16#3E4A7D

「Tag_Number」オペランドの値に応じて、「Tag_Value_1」入力の値がコピーされ、「Tag_Result」出力のオペランドに出力されます。

DEMUX: 逆多重化

説明

「逆多重化」命令を使用して、選択した出力に IN 入力の内容をコピーすることができます。 選択可能な出力の数は最大 32 まで拡張できます。番号付けは OUT0 で開始し、各新規出力によって続きます。K パラメータを使用して、IN 入力の内容がコピーされる出力を定義します。 その他の出力は、変更されません。 K パラメータの値が使用可能な出力の数よりも大きい場合、IN 入力の内容が OUTELSE パラメータにコピーされます。

「逆多重化」命令は、IN 入力とすべての出力のタグが同じデータタイプの場合にのみ実行できます。K パラメータは、指定できるのが整数のみであるため、例外です。

OUTELSE 出力の値は、次の条件の 1 つが満たされると無効です。

- K パラメータの値が、使用可能な出力の数を超えていること。
- 命令の実行中にエラーが発生していること。

構文

「逆多重化」命令には、以下の構文を使用します。

```
GRAPH
CALL DEMUX ???_???
      (K  := <operand>
       IN  := <operand>
       OUT0  => <Operand>
       OUT1  => <Operand>
       OUTn  => <Operand>
       OUTELSE  => <Operand>
      )
```

パラメータ

次の表に、「逆多重化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
K	Input	整数	I、Q、M、D、L、または定数	入力値(IN)がコピーされる出力を指定します。
IN	Input	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L、または定数	入力値
OUT0	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、	I、Q、M、D、L	最初の出力

		WCHAR、TOD、LTOD、DATE、LDT		
OUT1	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L	2 番目の出力
OUTn	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L	オプションの出力
OUTELSE	Output	2 進数、整数、浮動小数点数、タイマ、CHAR、WCHAR、TOD、LTOD、DATE、LDT	I、Q、M、D、L	K > n の場合に入力値(IN)がコピーされる出力。

命令のデータタイプを[???]ドロップダウンリストから選択できます。

使用可能なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
CALL DEMUX SINT_SINT
     (K := "Tag_Number"
      IN := "Tag_Value"
      OUT0 => "Tag_Output_1"
      OUT1 => "Tag_Output_2"
      OUTELSE => "Tag_Output_3"
     )
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

ネットワーク実行前の「逆多重化」命令の入力値:

パラメータ	オペランド	値	
K	Tag_Number	1	4
IN	Tag_Value	DW#16#FFFFFFFF	DW#16#3E4A7D

ネットワーク実行後の「逆多重化」命令の出力値:

パラメータ	オペランド	値	
OUT0	Tag_Output_1	不変	不変
OUT1	Tag_Output_2	DW#16#FFFFFFFF	不変
OUTELSE	Tag_Output_3	不変	DW#16#3E4A7D

「Tag_Number」オペランドの値に応じて、IN 入力の値が対応する出力にコピーされます。

シフトとローテーション

この章には下記に関する情報が記載されています：

- SHR: 右へシフト (S7-1500)
- SHL: 左へシフト (S7-1500)
- ROR: 右へローテーション (S7-1500)
- ROL: 左へローテーション (S7-1500)

SHR: 右へシフト

説明

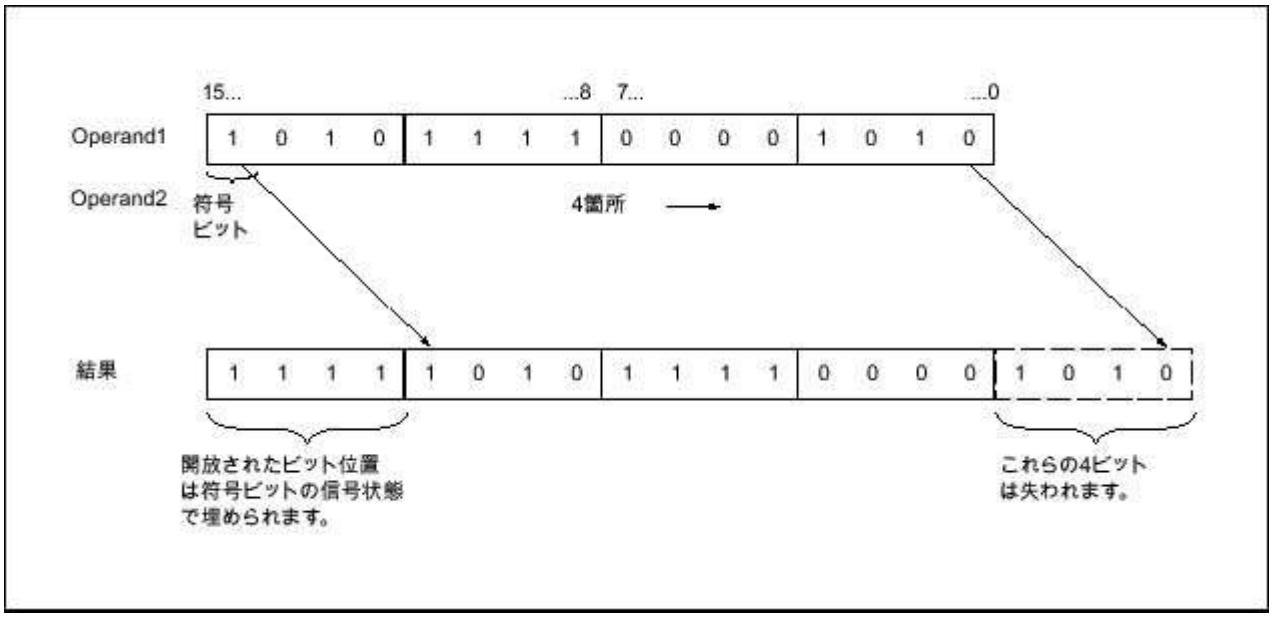
「右へシフト」命令を使用して、Operand1 の内容をビット単位で右へシフトします。 Operand2 を使用して、指定された値をシフトするビット位置の数を指定します。

Operand2 の値が「0」の場合、Operand1 の値が結果にコピーされます。

Operand2 の値が使用可能なビット桁数よりも大きい場合でも、Operand1 の値は使用可能なビット桁数だけ右にシフトされます。

Operand1 の左側の空きビット位置は、符号なしの値がシフトされるとゼロで埋められます。 指定された値に符号が存在する場合、解放されたビット位置は符号ビットのシグナル状態で埋められます。

次の図に、データタイプ INT の Operand1 の内容が、どのように 4 ビット位置右へシフトされるかを示します。



構文

「右へシフト」命令には、以下の構文を使用します。

```
GRAPH
<Result> := SHR_???(<Operand1>,<Operand2>)
```

パラメータ

次の表に、「右へシフト化」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	ビット列、整数	I、Q、M、D、L、または定数	シフトされる値
<Operand2>	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がシフトされるビット位置の数
<Result>	Output	ビット列、整数	I、Q、M、D、L	命令の結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := SHR_INT("Tag_InValue","Tag_Number")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	0011 1111 1010 1111
Tag_Number	3
Tag_OutValue	0000 0111 1111 0101

「Tag_InValue」オペランドの内容が、3 ビット位置右へシフトされます。 結果が「Tag_OutValue」出力に出力されます。

SHL: 左へシフト

説明

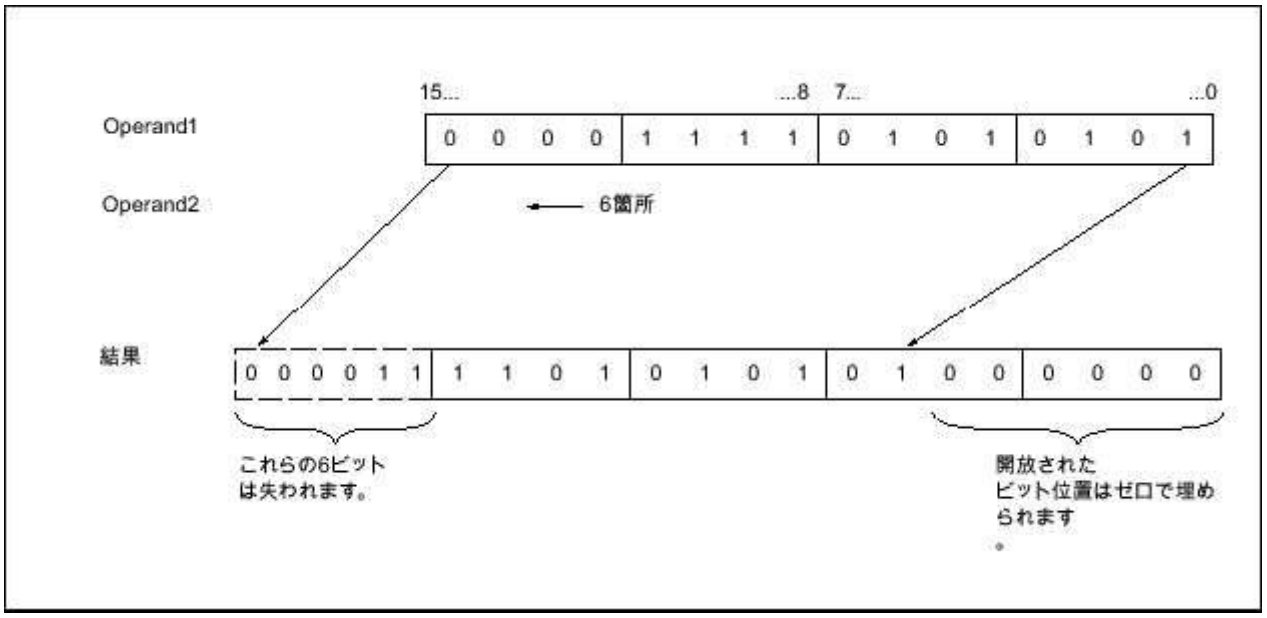
「左へシフト」命令を使用して、Operand1 の内容をビット単位で左へシフトします。 Operand2 を使用して、指定された値をシフトするビット位置の数を指定します。

Operand2 の値が「0」の場合、Operand1 の値が結果にコピーされます。

Operand2 の値が使用可能なビット桁数よりも大きい場合でも、Operand1 の値は使用可能なビット桁数だけ左にシフトされます。

シフトによって解放された Operand1 の右側のビット位置は、ゼロで埋められます。

次の図に、データタイプ WORD の Operand1 の内容が、どのように 6 ビット位置左へシフトされるかを示します。



構文

「左へシフト」命令には、以下の構文を使用します。

```
GRAPH
<Result> := SHL_???(<Operand1>,<Operand2>)
```

パラメータ

次の表に、「左へシフト」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	ビット列、整数	I、Q、M、D、L、または定数	シフトされる値

<Operand2>	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がシフトされるビット位置の数
<Result>	Output	ビット列、整数	I、Q、M、D、L	命令の結果

命令のデータタイプを[???]ドロップダウンリストから選択できます。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := SHL_WORD("Tag_InValue","Tag_Number")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	0011 1111 1010 1111
Tag_Number	4
Tag_OutValue	1111 1010 1111 0000

「Tag_InValue」オペランドの内容が 4 ビット位置左へシフトし、その結果として Tag_OutValue オペランドに出力します。

ROR: 右へローテーション

説明

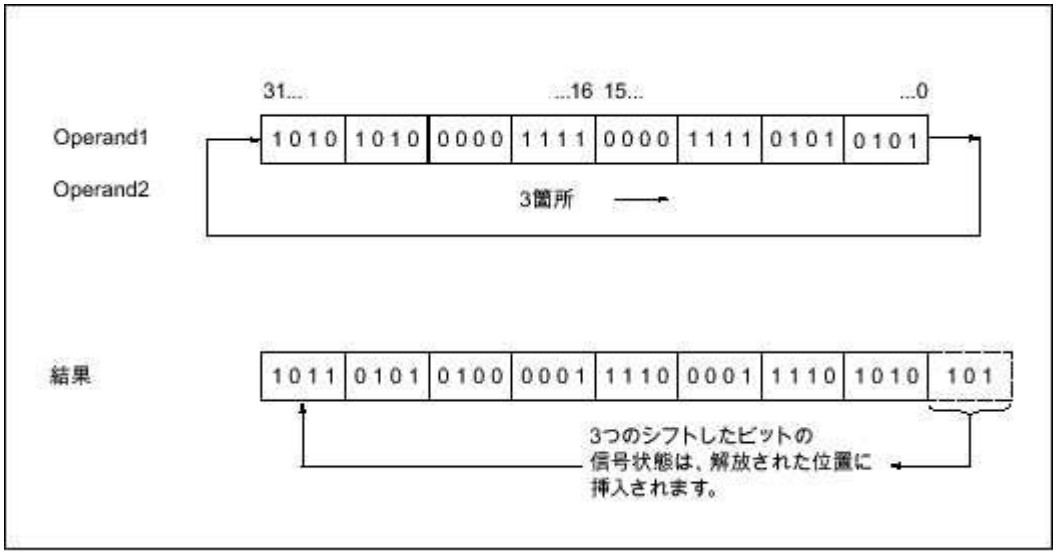
「右へローテーション」命令を使用して、Operand1 の内容をビット単位で右へローテーションします。 Operand2 を使用して、指定された値をローテーションするビット位置の数を指定します。

Operand2 の値が「0」の場合、Operand1 の値が結果にコピーされます。

Operand2 の値が使用可能なビット桁数よりも大きい場合でも、Operand1 の値は指定されたビット桁数だけローテーションされます。

ローテーションによって解放されたビット位置は、押し出されたビット位置で埋められます。

次の図に、データタイプ DWORD のオペランドの内容が、どのように 3 ビット位置右へローテーションされるかを示します。



構文

「右へローテーション」命令には、以下の構文を使用します。

```
GRAPH
<Result> := ROR(<Operand1>,<Operand2>)
```

パラメータ

次の表に、「右へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	ビット列、整数	I、Q、M、D、L、または定数	ローテーションされる値

<Operand2>	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がローテーションされるビット位置の数
<Result>	Output	ビット列、整数	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := ROR("Tag_InValue","Tag_Number")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	1010 1010 0000 1111 0000 1111 0101 0101
Tag_Number	5
Tag_OutValue	1010 1101 0101 0000 0111 1000 0111 1010

「Tag_InValue」オペランドの内容が 5 ビット位置右へシフトし、その結果として Tag_OutValue オペランドに出力します。

ROL: 左へローテーション

説明

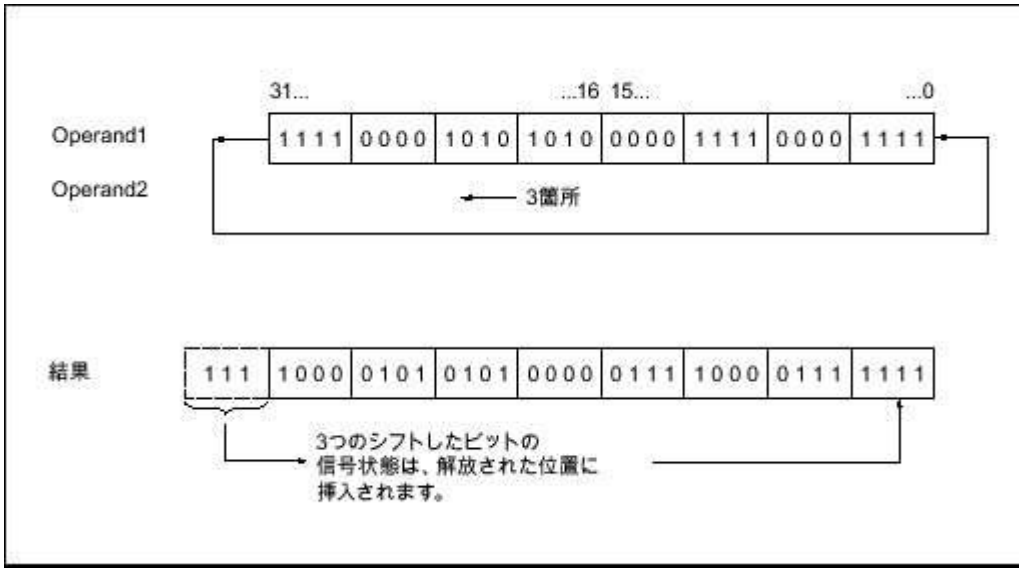
「左へローテーション」命令を使用して、Operand1 の内容をビット単位で左へローテーションします。 Operand2 を使用して、指定された値をローテーションするビット位置の数を指定します。

Operand2 の値が「0」の場合、Operand1 の値が結果にコピーされます。

Operand2 の値が使用可能なビット桁数よりも大きい場合でも、Operand1 の値は指定されたビット桁数だけローテーションされます。

ローテーションによって解放されたビット位置は、押し出されたビット位置で埋められます。

次の図に、データタイプ DWORD のオペランドの内容が、どのように 3 ビット位置左へローテーションされるかを示します。



構文

「左へローテーション」命令には、以下の構文を使用します。

```
GRAPH
<Result> := ROL(<Operand1>,<Operand2>)
```

パラメータ

次の表に、「左へローテーション」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	ビット列、整数	I、Q、M、D、L、または定数	ローテーションされる値

<Operand2>	Input	USINT, UINT, UDINT, ULINT	I、Q、M、D、L、または定数	値がローテーションされるビット位置の数
<Result>	Output	ビット列、整数	I、Q、M、D、L	命令の結果

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

```
GRAPH
"Tag_OutValue" := ROL("Tag_InValue","Tag_Number")
```

次の表に、命令が特定のオペランド値を使用してどのように動作するかを示します。

オペランド	値
Tag_InValue	1111 0000 1010 1010 0000 1111 0000 1111
Tag_Number	5
Tag_OutValue	0001 0101 0100 0001 1110 0001 1111 1110

「Tag_InValue」オペランドの内容が 5 ビット位置左へシフトし、その結果として Tag_OutValue オペランドに出力します。

レガシー

この章には下記に関する情報が記載されています：

DRUM: PLC 実行

説明

「PLC の実装」命令を使用して、プログラムされた出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)に関連するステップの OUT_VAL パラメータのプログラム値を割り当てます。そのため、この命令が特定のステップに留まっている間は、このステップは S_MASK パラメータでプログラムされた許可マスクの条件を満たす必要があります。 このステップのイベントが真(TRUE)になっていて現在のステップのプログラムされた時間が経過した場合、または JOG パラメータの値が「0」から「1」に切り替わった場合、この命令は次のステップに進みます。 RESET パラメータのシグナル状態が「1」に切り替わると、この命令はリセットされます。 これによって、プリセットされたステップ(DSP)と現在のステップが等しくなります。

このステップで消費された時間の量は、プリセットタイムベース(DTBP)と各ステップのプリセットカウンタ値(S_PRESET)の積によって計算されます。 新しいステップの開始時に、この計算値が DCC パラメータにロードされます。このパラメータには、現在のステップの残り時間が含まれています。たとえば、DTBP パラメータの値が「2」で、最初のステップのプリセット値が「100」(100 ミリ秒)の場合、DCC パラメータの値は「200」(200 ミリ秒)になります。

ステップは、時間値、イベント、またはその両方を使用してプログラム可能です。イベントビットおよび時間値「0」を持つステップは、イベントビットのシグナル状態が「1」になると直ちに次のステップに進みます。時間値のみを使用してプログラムされたステップは、時間を直ちに開始します。「0」よりも大きなイベントビットおよび時間値を使用してプログラムされたステップは、イベントビットのシグナル状態が「1」になると時間を開始します。 イベントビットは、シグナル状態「1」で初期化されます。

プログラムされた最後のステップ(LST_STEP)の実行中にこのステップに割り当てられた時間が経過した場合、Q パラメータのシグナル状態は「1」にセットされ、それ以外の場合は「0」にセットされます。 Q パラメータがセットされると、この命令はリセットされるまでこのステップに留まります。

設定可能なマスク(S_MASK)で、出力ワード(OUT_WORD)の個々のビットを選択し、出力値(OUT_VAL)によって出力ビット(OUT1 ~ OUT16)をセットまたはリセットできます。 設定可能なマスクのビットのシグナル状態が「1」の場合、OUT_VAL 値は対応するビットをセットまたはリセットします。 設定可能なマスクビットのシグナル状態が「0」の場合、対応するビットは変更されません。16 のステップすべてに設定可能なマスクのすべてのビットは、シグナル状態「1」で初期化されます。

OUT1 パラメータの出力ビットは、出力ワード(OUT_WORD)の最下位ビットに対応します。OUT16 パラメータの出力ビットは、出力ワード(OUT_WORD)の最上位ビットに対応します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「PLC の実装」命令には、以下の構文を使用します。

```
GRAPH
CALL DRUM, <InstanceDB>
     (RESET := <Operand>
      JOG := <Operand>
```

```
DRUM_EN := <Operand>
LST_STEP := <Operand>
EVENT1 - 16 := <Operand>
OUT1 - 16 => <Operand>
Q => <Operand>
OUT_WORD => <Operand>
ERR_CODE => <Operand>
)
```

パラメータ

次の表に、「PLC 実装」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RESET	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、リセット条件を示します。
JOG	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「0」から「1」に切り替わると、命令が次のステップに進みます。
DRUM_EN	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態が「1」になると、PLC はイベントおよび時間条件に基づいて進むことができます。
LST_STEP	Input	BYTE	I、Q、M、D、L、または定数	プログラムされた最後のステップの番号。
EVENT(i) 1 ≤ i ≤ 16	Input	BOOL	I、Q、M、D、L、または定数	イベントビット(i); 初期シグナル状態は「1」。
OUT(j), 1 ≤ j ≤ 16	Output	BOOL	I、Q、M、D、L	出力ビット(j)
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
OUT_WORD	Output	WORD	I、Q、M、D、L、P	PLC が出力値を書き込むワードアドレス。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
JOG_HIS	Static	BOOL	I、Q、M、D、L、または定数	JOG パラメータの履歴ビット
EOD	Static	BOOL	I、Q、M、D、L、または定数	シグナル状態「1」は、最後のステップに割り当てられた時間が経過したことを示します。
DSP	Static	BYTE	I、Q、M、D、L、P、または定数	PLC のプリセットされた最初のステップ(1～16)
DSC	Static	BYTE	I、Q、M、D、L、P、または定数	PLC の現在のステップ

DCC	Static	DWORD	I、Q、M、D、L、P、または定数	現在のステップの残り処理時間
DTBP	Static	WORD	I、Q、M、D、L、P、または定数	PLC のプリセットタイムベース
PrevTime	Static	TIME	I、Q、M、D、L、または定数	前回の呼び出しのシステム時刻
S_PRESET	Static	ARRAY[1..16] of WORD	I、Q、M、D、L、または定数	1 クロックパルス = 1 ミリ秒の場合の各ステップ[1 ~ 16]のプリセットされたカウンタ値。
OUT_VAL	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの出力値[1 ~ 16, 0 ~ 15]。
S_MASK	Static	ARRAY[1..16, 0..15] of BOOL	I、Q、M、D、L、または定数	各ステップの設定可能なマスク[1 ~ 16, 0 ~ 15]。 初期シグナル状態は「1」。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

ERR_CODE*	説明
W#16#0000	エラーは発生していません。
W#16#000B	LST_STEP パラメータの値が 1 未満であるか、または 16 を超えています。
W#16#000C	DSC パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
W#16#000D	DSP パラメータの値が 1 未満であるか、または LST_STEP パラメータの値を超えています。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例で命令はステップ 1 からステップ 2 に進みます。出力ビット(OUT1 ~ OUT16)および出力ワード(OUT_WORD)はステップ 2 に構成されたマスクおよび OUT_VAL パラメータの値によってセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

```
GRAPH
CALL DRUM, "DRUM_DB"
     (RESET := "Tag_Reset"
      JOG := "Tag_Input_Jog"
      DRUM_EN := "Tag_Input_Drum_EN"
      LST_STEP := "Tag_Number_LastStep"
```

```
EVENTn := "MyTag_Event_n"
OUTn => "MyTag_Output_n"
Q => "Tag_Output_Q"
OUT_WORD => "Tag_OutputWord"
ERR_CODE => "Tag_ErrorCode"
)
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータの初期化には以下の値が使用されます。

パラメータ	オペランド	アドレス	値
RESET	Tag_Reset	M0.0	FALSE
JOG	Tag_Input_JOG	M0.1	FALSE
DRUM_EN	Tag_Input_Drum_EN	M0.2	TRUE
LST_STEP	Tag_Number_LastStep	MB1	B#16#08
EVENT2	MyTag_Event_2	M20.0	FALSE
EVENT4	MyTag_Event_4	M20.1	FALSE
EVENT6	MyTag_Event_6	M20.2	FALSE
EVENT8	MyTag_Event_8	M20.3	FALSE
EVENT10	MyTag_Event_10	M20.4	FALSE
EVENT12	MyTag_Event_12	M20.5	FALSE
EVENT14	MyTag_Event_14	M20.6	FALSE
EVENT16	MyTag_Event_16	M20.7	FALSE

以下の値が命令の「DRUM_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSP	DBB13	W#16#0001
DSC	DBB14	W#16#0001
DCC	DBD16	DW#16#0000000A
DTBP	DBW20	W#16#0001
S_PRESET[1]	DBW26	W#16#0064
S_PRESET[2]	DBW28	W#16#00C8
OUT_VAL[1,0]	DBX58.0	TRUE
OUT_VAL[1,1]	DBX58.1	TRUE
OUT_VAL[1,2]	DBX58.2	TRUE
OUT_VAL[1,3]	DBX58.3	TRUE
OUT_VAL[1,4]	DBX58.4	TRUE

OUT_VAL[1,5]	DBX58.5	TRUE
OUT_VAL[1,6]	DBX58.6	TRUE
OUT_VAL[1,7]	DBX58.7	TRUE
OUT_VAL[1,8]	DBX59.0	TRUE
OUT_VAL[1,9]	DBX59.1	TRUE
OUT_VAL[1,10]	DBX59.2	TRUE
OUT_VAL[1,11]	DBX59.3	TRUE
OUT_VAL[1,12]	DBX59.4	TRUE
OUT_VAL[1,13]	DBX59.5	TRUE
OUT_VAL[1,14]	DBX59.6	TRUE
OUT_VAL[1,15]	DBX59.7	TRUE
OUT_VAL[2,0]	DBX60.0	FALSE
OUT_VAL[2,1]	DBX60.1	FALSE
OUT_VAL[2,2]	DBX60.2	FALSE
OUT_VAL[2,3]	DBX60.3	FALSE
OUT_VAL[2,4]	DBX60.4	FALSE
OUT_VAL[2,5]	DBX60.5	FALSE
OUT_VAL[2,6]	DBX60.6	FALSE
OUT_VAL[2,7]	DBX60.7	FALSE
OUT_VAL[2,8]	DBX61.0	FALSE
OUT_VAL[2,9]	DBX61.1	FALSE
OUT_VAL[2,10]	DBX61.2	FALSE
OUT_VAL[2,11]	DBX61.3	FALSE
OUT_VAL[2,12]	DBX61.4	FALSE
OUT_VAL[2,13]	DBX61.5	FALSE
OUT_VAL[2,14]	DBX61.6	FALSE
OUT_VAL[2,15]	DBX61.7	FALSE
S_MASK[2,0]	DBX92.0	FALSE
S_MASK[2,1]	DBX92.1	TRUE
S_MASK[2,2]	DBX92.2	TRUE
S_MASK[2,3]	DBX92.3	TRUE
S_MASK[2,4]	DBX92.4	TRUE
S_MASK[2,5]	DBX92.5	FALSE
S_MASK[2,6]	DBX92.6	TRUE
S_MASK[2,7]	DBX92.7	TRUE
S_MASK[2,8]	DBX93.0	FALSE
S_MASK[2,9]	DBX93.1	FALSE
S_MASK[2,10]	DBX93.2	TRUE
S_MASK[2,11]	DBX93.3	TRUE
S_MASK[2,12]	DBX93.4	TRUE

S_MASK[2,13]	DBX93.5	TRUE
S_MASK[2,14]	DBX93.6	FALSE
S_MASK[2,15]	DBX93.7	TRUE

出力パラメータは、命令の実行前に以下の値にセットされます。

パラメータ	オペランド	アドレス	値
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#FFFF
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	TRUE
OUT3	MyTag_Output_3	M4.2	TRUE
OUT4	MyTag_Output_4	M4.3	TRUE
OUT5	MyTag_Output_5	M4.4	TRUE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	TRUE
OUT8	MyTag_Output_8	M4.7	TRUE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE
OUT11	MyTag_Output_11	M5.2	TRUE
OUT12	MyTag_Output_12	M5.3	TRUE
OUT13	MyTag_Output_13	M5.4	TRUE
OUT14	MyTag_Output_14	M5.5	TRUE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	アドレス	値
OUT1	MyTag_Output_1	M4.0	TRUE
OUT2	MyTag_Output_2	M4.1	FALSE
OUT3	MyTag_Output_3	M4.2	FALSE
OUT4	MyTag_Output_4	M4.3	FALSE
OUT5	MyTag_Output_5	M4.4	FALSE
OUT6	MyTag_Output_6	M4.5	TRUE
OUT7	MyTag_Output_7	M4.6	FALSE
OUT8	MyTag_Output_8	M4.7	FALSE
OUT9	MyTag_Output_9	M5.0	TRUE
OUT10	MyTag_Output_10	M5.1	TRUE

OUT11	MyTag_Output_11	M5.2	FALSE
OUT12	MyTag_Output_12	M5.3	FALSE
OUT13	MyTag_Output_13	M5.4	FALSE
OUT14	MyTag_Output_14	M5.5	FALSE
OUT15	MyTag_Output_15	M5.6	TRUE
OUT16	MyTag_Output_16	M5.7	FALSE
Q	Tag_Output_Q	M6.0	FALSE
OUTWORD	Tag_OutputWord	MW8	W#16#4321
ERR_CODE	Tag_ErrorCode	MW10	W#16#0000

以下の値が命令の「DRUM_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
JOG_HIS	DBX12.0	FALSE
EOD	DBX12.1	FALSE
DSC	DBB14	W#16#0002
DCC	DBD16	DW#16#000000C8

DCAT: ディスクリート制御タイマアラーム

説明

「ディスクリート制御タイマアラーム」を使用して、CMD パラメータが開くまたは閉じるためのコマンドを発行した時点からの時間をカウントします。 時間は、指定した時間内でプリセット時間(PT)を超えるか、またはデバイスの開閉に関する情報を受信するまで蓄積します(O_FB または C_FB)。 デバイスの開閉に関する情報を受信する前にプリセット済み時間が経過すると、対応するアラームが有効になります。 プリセット済み時間の前にコマンド入力のシグナル状態が切り替わると、この時間経過が再開します。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「ディスクリート制御タイマアラーム」命令は入力条件に対して以下のように応答します。

- CMD パラメータのシグナル状態が「0」から「1」に切り替わると、Q, CMD_HIS、ET (ET < PT の場合のみ)、OA、および CA パラメータのシグナル状態は、以下のような影響を受けます。
  - Q および CMD_HIS パラメータが「1」にセットされます。
  - ET、OA、および CA パラメータが「0」にリセットされます。
- CMD パラメータのシグナル状態が「1」から「0」に切り替わると、Q、ET (ET < PT の場合のみ)、OA、CA、および CMD_HIS パラメータが「0」にリセットされます。
- CMD および CMD_HIS パラメータのシグナル状態が「1」で、O_FB パラメータが「0」にセットされると、命令が最後に実行されてからの時間差(ミリ秒)がパラメータ ET の値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、OA パラメータのシグナル状態が「1」にセットされます。 ET パラメータの値が PT パラメータの値を超えなければ、OA パラメータのシグナル状態が「0」にリセットされます。 CMD_HIS パラメータの値が、CMD パラメータの値にリセットされます。
- CMD、CMD_HIS、および O_FB パラメータのシグナル状態が「1」にセットされ、C_FB パラメータの値が「0」の場合、OA パラメータのシグナル状態が「0」にセットされます。 ET パラメータの値が、PT パラメータの値にセットされます。 O_FB パラメータのシグナル状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 CMD_HIS パラメータの値が、CMD パラメータの値にセットされます。
- CMD、CMD_HIS、および C_FB パラメータの値が「0」の場合、命令が最後に実行されてからの時間差(ms)が ET パラメータの値に加算されます。 ET パラメータの値が PT パラメータの値を超えると、CA パラメータのシグナル状態が「1」にリセットされます。 PT パラメータの値を超えなければ、CA パラメータのシグナル状態は「0」です。 CMD_HIS パラメータの値が、CMD パラメータの値にセットされます。
- CMD、CMD_HIS、および O_FB パラメータのシグナル状態が「0」のときに C_FB パラメータが「1」にセットされると、CA パラメータが「0」にセットされます。 ET パラメータの値が、PT パラメータの値にセットされます。 C_FB パラメータのシグナル状態「0」に切り替わると、命令が次回実行されるときにアラームがセットされます。 CMD_HIS パラメータの値が、CMD パラメータの値にセットされます。
- O_FB および C_FB パラメータのシグナル状態が同時に「1」の場合、両方のアラーム出力のシグナル状態が「1」にセットされます。

「ディスクリート制御タイマアラーム」命令は、エラー情報を返しません。

構文

「ディスクリート制御タイマアラーム」命令には、以下の構文を使用します。

```
GRAPH
CALL DCAT, <InstanceDB>
     (CMD := <Operand>
      O_FB := <Operand>
      C_FB := <Operand>
      Q => <Operand>
      OA => <Operand>
      CA => <Operand>
      )
```

パラメータ

次の表に、「ディスクリート制御タイマアラーム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CMD	Input	BOOL	I、Q、M、D、L、または定数	シグナル状態「0」は、「閉じる」コマンドを示します。 シグナル状態「1」は、「開く」コマンドを示します。
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
Q	Output	BOOL	I、Q、M、D、L	CMD パラメータのステータスを示します
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒。
PT	Static	DINT	D、L、または定数	プリセットタイマ値(1 クロックパルス = 1 ミリ秒)
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
CMD_HIS	Static	BOOL	D、L、または定数	CMD の履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例では、CMD パラメータが「0」から「1」に切り替わります。 命令の実行後、Q パラメータが「1」にセットされ、2 つのアラーム出力 OA および CA のシグナル状態は「0」になります。 インスタンスデータブロックの CMD_HIS パラメータがシグナル状態「1」にセットされ、ET パラメータが「0」にリセットされます。

注記

静的なパラメータをデータブロックで初期化できます。

```
GRAPH
CALL DCAT, "DCAT_DB"
      (CMD := "Tag_Input_CMD"
       O_FB := "Tag_Input_O_FB"
       C_FB := "Tag_Input_C_FB"
       Q => "Tag_Output_Q"
       OA => "Tag_Output_OA"
       CA => "Tag_Output_CA"
       )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
CMD	Tag_Input_CMD	TRUE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
Q	Tag_Output_Q	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令の「DCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#12
PT	DBD8	L#222
CMD_HIS	DBX16.0	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
Q	Tag_Output_Q	TRUE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE

以下の値が命令の「DCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
CMD_HIS	DBX16.0	TRUE

MCAT: モータ制御タイマアラーム

説明

命令「モータ制御タイマアラーム」を使用して、コマンド入力(開または閉)の 1 つが有効になる時点からの時間を累積します。 時間はプリセット済み時間が経過するまで、または関連のフィードバック入力が、デバイスが指定された時間内に要求された操作を実行したことを示す場合に蓄積されます。フィードバックが受信される前にプリセット済み時間が経過すると、対応するアラームがトリガされます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

「モータ制御タイマアラーム」命令の実行

次の表に、さまざまな入力条件に対する「モータ制御タイマアラーム」命令の応答を示します。

入力パラメータ								出力パラメータ								
ET	O_HIS	C_HIS	O_CMD	C_CMD	S_CMD	O_FB	C_FB	OO	CO	OA	CA	ET	O_HIS	C_HIS	Q	ステータス
X	1	1	X	X	X	X	X	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	X	1	1	0	0	1	1	PT	0	0	0	Alarm (アラーム)
X	X	X	X	X	1	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	X	X	1	1	X	X	X	0	0	0	0	X	0	0	1	Stop (停止)
X	0	X	1	0	0	X	X	1	0	0	0	0	1	0	1	Start opening (開き始める)
<PT	1	0	X	0	0	0	X	1	0	0	0	INC	1	0	1	Open (開く)
X	1	0	X	0	0	1	0	0	0	0	0	PT	1	0	1	Opened (開いた)
>=PT	1	0	X	0	0	0	X	0	0	1	0	PT	1	0	0	Opening alarm (アラームを開いている)
X	X	0	0	1	0	X	X	0	1	0	0	0	0	1	1	Start closing

															(閉じ始める)
<PT	0	1	0	X	0	X	0	0	1	0	0	INC	0	1	1	Close (閉じる)
X	0	1	0	X	0	0	1	0	0	0	0	PT	0	1	1	Closed (閉じた)
>= PT	0	1	0	X	0	X	0	0	0	0	1	PT	0	1	0	Closing alarm (アラームを閉じている)
X	0	0	0	0	0	X	X	0	0	0	0	X	0	0	1	Stopped (停止)

凡例:

INC	FB から ET への最後に処理されてからの時間差(ms)を追加します。
PT	PT は ET と同じ値にセットされます。
X	使用不可
<PT	ET < PT
>=PT	ET >= PT

入力パラメータ O_HIS および C_HIS のシグナル状態が両方とも「1」の場合、これらのパラメータのシグナル状態が直ちに「0」にセットされます。 この場合、上記の表の最後の行(X)が有効になります。 入力パラメータ O_HIS および C_HIS のシグナル状態が「1」であるかどうかをチェックすることができないため、この場合の出力パラメータは以下のようにセットされます。

OO = FALSE

CO = FALSE

OA = FALSE

CA = FALSE

ET = PT

Q = TRUE

構文

以下の構文が、「モータ制御タイマアラーム」命令に使用されます。

```
GRAPH
CALL MCAT, <InstanceDB>
     (O_CMD := <operand>
      C_CMD := <operand>
      S_CMD := <operand>
      O_FB := <operand>
      C_FB := <operand>
      OO => <Operand>
       CO => <Operand>
      OA => <Operand>
```

```
CA => <Operand>
Q => <Operand>
)
```

パラメータ

次の表に、「モータ制御タイマアラーム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
O_CMD	Input	BOOL	I、Q、M、D、L、または定数	[開]コマンド入力
C_CMD	Input	BOOL	I、Q、M、D、L、または定数	[閉]コマンド入力
S_CMD	Input	BOOL	I、Q、M、D、L、または定数	[停止]コマンド入力
O_FB	Input	BOOL	I、Q、M、D、L、または定数	開くときにフィードバック入力
C_FB	Input	BOOL	I、Q、M、D、L、または定数	閉じるときにフィードバック入力
OO	Output	BOOL	I、Q、M、D、L	[開]出力
CO	Output	BOOL	I、Q、M、D、L	[閉]出力
OA	Output	BOOL	I、Q、M、D、L	開くときにアラーム出力
CA	Output	BOOL	I、Q、M、D、L	閉じるときにアラーム出力
Q	Output	BOOL	I、Q、M、D、L	シグナル状態「0」は、エラー条件を示します。
ET	Static	DINT	D、L、または定数	現在の経過時間、1 クロックパルス = 1 ミリ秒
PT	Static	DINT	D、L、または定数	プリセットタイマ値(1 クロックパルス = 1 ミリ秒)
PREV_TIME	Static	DWORD	D、L、または定数	前のシステム時間
O_HIS	Static	BOOL	D、L、または定数	[開]履歴ビット
C_HIS	Static	BOOL	D、L、または定数	[閉]履歴ビット

有効なデータタイプの追加情報については、「関連項目」を参照してください。

例

次の例で、命令がどのように動作するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。

```
GRAPH
CALL MCAT, "MCAT_DB"
     (O_CMD := "Tag_Input_O_CMD"
      C_CMD := "Tag_Input_C_CMD"
```

```
S_CMD := "Tag_Input_S_CMD"
O_FB := "Tag_Input_O_FB"
C_FB := "Tag_Input_C_FB"
OO => "Tag_OutputOpen"
CO => "Tag_OutputClosed"
OA => "Tag_Output_OA"
CA => "Tag_Output_CA"
Q => "Tag_Output_Q"
)
```

次の表に、命令が特定の値を使用してどのように動作するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
O_CMD	Tag_Input_O_CMD	TRUE
C_CMD	Tag_Input_C_CMD	FALSE
S_CMD	Tag_Input_S_CMD	FALSE
O_FB	Tag_Input_O_FB	FALSE
C_FB	Tag_Input_C_FB	FALSE
OO	Tag_OutputOpen	FALSE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE
CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	FALSE

以下の値が命令の「MCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#2
PT	DBD8	L#22
O_HIS	DBX16.0	TRUE
C_HIS	DBX16.1	FALSE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OO	Tag_OutputOpen	TRUE
CO	Tag_OutputClosed	FALSE
OA	Tag_Output_OA	FALSE

CA	Tag_Output_CA	FALSE
Q	Tag_Output_Q	TRUE

以下の値が命令の「MCAT_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
ET	DBD4	L#0
O_HIS	DBX16.0	TRUE
CMD_HIS	DBX16.1	FALSE

IMC: 入力ビットをマスクビットと比較

説明

「入力ビットをマスクビットと比較」命令を使用して最大 16 のプログラム済み入力ビット(IN_BIT0～IN_BIT15)のシグナル状態を対応するマスクビットと比較します。 最大 16 のマスクされたステップをプログラム可能です。 IN_BIT0 パラメータの値が、CMP_VAL[x,0]マスクの値と比較されます。「x」はステップ番号を示しています。 CMP_STEP パラメータで、比較に使用するマスクのステップ番号を指定します。 プログラムされた値は、すべて同じ方法で比較されます。 プログラムされていない入力ビットやマスクビットは、既定のシグナル状態である FALSE になります。

比較で一致が検出されると、OUT パラメータのシグナル状態が「1」にセットされます。 それ以外の場合、OUT パラメータが「0」にセットされます。

CMP_STEP パラメータの値が 15 よりも大きい場合、この命令は実行されません。 ERR_CODE パラメータでエラーメッセージが出力されます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「入力ビットをマスクビットと比較」命令には、以下の構文を使用します。

```
GRAPH
CALL IMC, <InstanceDB>
     (IN_BIT0 - 15 := <Operand>
      CMP_STEP := <Operand>
      OUT => <Operand>
      ERR_CODE => <Operand>
      )
```

パラメータ

次の表に、「入力ビットをマスクビットと比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。

IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
CMP_STEP	Input	BYTE	I、Q、M、D、L、P、または定数	比較に使用されるマスクのステップ番号。
OUT	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、一致が検出されたことを示します。 シグナル状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000A	CMP_STEP パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

以下の例では、すべての 16 の入力ビットがステップ 2 のマスクと比較されます。入力ビットがステップ 2 のマスクに一致するため、OUT パラメータのシグナル状態は TRUE にセットされます。

注記
静的なパラメータをデータブロックで初期化できます。

```
GRAPH
CALL IMC, "IMC_DB"
     (IN_BIT0 - 15 := "Tag_Input_BITn"
      CMP_STEP := "Tag_CMP_STEP"
      OUT => "Tag_Output"
      ERR_CODE => "Tag_ErrorCode"
      )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE
IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE

IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
CMP_STEP	Tag_CMP_STEP	B#16#02
OUT	Tag_Output	FALSE
ERR_CODE	Tag_ErrorCode	W#16#0000

ステップ 2 のマスクの以下の値が命令の「IMC_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output	TRUE
ERR_CODE	Tag_ErrorCode	W#16#0000

SMC: スキャンマトリックスの比較

説明

「スキャンマトリックスの比較」命令を使用して最大 16 のプログラム済み入力ビット(IN_BIT0 ~ IN_BIT15)のシグナル状態を各ステップの比較マスクの対応するビットと比較します。 処理はステップ 1 から開始し、プログラムされた最後のステップ(LAST)まで、または一致が検出されるまで続行されます。 IN_BIT0 パラメータの入力ビットが、CMP_VAL[x,0]マスクの値と比較されます。「x」はステップ番号を示しています。 プログラムされた値は、すべて同じ方法で比較されます。 一致が検出されると、OUT パラメータのシグナル状態が「1」にセットされ、一致するマスクのステップ番号が OUT_STEP パラメータに書き込まれます。 プログラムされていない入力ビットやマスクビットは、既定のシグナル状態である FALSE になります。 複数のステップに一致するマスクが存在する場合、最初に検出されたマスクのみが OUT_STEP パラメータに示されます。 一致が検出されないと、OUT パラメータのシグナル状態が「0」にセットされます。 この場合、OUT_STEP パラメータの値は LAST パラメータの値よりも「1」だけ大きくなります。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「スキャンマトリックスの比較」命令には、以下の構文を使用します。

```
GRAPH
CALL SMC, <InstanceDB>
     (IN_BIT0 - 15 := <Operand>
      OUT => <Operand>
      OUT_STEP => <Operand>
      ERR_CODE => <Operand>
      )
```

パラメータ

次の表に、「スキャンマトリックスの比較」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN_BIT0	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 0 がマスクビット 0 と比較されます。
IN_BIT1	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 1 がマスクビット 1 と比較されます。
IN_BIT2	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 2 がマスクビット 2 と比較されます。

IN_BIT3	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 3 がマスクビット 3 と比較されます。
IN_BIT4	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 4 がマスクビット 4 と比較されます。
IN_BIT5	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 5 がマスクビット 5 と比較されます。
IN_BIT6	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 6 がマスクビット 6 と比較されます。
IN_BIT7	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 7 がマスクビット 7 と比較されます。
IN_BIT8	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 8 がマスクビット 8 と比較されます。
IN_BIT9	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 9 がマスクビット 9 と比較されます。
IN_BIT10	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 10 がマスクビット 10 と比較されます。
IN_BIT11	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 11 がマスクビット 11 と比較されます。
IN_BIT12	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 12 がマスクビット 12 と比較されます。
IN_BIT13	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 13 がマスクビット 13 と比較されます。
IN_BIT14	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 14 がマスクビット 14 と比較されます。
IN_BIT15	Input	BOOL	I、Q、M、D、L、または定数	入力ビット 15 がマスクビット 15 と比較されます。
OUT	Output	BOOL	I、Q、M、D、L	シグナル状態「1」は、一致が検出されたことを示します。 シグナル状態「0」は、一致が検出されなかったことを示します。
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
OUT_STEP	Output	BYTE	I、Q、M、D、L、P	一致するマスクを持つステップの番号が含まれているか、または一致が検出されない場合は、LAST パラメータの値よりも「1」だけ大きいステップ番号が含まれています。
LAST	Static	BYTE	I、Q、M、D、L、P、または定数	一致するマスクをスキャンする最後のステップのステップ番号を指定します。
CMP_VAL	Static	ARRAY OF WORD	I、Q、M、D、L、または定数	比較マスク[0 ~ 15, 0 ~ 15]: インデックスの最初の番号はステップ番号で、2 番目の番号はマスクビット番号です。

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
000E	LAST パラメータの値が 15 よりも大きくなっています。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

この例で、すべての 16 個の入力ビットは一致が検出されるまでステップ 0～5 までのマスクと比較されます。 ステップ 2 が入力ビットと一致したため、ステップ 0～2 のマスクのみがスキャンされました。

注記
静的なパラメータをデータブロックで初期化できます。

```
GRAPH
CALL SMC, "SMC_DB"
     (IN_BIT0 - 15 := "Tag_Input_BITn"
      OUT => "Tag_Output"
      OUT_STEP => "Tag_Output_STEP"
      ERR_CODE => "Tag_ErrorCode"
      )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータと出力パラメータに以下の値が使用されます。

パラメータ	オペランド	値
IN_BIT0	Tag_Input_BIT0	TRUE
IN_BIT1	Tag_Input_BIT1	TRUE
IN_BIT2	Tag_Input_BIT2	FALSE
IN_BIT3	Tag_Input_BIT3	TRUE
IN_BIT4	Tag_Input_BIT4	TRUE
IN_BIT5	Tag_Input_BIT5	FALSE
IN_BIT6	Tag_Input_BIT6	TRUE
IN_BIT7	Tag_Input_BIT7	TRUE
IN_BIT8	Tag_Input_BIT8	FALSE
IN_BIT9	Tag_Input_BIT9	TRUE
IN_BIT10	Tag_Input_BIT10	TRUE

IN_BIT11	Tag_Input_BIT11	FALSE
IN_BIT12	Tag_Input_BIT12	TRUE
IN_BIT13	Tag_Input_BIT13	TRUE
IN_BIT14	Tag_Input_BIT14	FALSE
IN_BIT15	Tag_Input_BIT15	TRUE
OUT	Tag_Output	FALSE
OUT_STEP	Tag_Output_STEP	B#16#00
ERR_CODE	Tag_ErrorCode	W#16#0000

ステップ 2 のマスクの以下の値が命令の「SMC_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
CMP_VAL [2,0]	DBX12.0	TRUE
CMP_VAL [2,1]	DBX12.1	TRUE
CMP_VAL [2,2]	DBX12.2	FALSE
CMP_VAL [2,3]	DBX12.3	TRUE
CMP_VAL [2,4]	DBX12.4	TRUE
CMP_VAL [2,5]	DBX12.5	FALSE
CMP_VAL [2,6]	DBX12.6	TRUE
CMP_VAL [2,7]	DBX12.7	TRUE
CMP_VAL [2,8]	DBX13.0	FALSE
CMP_VAL [2,0]	DBX13.1	TRUE
CMP_VAL [2,10]	DBX13.2	TRUE
CMP_VAL [2,11]	DBX13.3	FALSE
CMP_VAL [2,12]	DBX13.4	TRUE
CMP_VAL [2,13]	DBX13.5	TRUE
CMP_VAL [2,14]	DBX13.6	FALSE
CMP_VAL [2,15]	DBX13.7	TRUE
LAST	DB84	B#16#05

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output	TRUE
OUT_STEP	Tag_Output_STEP	B#16#02
ERR_CODE	Tag_ErrorCode	W#16#0000

LEAD_LAG: リード/ラグアルゴリズム

説明

「リード/ラグアルゴリズム」命令を使用して、アナログタグの信号を処理します。 GAIN パラメータのゲイン値はゼロよりも大きくなければなりません。「リード/ラグアルゴリズム」命令の結果は、以下の等式を使用して計算されます。

$$OUT = \left[\frac{LG\_TIME}{LG\_TIME + SAMPLE\_T}\right] PREV\_OUT + GAIN \left[\frac{LD\_TIME + SAMPLE\_T}{LG\_TIME + SAMPLE\_T}\right] IN - GAIN \left[\frac{LD\_TIME}{LG\_TIME + SAMPLE\_T}\right] * PREV\_IN$$

「リード/ラグアルゴリズム」命令は、処理を固定プログラムサイクルで行う場合のみ正しい結果が得られます。 LD_TIME、LG_TIME、および SAMPLE_T パラメータには同一のユニットを指定する必要があります。 LG_TIME > 4 + SAMPLE_T の場合、この命令は以下の関数を実行します。

OUT = GAIN * ((1 + LD_TIME * s) / (1 + LG_TIME * s)) * IN

GAIN パラメータの値がゼロ以下の場合、計算は実行されず、ERR_CODE パラメータでエラー情報が出力されます。

「リード/ラグアルゴリズム」命令をダイナミックフィードフォワード制御でループと共に補償器として使用します。 この命令は、2 つの操作で構成されています。「Lead」演算は OUT 出力のフェーズをシフトして、出力が入力の前に来るようにします。 一方「Lag」演算は出力をシフトして、出力が入力の後になるようにします。「Lag」演算は積分に相当するため、ノイズサプレッサやローパスフィルタとして使用できます。「Lead」演算は微分と同等なため、ハイパスフィルタとして使用できます。 2 つの演算(Lead および Lag)をまとめて実行すると、低周波域では出力フェーズが入力フェーズよりも遅れ、高周波域では出力フェーズが入力フェーズに先行します。 つまり「リード/ラグアルゴリズム」命令は帯域フィルタとして使用できます。

プログラムに命令を挿入すると[呼び出しオプション]ダイアログが開き、ここでブロックパラメータが個別のデータブロック(シングルインスタンス)に保存されるか、またはブロックインターフェースにローカルタグ(マルチインスタンス)として保存されるかを指定できます。 個別のデータブロックを作成した場合、このデータブロックは[プログラムブロック|システムブロック]にあるプロジェクトツリーの[プログラムリソース]フォルダにあります。 このトピックの詳細については、「関連項目」を参照してください。

構文

「リード/ラグアルゴリズム」命令には、以下の構文を使用します。

```
GRAPH
CALL LEAD_LAG, <InstanceDB>
     (IN := <Operand>
      SAMPLE_T := <Operand>
      OUT => <Operand>
      ERR_CODE => <Operand>
      )
```

パラメータ

次の表に、「リード/ラグアルゴリズム」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	REAL	I、Q、M、D、L、P、または定数	処理対象の現在のサンプル時間(サイクルタイム)の入力値。
SAMPLE_T	Input	INT	I、Q、M、D、L、P、または定数	サンプル時間
OUT	Output	REAL	I、Q、M、D、L、P	命令の結果
ERR_CODE	Output	WORD	I、Q、M、D、L、P	エラー情報
LD_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のリードタイム。
LG_TIME	Static	REAL	I、Q、M、D、L、P、または定数	サンプル時間としての同一ユニット内のラグタイム。
GAIN	Static	REAL	I、Q、M、D、L、P、または定数	% / %で表されるゲイン(定常状態時の出力の切り替えと入力の切り替えとの比率)。
PREV_IN	Static	REAL	I、Q、M、D、L、P、または定数	前の入力
PREV_OUT	Static	REAL	I、Q、M、D、L、P、または定数	前の出力

有効なデータタイプの追加情報については、「関連項目」を参照してください。

ERR_CODE パラメータ

次の表に、ERR_CODE パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0009	GAIN パラメータの値がゼロ以下です。
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例で、命令がどのように機能するかを示します。

注記
静的なパラメータをデータブロックで初期化できます。

```
GRAPH
CALL LEAD_LAG, "LEAD_LAG_DB"
     (IN := "Tag_Input"
      SAMPLE_T := "Tag_Input_SAMPLE_T"
      OUT => "Tag_Output_Result"
      ERR_CODE => "Tag_ErrorCode"
      )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

処理前

この例では、入力パラメータで以下の値が使用されます。

パラメータ	オペランド	値
IN	Tag_Input	2.0
SAMPLE_T	Tag_Input_SAMPLE_T	10

以下の値が命令の「LEAD_LAG_DB」インスタンスデータブロックに保存されます。

パラメータ	アドレス	値
LD_TIME	DBD12	2.0
LG_TIME	DBD16	2.0
GAIN	DBD20	1.0
PREV_IN	DBD24	6.0
PREV_OUT	DBD28	6.0

処理後

命令が実行された後に、以下の値が出力パラメータに書き込まれます。

パラメータ	オペランド	値
OUT	Tag_Output_Result	2.0

以下の値が命令の「LEAD_LAD_DB」インスタンスデータブロックに保存されます。

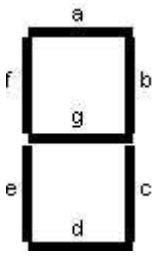
パラメータ	オペランド	値
PREV_IN	DBD24	2.0
PREV_OUT	DBD28	2.0

SEG: 7 セグメント表示のビットパターンの作成

説明

「7 セグメント表示のビットパターンの作成」命令は、指定されたソースワード(IN)内の 16 進数の 4 桁を 1 つずつ 7 セグメント表示用の 4 つの同等のコードに変換し、それを出力(OUT)のダブルワードに書き込みます。

入力の 16 進数値と出力のビットパターンの関係は以下のようになります。

桁	-gfedcba	表示	7 セグメント表示
0000	00111111	0	
0001	00000110	1	
0010	01011011	2	
0011	01001111	3	
0100	01100110	4	
0101	01101101	5	
0110	01111101	6	
0111	00000111	7	
1000	01111111	8	
1001	01100111	9	
1010	01110111	A	
1011	01111100	B	
1100	00111001	C	
1101	01011110	D	
1110	01111001	E	
1111	01110001	F	

構文

「7 セグメント表示のビットパターンの作成」命令には、以下の構文を使用します。

```
GRAPH
CALL SEG
     (IN := <Operand>
      OUT => <Operand>
      )
```

パラメータ

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	WORD	I、Q、M、D、L、P、または定数	4 つの 16 進数のソースワード
OUT	Output	DWORD	I、Q、M、D、L、P	変換先の 4 バイトのビットパターン

「7 セグメント表示のビットパターンの作成」は、いずれのエラー条件も認識しません。

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
CALL SEG
     (IN := "Tag_Input"
      OUT => "Tag_Output"
      )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値	
		16 進数	2 進数
IN	Tag_Input	W#16#1234	0001 0010 0011 0100
OUT	Tag_Output	DW#16065B4F66	000 00110 0101 1011 0100 1111 0110 0110 表示: 1234

BCDCPL: 10 の補数の作成

説明

「10 の補数の作成」命令を使用して、IN パラメータで指定された 7 桁の BCD 数の 10 の補数を作成します。 この命令は、次の数式を使用して計算します。

10000000 (BCD として)

– 7 桁の BCD 数

---

10 の補数(BCD として)

構文

「10 の補数の作成」命令には、以下の構文を使用します。

```
GRAPH
CALL BCDCPL
      (IN := <Operand>
       ERR_CODE => <Operand>
       )
```

パラメータ

次の表に、「10 の補数の作成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	ビット列	I、Q、M、D、L、P、または定数	7 桁の BCD 数
ERR_CODE	Output	DWORD	I、Q、M、D、L、P	命令の結果

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
CALL BCDCPL
      (IN := "Tag_Input"
       ERR_CODE => "Tag_Output"
       )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#01234567
ERR_CODE	Tag_Output	DW#16#08765433
*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

BITSUM: セットビット数カウント

説明

「セットビット数のカウント」命令を使用して、シグナル状態「1」にセットされるオペランドのビット数をカウントします。 ビットをカウントするオペランドは、IN パラメータで指定されます。 命令の結果は、RET_VAL パラメータに出力されます。

構文

「セットビット数カウント」命令には、以下の構文を使用します。

```
GRAPH
CALL BITSUM
      (IN := <Operand>
       RET_VAL => <Operand>
       )
```

パラメータ

次の表に、「セットビット数カウント」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	DWORD	I、Q、M、D、L、P、または定数	セットビット数をカウントするオペランド
RET_VAL	Output	INT	I、Q、M、D、L、P	セットビット数

例

次の例で、命令がどのように機能するかを示します。

```
GRAPH
CALL BITSUM
      (IN := "Tag_Input"
       RET_VAL => "Tag_Output"
       )
```

次の表に、命令が特定の値を使用してどのように機能するかを示します。

パラメータ	オペランド	値*
IN	Tag_Input	DW#16#12345678
RET_VAL	Tag_Output	W#16#000D (13 ビット)

*エラーコードはプログラミングエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えについては、「関連項目」を参照してください。

LAD 命令比較演算

この章には下記に関する情報が記載されています：

- 全般的 LAD 命令 (S7-1500)
- CMP>T: ステップ有効化時間よりも長い (S7-1500)
- CMP>U: 割り込みなしステップの有効化時間よりも長い (S7-1500)
- CMP>T_MAX: 最大ステップ有効化時間よりも長い (S7-1500)
- CMP>T_WARN: 警告時間よりも長い (S7-1500)

全般的 LAD 命令

説明

GRAPH でプログラミングしている場合、さまざまな LAD 命令が使用可能です。

次の表は、命令およびそれらの可用性の概要を示しています。

LAD 命令	固定の命令	移行/監視/インターロック
全般		
ネットワーク挿入	x	
空ボックス	x	x
分岐を開く	x	x
分岐を閉じる	x	x
-|: 入力の挿入	x	
ビット論理演算		
--| |--: a 接点	x	x
--| / |--: b 接点	x	x
--|NOT|--: RLO の反転	x	x
--( )---: 割り当て	x	
否定割り当て	x	
--(R)---: 出力のリセット	x	
--(S)---: 出力設定	x	
SET_BF: ビットフィールドのセット	x	
RESET_BF: ビットフィールドのリセット	x	
SR: フリップフロップのセット/リセット	x	
RS: フリップフロップのリセット/セット	x	
--|P|--: 立ち上がりエッジのオペランドスキャン	x	
--|N|--: 立ち下がりエッジのオペランドスキャン	x	
--(P)--: 信号立ち上がりエッジのオペランドの設定	x	
--(N)--: 信号立ち下がりエッジのオペランドの設定	x	
P_TRIG: 立ち上がりエッジの RLO スキャン	x	
N_TRIG: 立ち下がりエッジの RLO スキャン	x	
R_TRIG: :信号立ち上がりエッジの検出	x	
F_TRIG: :信号立ち下がりエッジの検出	x	
タイマ		
IEC タイマ		
TP: パルスの生成	x	

TON: オンディレータイマ	x	
TOF: オフディレータイマ	x	
TONR: タイムアキュムレータ	x	
---(TP)---: パルスタイマの起動	x	
---(TON)---: オンディレータイマの起動	x	
---(TOF)---: オフディレータイマの起動	x	
---(TONR)---: タイムアキュムレータ	x	
---(RT)---: タイマのリセット	x	
---(PT)---: 持続時間のロード	x	
SIMATIC タイマ(S7-1500)		
S_PULSE: パルスタイマパラメータの割り当てと開始	x	
S_PEXT: 拡張パルスタイマパラメータを割り当てと起動	x	
S_ODT: オンディレータイマパラメータの割り当てと開始	x	
S_ODTS: 保持型オンディレータイマパラメータの割り当てと開始	x	
S_OFFDT: オフディレータイマパラメータの割り当てと開始	x	
カウンタ		
IEC カウンタ		
CTU: カウントアップ	x	
CTD: カウントダウン	x	
CTUD: カウントアップ/カウントダウン	x	
SIMATIC カウンタ(S7-1500)		
S_CU: パラメータおよびカウントアップの割り当て	x	
S_CD: パラメータおよびカウントダウンの割り当て	x	
S_CUD: パラメータおよびカウントアップ/カウントダウンの割り当て	x	
比較演算		
CMP ==: 等価	x	x
CMP <>: 不等価	x	x
CMP >=: 以上	x	x
CMP <=: 以下	x	x
CMP >: 超過	x	x
CMP <: 未満	x	x
IN_RANGE: 範囲内の値	x	
OUT_RANGE: 範囲外の値	x	
---| OK |---: 有効性のチェック	x	
---| NOT_OK |---: 無効性のチェック	x	
EQ_Type: EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較	x	
NE_Type: UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較	x	
EQ_ElemType: EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較	x	

NE_ElemType: UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較	x	
IS_NULL: EQUALS ZERO ポインタの照会	x	
NOT_NULL: UNEQUALS ZERO ポインタの照会	x	
IS_ARRAY: 配列のチェック	x	
CMP>T: ステップ有効化時間よりも長い		x
CMP>U: 割り込みなしステップの有効化時間よりも長い		x
四則演算		
CALCULATE: 計算	x	
ADD: 加算	x	
SUB: 減算	x	
MUL: 乗算	x	
DIV: 除算	x	
MOD: 除算の剰余を返す	x	
NEG: 2 の補数の作成	x	
INC: インクリメント	x	
DEC: デクリメント	x	
ABS: 絶対値の形成	x	
MIN: 最小値の取得	x	
MAX: 最大値の取得	x	
LIMIT: 制限値の設定	x	
SQR: 2 乗値の形成	x	
SQRT: 平方根の形成	x	
LN: 自然対数の形成	x	
EXP: 指数値の形成	x	
SIN: 正弦値の形成	x	
COS: 余弦値の形成	x	
TAN: 正接値の形成	x	
ASIN: 逆正弦値の形成	x	
ACOS: 逆余弦値の形成	x	
ATAN: 逆正接値の形成	x	
FRAC: 小数部を返す		
EXPT: 累乗	x	
ムーブ操作		
MOVE: 移動値	x	
Deserialize: 逆シリアル化	x	
Serialize: シリアル化	x	
FieldRead: フィールドの読み出し	x	
FieldWrite: フィールドの書き込み	x	
MOVE_BLK: ブロックの移動	x	

MOVE_BLK_VARIANT: ブロックの移動	x	
UMOVE_BLK: 割り込みなしブロック転送	x	
FILL_BLK: フィル命令	x	
UFILL_BLK: 割り込み不可能なフィル命令	x	
SWAP: スワップ	x	
ReadFromArrayDB: ARRAY データブロックからの読み出し	x	
WriteToArrayDB: ARRAY データブロックへの書き込み	x	
ReadFromArrayDBL: ロードメモリの ARRAY データブロックからの読み出し	x	
WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み	x	
VariantGet: VARIANT タグ値の読み出し	x	
VariantPut: VARIANT タグ値の書き込み	x	
CountOfElements: ARRAY エレメント数の取得	x	
BLKMOV: ブロックの移動(S7-1500)	x	
UBLKMOV: 割り込み不可能なブロックの移動(S7-1500)	x	
FILL: フィル命令(S7-1500)	x	
変換操作		
CONVERT: 値の変換	x	
ROUND: 数値の四捨五入	x	
CEIL: 浮動小数点数から次に大きい整数を生成	x	
FLOOR: 浮動小数点数から次に小さい整数を生成	x	
TRUNC: 値の切り捨て	x	
SCALE_X: スケール	x	
NORM_X: 正規化	x	
VARIANT_TO_DB_ANY: VARIANT の DB_ANY への変換	x	
DB_ANY_TO_VARIANT: DB_ANY の VARIANT への変換	x	
SCALE: スケール(S7-1500)	x	
UNSCALE: スケール解除(S7-1500)	x	
プログラム制御演算		
ランタイム制御		
ENDIS_PW: パスワードの適正化の制限および有効化	x	
RE_TRIGR: サイクルタイムモニタのリスタート	x	
STP: プログラム終了	x	
GET_ERROR: ローカルでエラーを取得	x	
GET_ERR_ID: ローカルでエラー ID を取得	x	
INIT_RD: すべての保持データの初期化	x	
WAIT: 遅延時間の設定	x	
ランタイム: ランタイム測定	x	
ワード論理演算		
AND: AND 論理積演算	x	

OR: OR 論理和演算	x	
XOR : EXCLUSIVE OR 排他的論理和演算	x	
INV: 1 の補数の作成	x	
DECO: デコード	x	
ENCO: エンコード	x	
SEL: 選択	x	
MUX: 多重化	x	
DEMUX: 逆多重化	x	
シフトとローテーション		
SHR: 右へシフト	x	
SHL: 左へシフト	x	
ROR: 右へローテーション	x	
ROL: 左へローテーション	x	
追加の命令		
DRUM: PLC 実行	x	
DCAT: ディスクリート制御タイマアラーム	x	
MCAT: モータ制御タイマアラーム	x	
IMC: 入力ビットとマスクのビットの比較	x	
SMC: スキャンマトリックスの比較	x	
LEAD_LAG: リード/ラグアルゴリズム	x	
SEG: 7 セグメント表示のビットパターンの作成	x	
BCDCPL: 10 の補数の作成	x	
BITSUM: セットビット数カウント	x	

CMP>T: ステップ有効化時間よりも長い

説明

「ステップ有効化時間よりも長い」命令を使用して、監視条件でステップの合計時間をモニタします。プロセス中のいかなるイベントやエラーの時間も記録されます。

プログラムされた条件で、ステップの現在または最後の起動時間(Operand1)が、定義された持続時間(Operand2)と比較されます(ミリ秒)。 比較条件が満たされている場合、この命令は論理演算の結果(RLO)「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

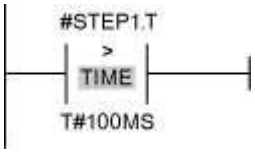
パラメータ

次の表に、「ステップ有効化時間よりも長い」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間
<Operand2>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#STEP1.T の起動時間が 100 ミリ秒未満であれば、RLO が「0」になります。 ステップ 1 の起動時間が 100 ミリ秒を超えると、直ちに RLO が「1」になります。

CMP>U: 割り込みなしステップの有効化時間よりも長い

説明

「割り込みなしステップの有効化時間よりも長い」命令を使用して、監視条件でステップの時間から障害を引いた時間をモニタします。 いかなるイベントやエラーの時間も記録されず、純粋なステップ時間のみモニタされます。

プログラムされた条件で、ステップの合計起動時間(Operand1)が、定義された持続時間(Operand2)と比較されます(ミリ秒)。 比較条件が満たされている場合、この命令は論理演算の結果(RLO)「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

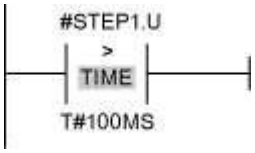
パラメータ

次の表に、「割り込みなしステップの有効化時間よりも長い」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間-障害
<Operand2>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#STEP1.U の起動時間から可能性のある障害時間を引いた時間が 100 ミリ秒未満であれば、RLO が「0」になります。ステップ 1 の起動時間が 100 ミリ秒を超えると、直ちに RLO が「1」になります。

CMP>T_MAX: 最大ステップ有効化時間よりも長い

説明

「最大ステップ有効化時間よりも長い」命令を使用して、監視条件でステップの最大合計時間をモニタします。 プロセス中のいかなるイベントやエラーの時間も記録されます。

プログラムされた条件で、ステップの現在または最後の起動時間(Operand1)が、最大持続時間(Operand2)と比較されます(ミリ秒)。 比較条件が満たされた場合、この命令は論理演算の結果(RLO)「1」を返します。 比較条件が満たされなかった場合、この命令は RLO「0」を返します。

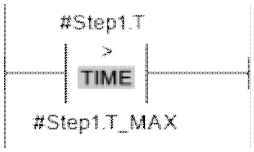
パラメータ

次の表に、「最大ステップ有効化時間よりも長い」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間
<Operand2>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#STEP1.T の起動時間が#Step1.T_MAX の最大ステップ起動時間より少ない限り、RLO は「0」になります。 最大ステップ起動時間を超えると、直ちに RLO が「1」になります。

CMP>T_WARN: 警告時間よりも長い

説明

「警告時間よりも長い」命令を使用してステップの時間をモニタし、時間が監視条件を超えると警告を出します。プロセス中のいかなるイベントやエラーの時間も記録されます。

プログラムされた条件で、ステップの現在または最後の起動時間(Operand1)が、定義された持続時間(Operand2)と比較されます(ミリ秒)。 比較条件が満たされた場合、この命令は論理演算の結果(RLO)「1」を返し、警告を出力します。比較条件が満たされなかった場合、この命令は RLO「0」を返します。

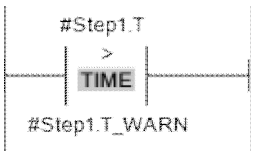
パラメータ

次の表に、「警告時間よりも長い」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
<Operand1>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間
<Operand2>	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#Step1.T の起動時間が#Step1.T_WARN の警告時間を超えない限り、RLO は「0」になります。 警告時間を超えると、直ちに RLO が「1」になります。

FBD 命令比較演算

この章には下記に関する情報が記載されています：

- 全般的 FBD 命令 (S7-1500)
- CMP>T: ステップ有効化時間よりも長い (S7-1500)
- CMP>U: 割り込みなしステップの有効化時間よりも長い (S7-1500)
- CMP>T_MAX: 最大ステップ有効化時間よりも長い (S7-1500)
- CMP>T_WARN: 警告時間よりも長い (S7-1500)

全般的 FBD 命令

説明

GRAPH でプログラミングしている場合、さまざまな FBD 命令が使用可能です。

次の表は、命令およびそれらの可用性の概要を示しています。

FBD 命令	固定の操作	移行/監視/インターロック
全般		
新規ネットワーク	x	
空ボックス	x	x
分岐: 分岐を開く	x	x
-|: 入力の挿入	x	x
-o|: RLO の反転	x	x
ビット論理演算		
&: AND 演算	x	x
>=1: OR 論理和演算	x	x
X: EXCLUSIVE OR 排他的論理和演算	x	
=: 割り当て	x	
/=: 割り当てを無効にする	x	
R: 出力のリセット	x	
S: 出力設定	x	
SET_BF: ビットフィールドのセット	x	
RESET_BF: ビットフィールドのリセット	x	
SR: フリップフロップのセット/リセット	x	
RS: フリップフロップのリセット/セット	x	
P: 立ち上がりエッジのオペランドスキャン	x	
N: 立ち下がりエッジのオペランドスキャン	x	
P=: 信号立ち上がりエッジのオペランドの設定	x	
N=: 信号立ち下がりエッジのオペランドの設定	x	
P_TRIG: 立ち上がりエッジの RLO スキャン	x	
N_TRIG: 立ち下がりエッジの RLO スキャン	x	
R_TRIG: :信号立ち上がりエッジの検出	x	
F_TRIG: :信号立ち下がりエッジの検出	x	
タイマ		
IEC タイマ		
TP: パルスの生成	x	

TON: オンディレータイマ	x	
TOF: オフディレータイマ	x	
TONR: タイムアキュムレータ	x	
TP: パルスタイマの起動	x	
TON: オンディレータイマの起動	x	
TOF: オフディレータイマの起動	x	
TONR: タイムアキュムレータ	x	
RT: タイマのリセット	x	
PT: 持続時間のロード	x	
SIMATIC タイマ		
S_PULSE: パルスタイマパラメータの割り当てと開始	x	
S_PEXT: 拡張パルスタイマパラメータを割り当てと起動	x	
S_ODT: オンディレータイマパラメータの割り当てと開始	x	
S_ODTS: 保持型オンディレータイマパラメータの割り当てと開始	x	
S_OFFDT: オフディレータイマパラメータの割り当てと開始	x	
カウンタ		
IEC カウンタ		
CTU: カウントアップ	x	
CTD: カウントダウン	x	
CTUD: カウントアップ/カウントダウン	x	
SIMATIC カウンタ		
S_CU: パラメータおよびカウントアップの割り当て	x	
S_CD: パラメータおよびカウントダウンの割り当て	x	
S_CUD: パラメータおよびカウントアップ/カウントダウンの割り当て	x	
比較演算		
CMP ==: 等価	x	x
CMP <>: 不等価	x	x
CMP >=: 以上	x	x
CMP<=: 以下	x	x
CMP >: 超過	x	x
CMP<: 未満	x	x
IN_RANGE: 範囲内の値	x	
OUT_RANGE: 範囲外の値	x	
---| OK |---: 有効性のチェック	x	
---| NOT_OK |---: 無効性のチェック	x	
EQ_Type: EQUAL (等しい)かどうかデータタイプをタグのデータタイプと比較	x	
NE_Type: UNEQUAL (等しくない)かどうかデータタイプをタグのデータタイプと比較	x	
EQ_ElemType: EQUAL (等しい)かどうか配列要素のデータタイプをタグのデータタイプと比較	x	

NE_ElemType: UNEQUAL (等しくない)かどうか配列要素のデータタイプをタグのデータタイプと比較	x	
IS_NULL: EQUALS ZERO ポインタの照会	x	
NOT_NULL: UNEQUALS ZERO ポインタの照会	x	
IS_ARRAY: 配列のチェック	x	
CMP>T: ステップ有効化時間よりも長い		x
CMP>U: 割り込みなしステップの有効化時間よりも長い		x
四則演算		
CALCULATE: 計算	x	
ADD: 加算	x	
SUB: 減算	x	
MUL: 乗算	x	
DIV: 除算	x	
MOD: 除算の剰余を返す	x	
NEG: 2 の補数の作成	x	
INC: インクリメント	x	
DEC: デクリメント	x	
ABS: 絶対値の形成	x	
MIN: 最小値の取得	x	
MAX: 最大値の取得	x	
LIMIT: 制限値の設定	x	
SQR: 2 乗値の形成	x	
SQRT: 平方根の形成	x	
LN: 自然対数の形成	x	
EXP: 指数値の形成	x	
SIN: 正弦値の形成	x	
COS: 余弦値の形成	x	
TAN: 正接値の形成	x	
ASIN: 逆正弦値の形成	x	
ACOS: 逆余弦値の形成	x	
ATAN: 逆正接値の形成	x	
FRAC: 小数部を返す	x	
EXPT: 累乗	x	
ムーブ操作		
MOVE: 移動値	x	
Deserialize: 逆シリアル化	x	
Serialize: シリアル化	x	
FieldRead: フィールドの読み出し	x	
FieldWrite: フィールドの書き込み	x	
MOVE_BLK: ブロックの移動	x	

MOVE_BLK_VARIANT: ブロックの移動	x	
UMOVE_BLK: 割り込みなしブロック転送	x	
FILL_BLK: フィル命令	x	
UFILL_BLK: 割り込み不可能なフィル命令	x	
SWAP: スワップ	x	
ReadFromArrayDB: ARRAY データブロックからの読み出し	x	
WriteToArrayDB: ARRAY データブロックへの書き込み	x	
ReadFromArrayDBL: ロードメモリの ARRAY データブロックからの読み出し	x	
WriteToArrayDBL: ロードメモリの配列データブロックへの書き込み	x	
VariantGet: VARIANT タグ値の読み出し	x	
VariantPut: VARIANT タグ値の書き込み	x	
CountOfElements: ARRAY エレメント数の取得	x	
BLKMOV: ブロックの移動(S7-1500)	x	
UBLKMOV: 割り込み不可能なブロックの移動(S7-1500)	x	
FILL: フィル命令(S7-1500)	x	
変換操作		
CONVERT: 値の変換	x	
ROUND: 数値の四捨五入	x	
CEIL: 浮動小数点数から次に大きい整数を生成	x	
FLOOR: 浮動小数点数から次に小さい整数を生成	x	
TRUNC: 値の切り捨て	x	
SCALE_X: スケール	x	
NORM_X: 正規化	x	
VARIANT_TO_DB_ANY: VARIANT の DB_ANY への変換	x	
DB_ANY_TO_VARIANT: DB_ANY の VARIANT への変換	x	
SCALE: スケール(S7-1500)	x	
UNSCALE: スケール解除(S7-1500)	x	
プログラム制御演算		
ランタイム制御		
ENDIS_PW: パスワードの適正化の制限および有効化	x	
RE_TRIGR: サイクルタイムモニタのリスタート	x	
STP: プログラム終了	x	
GET_ERROR: ローカルでエラーを取得	x	
GET_ERR_ID: ローカルでエラー ID を取得	x	
INIT_RD: すべての保持データの初期化	x	
WAIT: 遅延時間の設定	x	
ランタイム: ランタイム測定	x	
ワード論理演算		
AND: AND 論理積演算	x	

OR: OR 論理和演算	x	
XOR : EXCLUSIVE OR 排他的論理和演算	x	
INV: 1 の補数の作成	x	
DECO: デコード	x	
ENCO: エンコード	x	
SEL: 選択	x	
MUX: 多重化	x	
DEMUX: 逆多重化	x	
シフトとローテーション		
SHR: 右へシフト	x	
SHL: 左へシフト	x	
ROR: 右へローテーション	x	
ROL: 左へローテーション	x	
追加の命令		
DRUM: PLC 実行	x	
DCAT: ディスクリート制御タイマアラーム	x	
MCAT: モータ制御タイマアラーム	x	
IMC: 入力ビットとマスクのビットの比較	x	
SMC: スキャンマトリックスの比較	x	
LEAD_LAG: リード/ラグアルゴリズム	x	
SEG: 7 セグメント表示のビットパターンの作成	x	
BCDCPL: 10 の補数の作成	x	
BITSUM: セットビット数カウント	x	

CMP>T: ステップ有効化時間よりも長い

説明

「ステップ有効化時間よりも長い」命令を使用して、監視条件でステップの合計時間をモニタすることができます。 プロセス中のいかなるイベントやエラーの時間も記録されます。

プログラムされた条件で、ステップの現在または最後の起動時間(Operand1)が、定義された持続時間(Operand2)と比較されます(ミリ秒)。比較条件が満たされている場合、この命令は論理演算の結果(RLO)「1」を返します。 比較条件が満たされていない場合、この命令は RLO「0」を返します。

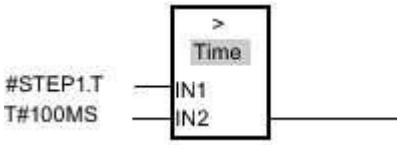
パラメータ

次の表に、「ステップ有効化時間よりも長い」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間
IN2	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#STEP1.T の起動時間が 100 ミリ秒未満であれば、RLO が「0」になります。ステップ 1 の起動時間が 100 ミリ秒を超えると、直ちに RLO が「1」になります。

CMP>U: 割り込みなしステップの有効化時間よりも長い

説明

「割り込みなしステップの有効化時間よりも長い」命令を使用して、監視条件でステップの時間から障害を引いた時間をモニタします。 いかなるイベントやエラーの時間も記録されず、純粋なステップ時間のみモニタされます。

プログラムされた条件で、ステップの合計起動時間(Operand1)が、定義された持続時間(Operand2)と比較されます(ミリ秒)。比較条件が満たされている場合、この命令は論理演算の結果(RLO)「1」を返します。比較条件が満たされていない場合、この命令は RLO「0」を返します。

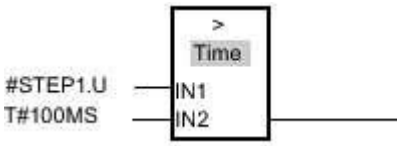
パラメータ

次の表に、「割り込みなしステップの有効化時間よりも長い」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間-障害
IN2	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#STEP1.U の起動時間から可能性のある障害時間を引いた時間が 100 ミリ秒未満であれば、RLO が「0」になります。ステップ 1 の起動時間が 100 ミリ秒を超えると、直ちに RLO が「1」になります。

CMP>T_MAX: 最大ステップ有効化時間よりも長い

説明

「最大ステップ有効化時間よりも長い」命令を使用して、監視条件でステップの最大合計時間をモニタします。プロセス中のいかなるイベントやエラーの時間も記録されます。

プログラムされた条件で、ステップの現在または最後の起動時間(Operand1)が、最大持続時間(Operand2)と比較されます(ミリ秒)。比較条件が満たされた場合、この命令は論理演算の結果(RLO)「1」を返します。比較条件が満たされなかった場合、この命令は RLO「0」を返します。

パラメータ

次の表に、「最大ステップ有効化時間よりも長い」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間
IN2	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。

#STEP1.T の起動時間が#Step1.T_MAX の最大ステップ起動時間より少ない限り、RLO は「0」になります。 最大ステップ起動時間を超えると、直ちに RLO が「1」になります。

CMP>T_WARN: 警告時間よりも長い

説明

「警告時間よりも長い」命令を使用してステップの時間をモニタし、時間が監視条件を超えると警告を出します。プロセス中のいかなるイベントやエラーの時間も記録されます。

プログラムされた条件で、ステップの現在または最後の起動時間(Operand1)が、定義された持続時間(Operand2)と比較されます(ミリ秒)。 比較条件が満たされた場合、この命令は論理演算の結果(RLO)「1」を返し、警告を出力します。比較条件が満たされなかった場合、この命令は RLO「0」を返します。

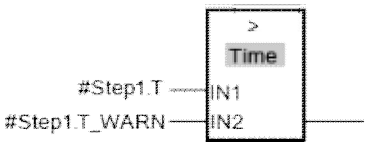
パラメータ

次の表に、「警告時間よりも長い」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	ステップの現在または最後の起動時間
IN2	Input	整数、浮動小数点数、タイマ、文字列、DATE、DT、DTL、TOD、LTOD、LDT	I、Q、M、D、L、または定数	比較に使用される時間

例

次の例は、ネットワークにおける命令を示しています。



#Step1.T の起動時間が#Step1.T_WARN の警告時間を超えない限り、RLO は「0」になります。 警告時間を超えると、直ちに RLO が「1」になります。

拡張命令

この章には下記に関する情報が記載されています：

日付と時刻

この章には下記に関する情報が記載されています：

- T_COMP: 時間タグ比較 (S7-1500)
- T_CONV: 時間の変換と抽出 (S7-1200, S7-1500)
- T_ADD: 時間加算 (S7-1200, S7-1500)
- T_SUB: 時間の減算 (S7-1200, S7-1500)
- T_DIFF: 時間差 (S7-1200, S7-1500)
- T_COMBINE: 時間結合 (S7-1200, S7-1500)
- 時刻のファンクション (S7-1200, S7-1500)

T_COMP: 時間タグ比較

説明

この命令を使用して、「タイマ」または「日付と時刻」データタイプの 2 つのタグの内容を比較します。

この命令は、以下のデータタイプの比較をサポートしています。 DATE, TIME, LTIME, TOD (TIME_OF_DAY), LTOD (LTIME_OF_DAY), DT (DATE_AND_TIME), LDT (DATE_AND_LTIME), DTL, S5Time.

比較を実行するデータタイプの長さと形式は同一であることが必要です。

比較結果は OUT パラメータに戻り値として出力されます。 このため、比較で使用する条件が満たされている場合、パラメータ OUT が「1」に設定されます。

次の比較オプションを使用できます。

シンボル	説明
EQ	IN1 および IN2 パラメータで時間が同じ場合、戻り値のシグナル状態は「1」です。
NE	IN1 および IN2 パラメータで時間が同じでない場合、戻り値のシグナル状態は「1」です。
GE	IN1 パラメータの時間が IN2 パラメータの時間よりも大きい(新しい)か、または等しい場合、戻り値のシグナル状態は「1」です。
LE	IN1 パラメータの時間が IN2 パラメータの時間よりも小さい(古い)か、または等しい場合、戻り値のシグナル状態は「1」です。
GT	IN1 パラメータの時間が IN2 パラメータの時間よりも大きい場合(新しい)、戻り値のシグナル状態は「1」です。
LT	IN1 パラメータの時間が IN2 パラメータの時間よりも小さい場合(古い)、戻り値のシグナル状態は「1」です。

パラメータ

以下の表に、「T_COMP」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL, S5Time	I、Q、M、D、L、P、または定数	比較する最初の値
IN2	Input	DATE, TIME, LTIME, TOD, LTOD, DT, LDT, DTL, S5Time	I、Q、M、D、L、P、または定数	比較する 2 番目の値
OUT	Output	BOOL	I、Q、M、D、L、P	戻り値

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

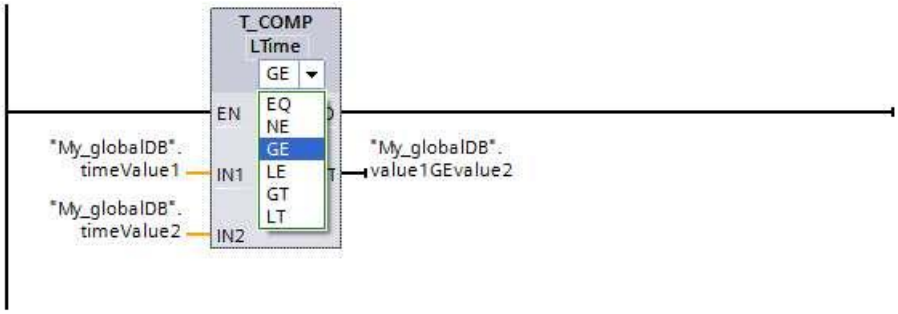
次の例では、「以上」の比較オプションを使用して、LTIME データタイプの 2 つの時刻を比較します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

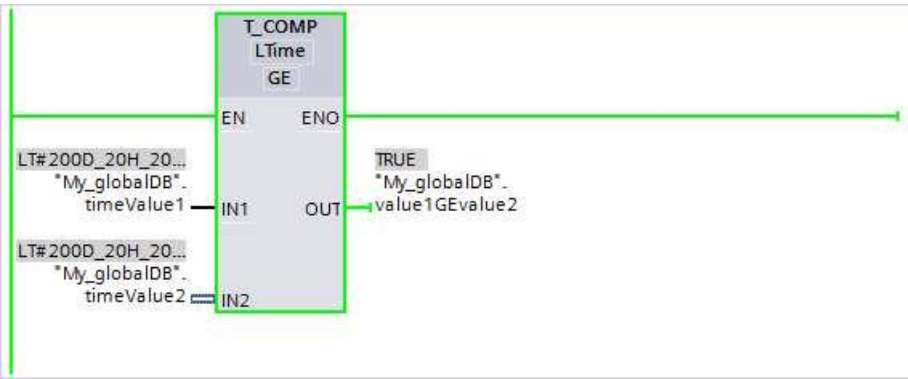
My_globalDB_T_Comp

	Name	Data type	Start value
1	Static		
2	timeValue1	LTime	LT#200d20h20...
3	timeValue2	LTime	LT#200d20h20...
4	value1GEvalue2	Bool	false

この命令のパラメータを以下のように相互接続します。 [GE]比較オプションを選択します。



比較する最初の値(「timeValue1」)の時刻が 2 番目の値(「timeValue2」)以上であるため、戻り値(「value1GEvalue2」)はシグナル状態「TRUE」を示します。



T_CONV: 時間の変換と抽出

説明

「T_CONV」命令を使用し、IN 入力パラメータのデータタイプを OUT 出力で出力されるデータタイプに変換します。 変換するデータフォーマットは、入力および出力の命令ボックスから選択します。

パラメータ

以下の表に、「T_CONV」命令のパラメータを示します。 同じデータタイプの入力および出力パラメータを使用した場合、この命令は対応する値をコピーします。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	整数、TIME、日付と時刻*	WORD、整数、タイマ、日付と時刻*	I、Q、M、D、L、P、または定数	変換する値
OUT	Return	整数、TIME、日付と時刻*	WORD、整数、タイマ、日付と時刻*	I、Q、M、D、L、P	変換の結果
* サポートされるデータタイプの範囲は CPU によって異なります。 S7-1200 および S7-1500 モジュールでサポートされているデータタイプについては、有効なデータタイプの概要を参照してください。					

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

次の例では、DATE AND TIME データタイプの時刻を LTIME OF DAY データタイプの時刻に変換します。

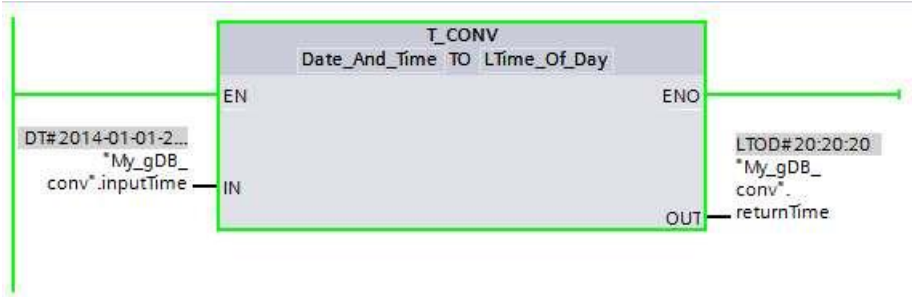
グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

My_gDB_conv

		Name	Data type	Start value
1		Static		
2		inputTime	Date_And_Time	DT#2014-01-01-20:20:20
3		returnTime	LTime_Of_Day	LTOD#00:00:00
4		<Add new>		

この命令のパラメータを以下のように相互接続し、データタイプを選択します。

変換する値(「inputTime」)は、新しい値(「returnTime」)として出力パラメータに出力されます。 日付の情報は失われます。



T_ADD: 時間加算

説明

この命令を使用して、IN1 入力の時間情報を IN2 入力の時間情報に加算します。 結果を OUT 出力パラメータで照会することができます。 次の書式を追加できます。

- 時間を別の時間に加算。

  例: TIME データタイプを他の TIME データタイプに加算。
- 時間を時刻に加算。

  例: TIME データタイプを DTL データタイプに加算。

入力パラメータ IN1 および出力パラメータ OUT の値のデータタイプは、入力および出力の命令ボックス内で選択して定義します。 時間情報は、IN2 入力パラメータ(S7-1500 では LTIME も可)で TIME 形式でのみ指定可能です。

パラメータ

次の表は、「T_ADD」命令のパラメータを変換の可能性ごとに示します。

時間を別の時間に加算

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I、Q、M、D、L、P、または定数	加算する最初の数
IN2	Input	TIME	TIME, LTIME	I、Q、M、D、L、P、または定数	加算する 2 番目の数
OUT	Return	DINT, DWORD, TIME, TOD	TIME, LTIME,	I、Q、M、D、L、P	加算の結果 選択するデータタイプは、IN1 および IN2 入力パラメータに対して選択されたデータタイプによって異なります。

時間を時刻に加算

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	DT, TOD, LTOD, LDT, DTL	I、Q、M、D、L、P、または定数	加算する最初の数 パラメータ IN2 の LTIME には、LTOD、LDT または DTL のみを使用できます。

IN2	Input	TIME	TIME, LTIME	I、Q、M、D、L、P、または定数	加算する 2 番目の数
OUT	Return	DINT, DWORD, TIME, TOD, UDINT, DTL	DT, DTL, LDT, TOD, LTOD	I、Q、M、D、L、P	加算の結果 選択するデータタイプは、IN1 および IN2 入力パラメータに対して選択されたデータタイプによって異なります。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

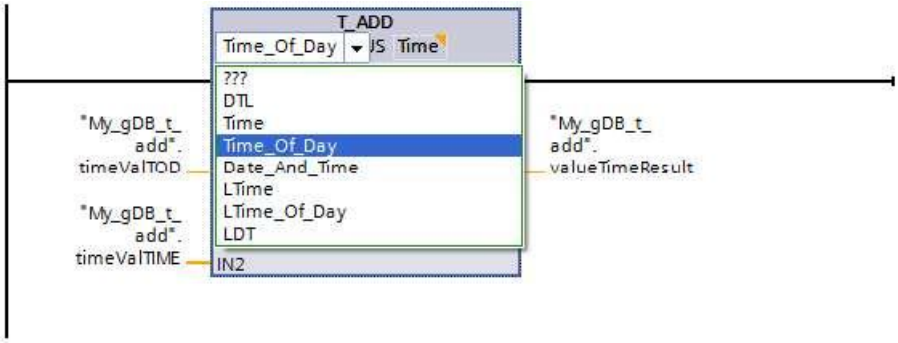
次の例では、TIME データタイプの時間を TOD データタイプの時刻に加算します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

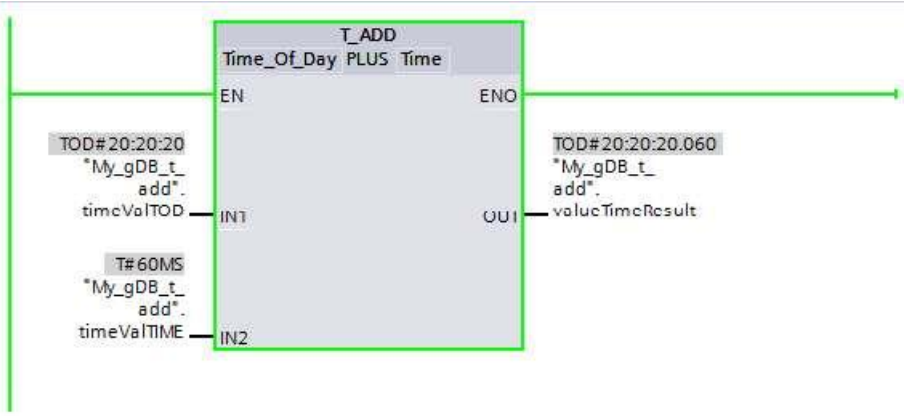
My_gDB_t_add

	Name	Data type	Start value
1	Static		
2	timeValTOD	Time_Of_Day	TOD#20:20:20
3	timeValTIME	Time	T#60ms
4	valueTimeResult	Time_Of_Day	TOD#00:00:00

この命令のパラメータを以下のように相互接続します。 時刻(「timeValTOD」)および時間(「timeValTIME」)のデータタイプを選択します。



時刻(「timeValTOD」)と時間(「timeValTIME」)は加算され、結果は出力パラメータ OUT(「valueTimeResult」)に時刻として表示されます。



T_SUB: 時間の減算

説明

この命令を使用して、IN1 入力パラメータの時間情報から IN2 入力パラメータの時間情報を減算します。 OUT 出力パラメータの差を照会できます。 以下のフォーマットを減算できます。

- 時間を別の時間から減算

  例: データタイプ TIME の時間を他のデータタイプ TIME の時間から減算。 結果は、TIME のフォーマットでタグに出力できます。

- 時刻から時間を減算

  例: データタイプ TIME の時間をデータタイプ DTL の時刻から減算。 結果は、DTL のフォーマットでタグに出力できます。

値の書式は、IN1 入力パラメータおよび OUT 出力パラメータで命令の入力および出力パラメータのデータタイプを選択して決定します。

パラメータ

次の表に、「T_SUB」命令のパラメータを変換の可能性ごとに示します。

時間を別の時間から減算

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	TIME	TIME, LTIME	I、Q、M、D、L、P、または定数	被減数
IN2	Input	TIME	TIME, LTIME	I、Q、M、D、L、P、または定数	減数
OUT	Return	DINT, DWORD, TIME, TOD, UDINT	TIME, LTIME	I、Q、M、D、L、P	減算の結果

時刻から時間を減算

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	DTL, TOD	TOD, LTOD, DTL, DT, LDT	I、Q、M、D、L、P、または定数	被減数 パラメータ IN2 の LTIME には、LTOD、LDT または DTL のみを使用できます。
IN2	Input	TIME	TIME, LTIME	I、Q、M、D、L、P、または定数	減数

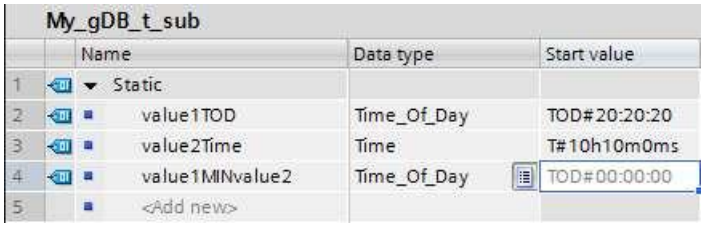
OUT	Return	DTL, DINT, DWORD, TIME, TOD, UDINT	TOD, LTOD, DTL, DT, LDT	I、Q、M、D、L、P	減算の結果

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

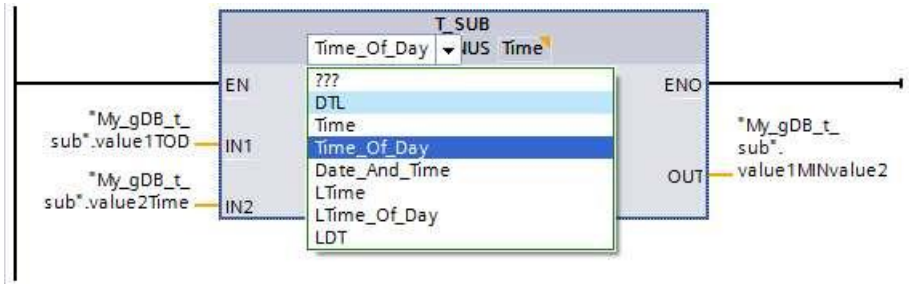
例

次の例では、TIME データタイプの時間を TOD データタイプの時刻から減算します。

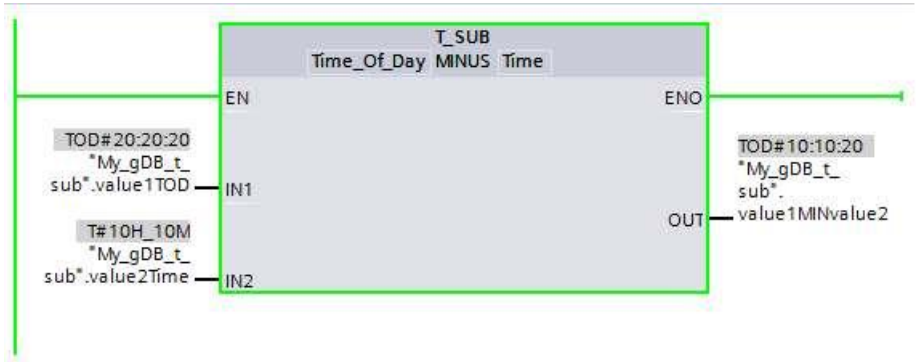
グローバルデータブロックにデータを保存するために、3 つのタグを作成します。



この命令のパラメータを以下のように相互接続します。 時刻(「value1TOD」)のデータタイプと時間(「value2Time」)のデータタイプを選択します。



時刻(「value1TOD」)と時間(「value2Time」)は減算され、結果は出力パラメータ OUT(「value1MINvalue2」)に時刻として表示されます。



T_DIFF: 時間差

説明

この命令を使用して、IN1 入力パラメータの時間情報から IN2 入力パラメータの時間情報を減算します。 結果は、出力パラメータ「OUT」に送信されます。

- IN2 入力パラメータの時間情報が IN1 入力パラメータの時間情報よりも大きい場合、結果は、OUT 出力パラメータで負の値として出力されます。
- 減算の結果が TIME の範囲外である場合、結果は「0」(0:00)にセットされ、許可出力 ENO は「0」です。

パラメータ

以下の表に、「T_DIFF」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I、Q、M、D、L、P、または定数	被減数
IN2	Input	DTL, DATE, TOD	DTL, DATE, DT, TOD, LTOD, LDT	I、Q、M、D、L、P、または定数	減数
OUT	Return	TIME, INT	TIME, LTIME, INT	I、Q、M、D、L、P	入力パラメータの差

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

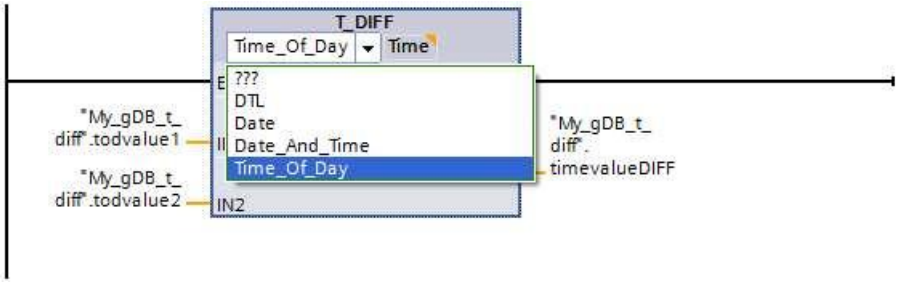
次の例では、TOD データタイプの 2 つの時刻の時間差を計算します 時間差を「TIME」データタイプで指定します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

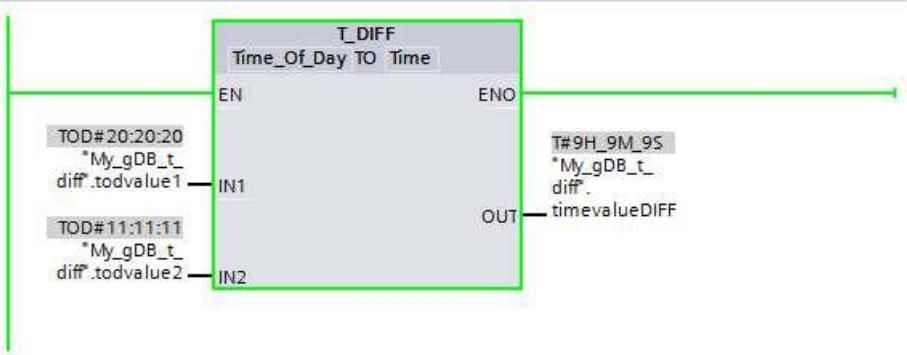
My_gDB_t_diff

	Name	Data type	Start value
1	Static		
2	todvalue1	Time_Of_Day	TOD#20:20:20
3	todvalue2	Time_Of_Day	TOD#11:11:11
4	timevalueDIFF	Time	T#0ms

この命令のパラメータを以下のように相互接続し、データタイプを選択します。 最初の選択オプションを使用して、その時点のデータタイプを指定します。 2 番目の選択オプションを使用して、時間差のデータタイプを指定します。



最初の時刻(「todvalue1」)と 2 番目の時刻(「todvalue2」)は減算され、時間差は出力パラメータ OUT(「timevalueDIFF」)に時間として表示されます。



T_COMBINE: 時間結合

説明

この命令は、日付の値と時刻の値を結合し、結合された日付と時刻の値に変換します。

- 日付は入力パラメータ IN1 に出力されます。 1990-01-01～2089-12-31 の間の値は、データタイプ DATE として使用する必要があります(これはチェックされません)。
- 時刻は IN2 入力値(TOD/LTOD データタイプ)で入力されます。
- 日付と時刻の値の結合済みデータタイプは、OUT 出力値に出力されます。

パラメータ

以下の表に、「T_COMBINE」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN1	Input	DATE	DATE	I、Q、M、D、L、P、または定数	日付の入力タグ
IN2	Input	TOD	TOD, LTOD	I、Q、M、D、L、P、または定数	時間の入力タグ
OUT	Return	DTL	DT, DTL, LDT	I、Q、M、D、L、P	日付と時刻の戻り値

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

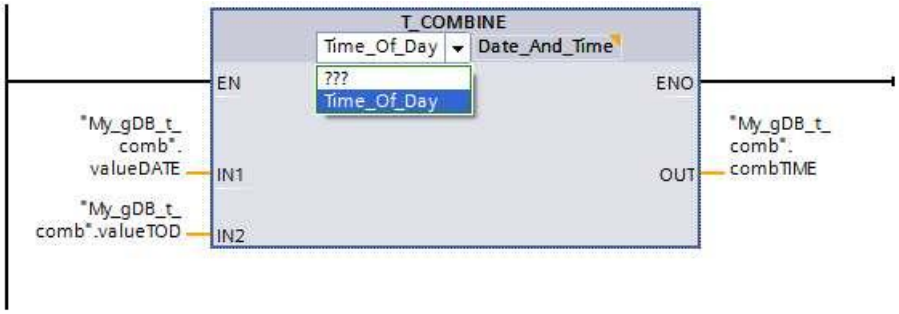
次の例では、TOD データタイプの時刻を DATE データタイプの日付に結合します。 戻り値を「DT」データタイプで指定します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

My_gDB_t_comb

	Name	Data type	Start value
1	Static		
2	valueDATE	Date	D#2014-01-01
3	valueTOD	Time_Of_Day	TOD#20:22:20
4	combTIME	Date_And_Time	DT#1990-01-01-00:00:00

この命令のパラメータを以下のように相互接続し、データタイプを選択します。 最初の選択オプションを使用して、時刻(「valueTOD」)のデータタイプを指定します。 2 番目の選択オプションを使用して、戻り値(「combTIME」)のデータタイプを指定します。



DATE(「valueDATE」)は時刻(「valueTOD」)の指定によって拡張され、戻り値は出力パラメータOUT(「combTIME」)に表示されます。

My_gDB_t_comb

	Name	Data type	Start value	Monitor value
1	Static			
2	valueDATE	Date	D#2014-01-01	D#2014-1-1
3	valueTOD	Time_Of_Day	TOD#20:22:20	TOD#20:22:20
4	combTIME	Date_And_Time	DT#1990-01-01-0	DT#2014-01-01-20:22:20

時刻のファンクション

この章には下記に関する情報が記載されています：

- WR_SYS_T: 日時の設定 (S7-1200, S7-1500)
- RD_SYS_T: 時刻の読み出し (S7-1200, S7-1500)
- RD_LOC_T: 現地時間読み出し (S7-1200, S7-1500)
- WR_LOC_T: ローカル時刻の書き込み (S7-1200, S7-1500)
- SET_TIMEZONE: タイムゾーン設定 (S7-1200, S7-1500)
- SNC_RTCB:スレーブクロックの同期 (S7-1500)
- TIME_TCK: システム時間の読み出し (S7-1500)
- RTM: ランタイムメータ (S7-1200, S7-1500)

WR_SYS_T: 日時の設定

説明

この命令を使用して、CPU の日付と時刻を設定します。 日付と時刻を入力パラメータ IN に入力します。 この値は、次の範囲内であることが必要です。

- DT の場合: 最小 DT#1990-01-01-0:0:0、最大 DT#2089-12-31-23:59:59.999
- LDT の場合:最小 LDT#1970-01-01-0:0:0.000000000、最大 LDT##2200-12-31 23:59.999
- DTL の場合:最小 DTL#1970-01-01-00:00:00.0、最大 DTL#2200-12-31 23:59.999

RET_VAL 出力パラメータで命令の実行中にエラーが発生したかどうかを照会できます。

「WR_SYS_T」命令は、ローカルタイムゾーンまたはサマータイムについての情報を渡すために使用することはできません。

パラメータ

以下の表に、「WR_SYS_T」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	DTL	DT, DTL, LDT	I、Q、M、D、L、P、または定数 *	日付と時刻
RET_VAL	Return	INT	INT	I、Q、M、D、L、P	命令のステータス
* データタイプ DT および DTL が使用できないメモリ領域: 入力、出力、およびビットメモリ。					

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

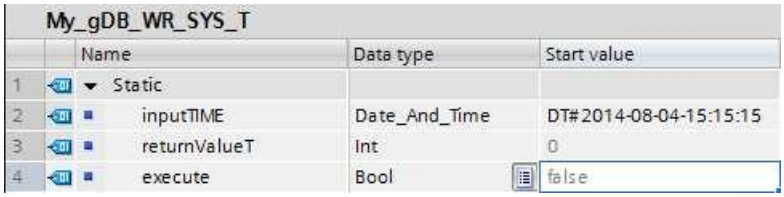
エラーコード * (W#16#....)	説明
0000	エラーは発生していません。
8080	日付のエラー
8081	時間のエラー
8082	月が無効です
8083	日が無効です
8084	時情報が無効です
8085	分情報が無効です
8086	秒情報が無効です

8087	ナノ秒情報が無効です
80B0	リアルタイムクロックが故障しました
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

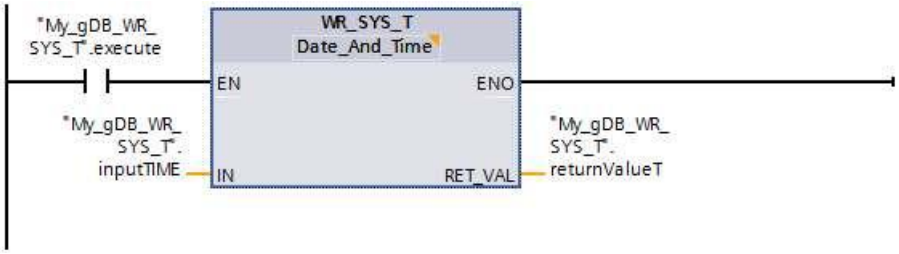
例

次の例では、CPU クロックの日付と時刻を設定します。 使用されるデータタイプは DATE AND TIME です。

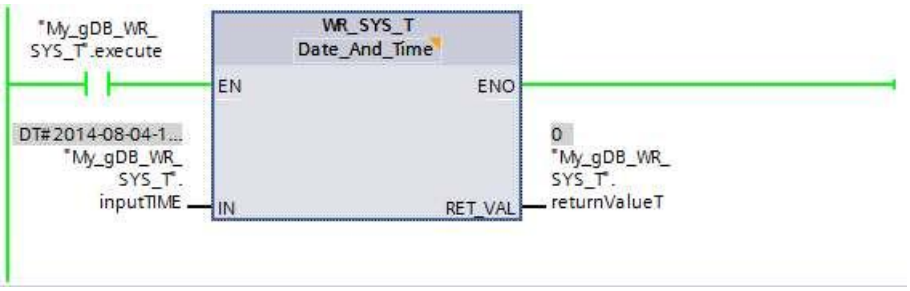
グローバルデータブロックにデータを保存するために、3 つのタグを作成します。



この命令のパラメータを以下のように相互接続します。 DATE AND TIME データタイプを選択します。



ノーマルオープン(「execute」)がシグナル状態「TRUE」を表示する場合、「WR_SYS_T」命令が実行されます。 CPU クロックのモジュール時刻は、設定される時刻(「inputTIME」)で上書きされます。 出力パラメータ RET_VAL(「returnValueT」)は、処理がエラーなしで行われたことを示します。



新しいモジュール時刻(「inputTIME」)が CPU クロックによって正しく受信されたかどうかを次のように確認することができます。

- S7-1500CPU の表示を使用: CPU 表示で、[設定|日付と時刻|全般]に進みます。
- 使用 TIA Portal: 「RD_SYS_T」命令を使用して CPU クロックのモジュール時刻を読み出します。
- 使用 TIA Portal: CPU の[オンライン&診断]エントリに進み、[ファンクション|日時の設定]タブを開きます。

  協定世界時(UTC)が CPU クロックのモジュール時刻に設定されます。 中央ヨーロッパ標準時(ローカル時刻)が TIA Portal に設定されます。 これにより、TIA Portal の[オンライン&診断]エントリで設

定される時刻(「inputTIME」)に 1 時間が加算されます。 1 時間が追加で加算されるのは、TIA Portal の設定がサマータイムに基づいているためです。 現地時間計算は、12 時間制で出力されます。

RD_SYS_T: 時刻の読み出し

説明

この命令を使用して、CPU クロックの現在の日付と現在の時刻を読み出します。

読み出された日付は、命令の OUT 出力パラメータに出力されます。 与えられた値は、ローカルタイムゾーンまたはサマータイムに関する情報を含んでいません。

RET_VAL 出力で命令の実行中にエラーが発生したかどうかを照会できます。

パラメータ

以下の表に、「RD_SYS_T」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I、Q、M、D、L、P	命令のステータス
OUT	Output	DTL	DT, DTL, LDT	I、Q、M、D、L、P	CPU の日付と時刻

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#....)	説明
0000	エラーは発生していません。
8081	OUT パラメータで指定された時間値が有効な値の範囲外にあります。 • DT の場合:最小 DT#1990-01-01-0:0:0、最大 DT#2089-12-31-23:59:59.999 • LDT の場合:最小 LDT#1970-01-01-0:0:0.000000000、最大 LDT#2262-04-11-23:47:16.854775807 • DTL の場合:最小 DTL#1970-01-01-00:00:00.0、最大 DTL#2262-04-11-23:47:16.854775807
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

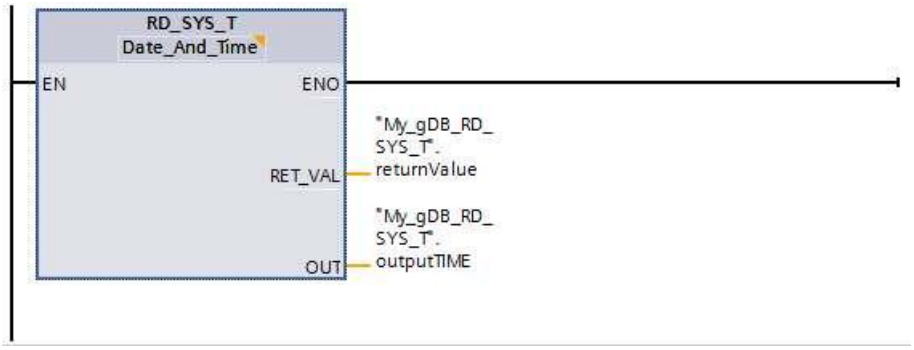
次の例では、CPU クロックのモジュール時刻を読み出します。 使用されるデータタイプは DATE AND TIME です。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

My_gDB_RD_SYS_T

	Name	Data type	Start value
1	Static		
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00
3	returnValue	Int	0

この命令のパラメータを以下のように相互接続します。 DATE AND TIME データタイプを選択します。



CPU クロックのモジュール時刻が読み出され、出力パラメータ OUT(「outputTIME」)に表示されます。出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われたことを示します。

My_gDB_RD_SYS_T

	Name	Data type	Start value	Monitor value
1	Static			
2	outputTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-15:15:15
3	returnValue	Int	0	0

RD_LOC_T: 現地時間読み出し

説明

この命令を使用して、CPU クロックから現在のローカルタイムを読み出し、これを OUT 出力します。 CPU クロックの設定でセットしたタイムゾーンおよびサマータイムの開始および標準時間の情報を使用して、ローカルタイムを出力します。

パラメータ

以下の表に、「RD_LOC_T」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
RET_VAL	Return	INT	INT	I、Q、M、D、L、P	命令のステータス
OUT	Output	DTL	DT, LDT, DTL	I、Q、M、D、L、P	ローカルタイム

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#....)	説明
0000	エラーは発生していません。
0001	エラーは発生していません。 ローカルタイムがサマータイムで出力されます。
8080	ローカルタイムを読み出せません。
8081	OUT パラメータで指定された時間値が有効な値の範囲外にあります。 • DT の場合:最小 DT#1990-01-01-0:0:0、最大 DT#2089-12-31-23:59:59.999 • LDT の場合: 最小 LDT#1970-01-01-0:0:0.000000000、最大 LDT#2262-04-11-23:47:16.854775807 • DTL の場合: 最小 DTL#1970-01-01-00:00:00.0、最大 DTL#2262-04-11-23:47:16.854775807
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

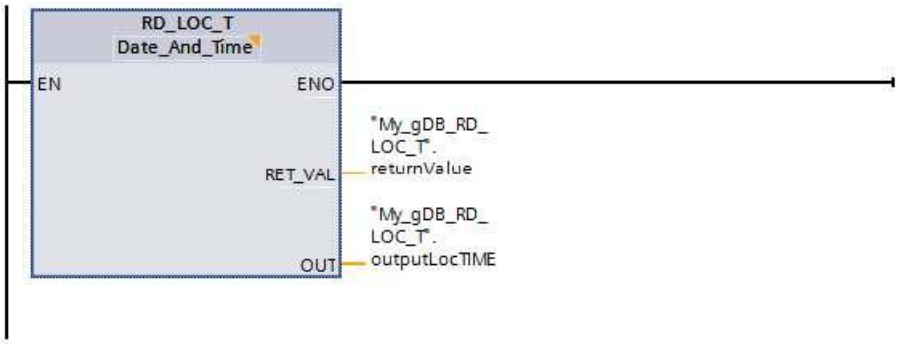
次の例では、CPU クロックのローカル時刻を読み出します。 使用されるデータタイプは DATE AND TIME です。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

My_gDB_RD_LOC_T

	Name	Data type	Start value
1	Static		
2	outputLocTIME	Date_And_Time	DT#1990-01-01-00:00:00
3	returnValue	Int	0

この命令のパラメータを以下のように相互接続します。 DATE AND TIME データタイプを選択します。



CPU クロックのローカル時刻が読み出され、出力パラメータ OUT(「outputLocTIME」)に表示されます。 出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われたことを示し、この呼び出しでローカル時刻がサマータイムとして出力されます。

My_gDB_RD_LOC_T

	Name	Data type	Start value	Monitor value
1	Static			
2	outputLocTIME	Date_And_Time	DT#1990-01-01-00:00:00	DT#2014-08-04-16:15:15.001
3	returnValue	Int	0	1

WR_LOC_T: ローカル時刻の書き込み

説明

「WR_LOC_T」命令を使用して、CPU クロックの日付と時刻を設定します。 入力パラメータ LOCTIME で、ローカルタイムとして日付と時刻を入力します。

この値は、次の範囲内であることが必要です。

- DT の場合: 最小 DT#1990-01-01-00:00:00、最大 DT#2089-12-31-23:59:59.999
- DTL の場合:最小 DTL#1970-01-01-00:00:00.0、最大 DTL#2200-12-31 23:59.999
- LDT の場合:最小 LDT#1970-01-01-0:0:0.000000000、最大 LDT#2200-12-31 23:59.999

ローカルタイムとシステム時刻の時間情報の精度は製品に依存しますが、誤差は 1 ミリ秒以下です。LOCTIME パラメータの入力値が、CPU によってサポートされている値未満の場合、システム時刻の計算時に切り上げられます。

RET_VAL 出力パラメータで命令の実行中にエラーが発生したかどうかを照会できます。

サマータイムまたは標準時間への時間切り替え時の WR_LOC_T の使用

- 標準時間からサマータイムへの切り替え

  以下では、切り替え時刻は 2:00 AM であり、時刻が 1 時間だけ進められると仮定されています。 つまり、02:00:00:000000000 AM と 02:59:59:999999999 AM の間の 1 時間は存在しません。

  LOCTIME に対して対応する時刻を指定すると、エラーコード W#16#8089 が生成されます。

  DST は無関係です。
- サマータイムから標準時間への切り替え

  以下では、切り替え時刻は 3:00 AM であり、時刻が 1 時間だけ戻されると仮定されています。 つまり、02:00:00:000000000 AM と 02:59:59:999999999 AM の間に 2 時間が存在します。

  したがって、02:00:00:000000000 AM と 02:59:59:999999999 AM の間にある LOCTIME のすべての時刻について、その時刻が時間切り替えの前か後かを宣言する必要があります。 DST パラメータはこのために使用されます。
  - DST=TRUE の場合、時刻は 2 時間の最初の 1 時間、つまり、まだサマータイムに存在します。
  - DST=FALSE の場合、時刻は 2 時間の 2 番目の 1 時間、つまり、標準時間に存在します。

  この二重時間外にある LOCTIME のすべての時刻について、DST は無関係です。

パラメータ

以下の表に、「WR_LOC_T」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
LOCTIME	Input	DTL	DT, DTL, LDT	D、L、P または定数	ローカルタイム

DST	Input	BOOL	BOOL	I、Q、M、D、L、P、T、C または定数	Daylight Saving Time 標準時間への切り替え時の「二重時間」の間にのみ評価されます。 • TRUE = サマータイム(最初の時間) • FALSE = 標準時間(2 番目の時間)
RET_VAL	Return	INT	INT	I、Q、M、D、L、P	エラーメッセージ(「RET_VAL パラメータ」を参照)

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#....)	説明
0000	エラーは発生していません。
8080	LOCTIME パラメータの値が無効です。
8081	LOCTIME パラメータで指定された時間値が有効な値の範囲外にあります。 • DT の場合:最小 DT#1990-01-01-00:00:00、最大 DT#2089-12-31-23:59:59.999 • DTL の場合:最小 DTL#1970-01-01-00:00:00.0、最大 DTL#2200-12-31 23:59.999 • LDT の場合:最小 LDT#1970-1-1-0:0:0.000000000、最大 LDT#2200-12-31 23:59.999
8082**	月(DTL フォーマットでバイト 2)に対して無効な値が指定されました。
8083**	日(DTL フォーマットでバイト 3)に対して無効な値が指定されました。
8084**	時(DTL フォーマットでバイト 5)に対して無効な値が指定されました。
8085**	分(DTL フォーマットでバイト 6)に対して無効な値が指定されました。
8086**	秒(DTL フォーマットでバイト 7)に対して無効な値が指定されました。
8087**	ナノ秒(DTL フォーマットでバイト 8～11)に対して無効な値が指定されました。
8089	時間値が存在しません(サマータイムへの変更により既に時刻を過ぎています)。
80B0	リアルタイムクロックが故障しました。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	
** DTL フォーマットの LOCTIME パラメータでのローカルタイム情報のみ。	

例

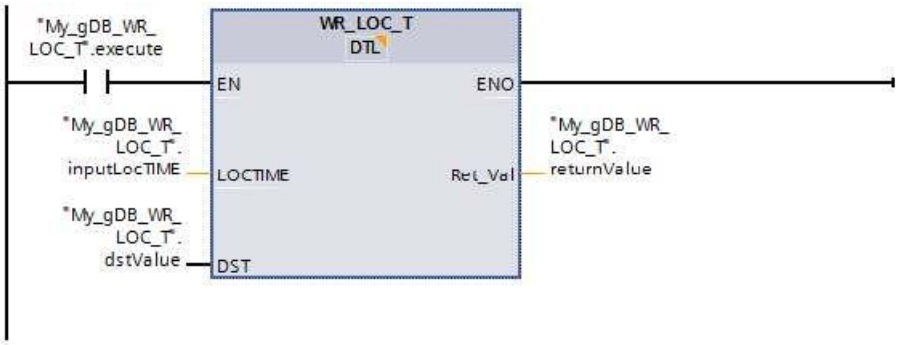
次の例では、CPU クロックのローカル時刻を設定します。 使用されるデータタイプは DTL です。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

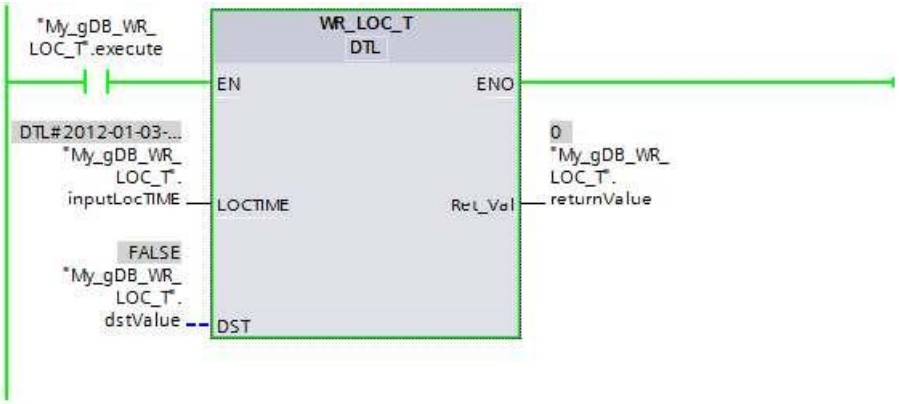
My_gDB_WR_LOC_T

	Name	Data type	Start value
1	Static		
2	inputLocTIME	DTL	DTL#2012-01-03-12:12:12
3	dstValue	Bool	false
4	returnValue	Int	0
5	execute	Bool	false

この命令のパラメータを以下のように相互接続します。 DTL データタイプを選択します。



ノーマルオープン(「execute」)がシグナル状態「TRUE」を表示する場合、「WR_LOC_T」命令が実行されます。 CPU クロックのローカル時刻は、設定される時刻(「inputLocTIME」)で上書きされます。 出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われたことを示します。 入力パラメータ DST(「dstValue」)は、時間情報に標準時間を使用することを指定します。 このパラメータは「二重時間」の場合のみ有効です。



新しいローカル時刻(「inputLocTIME」)が CPU クロックによって正しく受信されたかどうかを次のように確認することができます。

- S7-1500CPU の表示を使用: CPU 表示で、[設定|日付と時刻|全般]に進みます。
- 使用 TIA Portal: 「RD_LOC_T」命令を使用して CPU クロックのローカル時刻を読み出します。
- 使用 TIA Portal: CPU の[オンライン&診断]エントリに進み、[ファンクション|日時の設定]タブを開きます。

  ローカル時刻は、12 時間制で出力されます。

SET_TIMEZONE: タイムゾーン設定

説明

命令「SET_TIMEZONE」を使用して、ローカルタイムゾーンと、サマータイム/標準時間の切り替えのパラメータをセットします。

命令「SET_TIMEZONE」で実行される設定は、CPU のプロパティの時刻設定と対応しています。 命令「SET_TIMEZONE」の実行のために、システムデータタイプ TimeTransformationRule で該当パラメータを定義します。

ローカル時刻はシステム時刻に基づき、タイムゾーンとサマータイム/標準時間の切り替えの設定を使用して計算されます。 CPU のシステム時刻は、UTC 時刻です。 システム時刻は、システム内での通信のみに使用されます。

注記

S7-1500 シリーズの CPU での使用

「SET_TIMEZONE」命令は、ファームウェアバージョン V1.7 以降の S7-1500 シリーズの CPU のみで使用できます。

パラメータ

以下の表に、「SET_TIMEZONE」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、P、または定数	パラメータ TimeZone に保存されるほらメータの設定。
TimeZone	Input	TimeTransformationRule	D、L	パラメータ TimeZone の TimeTransformationRule システムデータタイプ(下記を参照)を相互接続します。
DONE	Output	BOOL	I、Q、M、D、L、P	ステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L、P	ステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L、P	ステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。

STATUS	Output	WORD	I、Q、M、D、L、P	詳細なエラー情報およびステータス情報は、パラメータ STATUS で出力されます。 このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

TimeZone パラメータ

ローカルタイムゾーンと、サマータイム/標準時間の切り替えのパラメータをシステムデータタイプ TimeTransformationRule で格納します。

データブロックまたはファンクションブロックのローカルインターフェースにデータタイプとして TimeTransformationRule を入力して、TimeTransformationRule を作成します。

TimeTransformationRule の構造体は、次の通りです。

名前	データタイプ	説明
TimeTransformationRule	STRUCT	
Bias	INT	ローカル時刻とシステム時刻(UTC)の間の時間差(分単位)。この値は、-720 分 ~ +780 分(-12 ~ +13 時間)の範囲である必要があります。 この値(UTC -12 ~ +13 時間)は、CPU のプロパティに指定するタイムゾーンに対応しています。
DaylightBias	INT	標準時刻とサマータイムの間の時間差(分単位)。 この値は、0 ~ 120 分の範囲である必要があります。 • 値「0」は、サマータイムと標準時間の切り替えを無効にします。 「DaylightStart...」および「StandardStart...」の値は、「0」に設定されます。 バイアスの値のみが評価されます(ローカル時刻/システム時刻の時間差)。 • 値が「0」でない場合は、TimeTransformationRule 構造体のすべてのタグが評価されます。 エントリが無効である場合、エラーコード 808F がパラメータ STATUS に出力されます。
サマータイムへの切り替えの時刻指定。以下の時刻は、常に、ローカル時刻を示します。		
DaylightStartMonth	USINT	サマータイムへの切り替えが行われる月: 1 = 1 月 2 = 2 月 3 = 3 月 ... 12 = 12 月
DaylightStartWeek	USINT	サマータイムへの切り替えが行われる週。 1 = 月の最初の週 ...

			5 = 月の最後の週
	DaylightStartWeek-day	USINT	サマータイムへの切り替えが行われる平日: 1 = 日曜日 ... 7 = 土曜日
	DaylightStartHour	USINT	サマータイムへの切り替えが行われる時
	DaylightStartMinute	USINT	サマータイムへの切り替えが行われる分
	標準時刻への切り替えの時刻指定。以下の時刻は、常に、ローカル時刻を示します。		
	StandardStartMonth	USINT	標準時刻への切り替えが行われる月: 1 = 1 月 2 = 2 月 3 = 3 月 ... 12 = 12 月
	StandardStartWeek	USINT	標準時刻への切り替えが行われる週: 1 = 月の最初の週 ... 5 = 月の最後の週
	StandardStartWeek-day	USINT	標準時刻への切り替えが行われる平日: 1 = 日曜日 ... 7 = 土曜日
	StandardStartHour	USINT	標準時刻への切り替えが行われる時
	StandardStartMinute	USINT	標準時刻への切り替えが行われる分
	TimeZoneName	STRING[80]	タイムゾーンの名前、例:「(GMT+01:00)アムステルダム、ベルリン、ベルン、ローマ、ストックホルム、ウイーン」

STATUS パラメータ

エラーコード* (W#16#....)	説明
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
808F	TimeZone パラメータの TimeTransformationRule の構造体、内容またはデータタイプが、無効であるか矛盾しています。
80C3	一時的なリソースエラー: CPU は現在、可能な最大数の同時ブロック呼び出しを処理しています。 "SET_TIMEZONE」は、少なくとも 1 つのブロック呼び出しを終了しない限り、実行できません。

* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

例

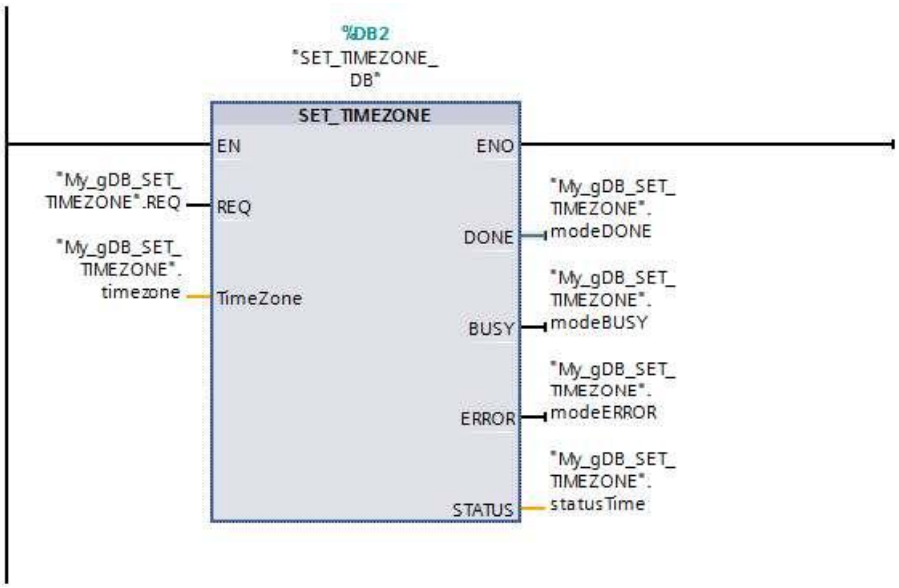
次の例では、ローカルタイムゾーンと、サマータイム/標準時間の切り替えのパラメータを設定します。

グローバルデータブロックにデータを保存するために、次を作成します。「timezone」構造体(TimeTransformationRule データタイプ)、および「REQ」、「modeDONE」、「modeBUSY」、「modeERROR」、および「statusTime」タグ。

My_gDB_SET_TIMEZONE

	Name	Data type	Start value
1	Static		
2	REQ	Bool	false
3	timezone	TimeTransformationRule	
4	Bias	Int	60
5	DaylightBias	Int	60
6	DaylightStartMonth	USInt	3
7	DaylightStartWeek	USInt	5
8	DaylightStartWeek...	USInt	1
9	DaylightStartHour	USInt	2
10	DaylightStartMinute	USInt	0
11	StandardStartMonth	USInt	10
12	StandardStartWeek	USInt	5
13	StandardStartWeek...	USInt	1
14	StandardStartHour	USInt	3
15	StandardStartMinute	USInt	0
16	TimeZoneName	String[80]	'My_GMT+'
17	modeDONE	Bool	false
18	modeBUSY	Bool	false
19	modeERROR	Bool	false
20	statusTime	Word	16#0

この命令のパラメータを以下のように相互接続します。



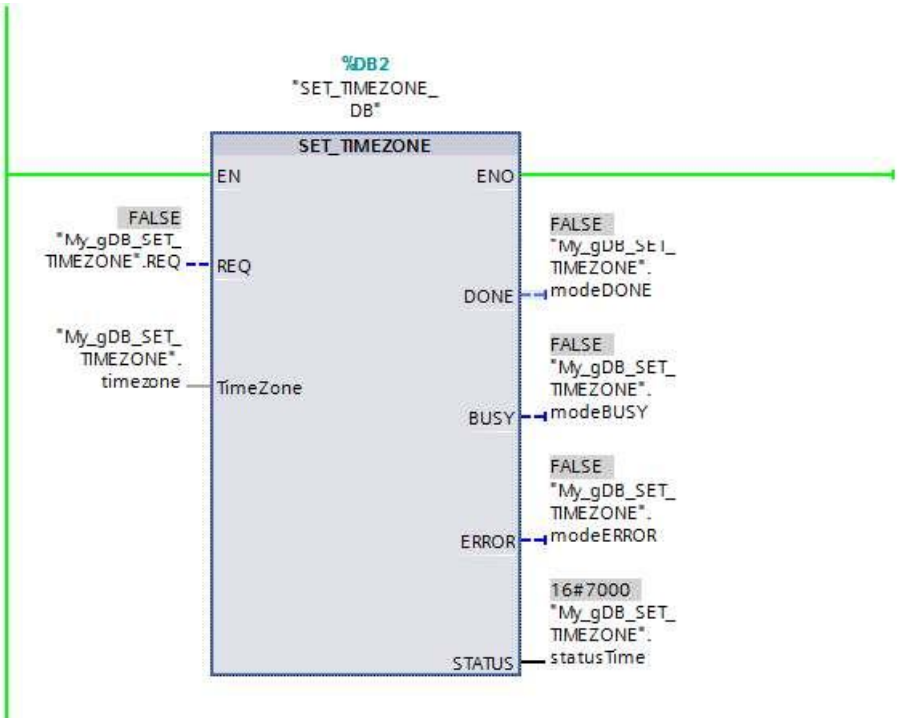
パラメータ REQ がシグナル状態「TRUE」を表示する場合、CPU クロックのタイムゾーンのデータは、設定データ(「timezone」)で上書きされます。 これはまた、次のことも意味します。

- 出力パラメータ BUSY(「modeBUSY」)は、シグナル状態「TRUE」になります。 処理後に、出力パラメータ BUSY は値「FALSE」を示し、出力パラメータ DONE(「modeDONE」)は値「TRUE」を示します。
- 出力パラメータ STATUS(「statusTime」)は、処理がどのように行われているかを示します*。 (*ジョブ処理が開始し(値「7001」)、命令は既に動作中として表示されます(値「7002」)。)
- 出力パラメータ ERROR(「modeERROR」)は、処理がエラーなしで行われていることを示します(シグナル状態は「FALSE」です)。

My_gDB_SET_TIMEZONE

	Name	Data type	Start value	Monitor value
1	Static			
2	REQ	Bool	false	TRUE
3	timezone	TimeTransformationRule		
4	Bias	Int	60	60
5	DaylightBias	Int	60	60
6	DaylightStartMonth	USInt	3	3
7	DaylightStartWeek	USInt	5	5
8	DaylightStartWeek...	USInt	1	1
9	DaylightStartHour	USInt	2	2
10	DaylightStartMinute	USInt	0	0
11	StandardStartMonth	USInt	10	10
12	StandardStartWeek	USInt	5	5
13	StandardStartWeek...	USInt	1	1
14	StandardStartHour	USInt	3	3
15	StandardStartMinute	USInt	0	0
16	TimeZoneName	String[80]	'My_GMT+'	'My_GMT+'
17	modeDONE	Bool	false	FALSE
18	modeBUSY	Bool	false	TRUE
19	modeERROR	Bool	false	FALSE
20	statusTime	Word	16#0	16#7002

注記: 「SET_TIMEZONE」命令はレベルトリガされます。 パラメータ REQ がシグナル状態「TRUE」を表示する場合のみ、命令が実行されます。



設定データ(「timezone」)が CPU クロックによって正しく受信されたかどうかを次のように確認することができます。

- S7-1500CPU の表示を使用: CPU 表示で、[設定|日付と時刻|サマータイム]に進みます。
- 使用 TIA Portal: 「RD_LOC_T」命令を使用して CPU クロックのローカル時刻を読み出します。
- 使用 TIA Portal: 「RD_SYS_T」命令を使用して CPU クロックのモジュール時刻を読み出します。

SNC_RTCB:スレーブクロックの同期

定義: スレーブクロックの同期

クロックスレーブの同期は、バスセグメントのクロックマスタからこのバスセグメントのすべてのクロックスレーブへ転送される日付と時刻を参照します。

説明

この命令を使用して、割り当てられた同期間隔に関係なく、バスセグメント上に存在するすべてのスレーブクロックを同期します。 同期は、リアルタイムクロックが少なくとも 1 つのバスセグメントでマスタクロックとして割り当てられた CPU で「SNC_RTCB」が呼び出された場合のみ正常に行われます。

パラメータ

次の表に、「SNC_RTCB」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Output	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	同期中にエラーが発生しませんでした。
0001	既存のクロックに、いずれのバスセグメントに対するマスタクロックファンクションも割り当てられませんでした。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

TIME_TCK: システム時間の読み出し

説明

「TIME_TCK」命令を使用して、CPU のシステム時刻を読み出します。 システム時刻は 0 から最大 2147483647 ミリ秒をカウントする時刻カウンタです。オーバーフローが発生した場合、システム時刻は「0」から開始して再度カウントされます。 時間スケールとシステム時刻の精度は 1 ミリ秒です。システム時刻は、CPU の動作モードのみに影響されます。 システム時刻を使用して、たとえば 2 つの「TIME_TCK」呼び出しの結果を比較して、プロセスの時間を計測することができます。 この命令は、エラー情報を提供しません。

次の表は、CPU の動作モードによってシステム時間がどのように変わるかの概要を示します。

モード	システム時刻 ...
起動	... 常に更新されます。
実行	
停止	... 停止され、現在地を保持
ウォームリスタート	... 削除され、「0」で再起動

パラメータ

以下の表に、「TIME_TCK」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Return	TIME	I、Q、M、D、L	RET_VAL パラメータには、0 ～2^{31} -1 ミリ秒の範囲の読み出されたシステム時間が含まれます。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

次の例では、CPU のシステム時刻を読み出します。 戻り値を「TIME」データタイプで指定します。

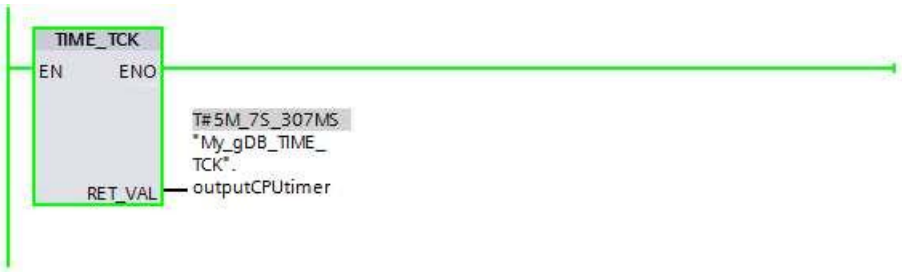
グローバルデータブロックにデータを保存するために、1 つのタグを作成します。

My_gDB_TIME_TCK

	Name	Data type	Start value
1	Static		
2	outputCPUtimer	Time	T#0ms

この命令のパラメータを以下のように相互接続します。

CPU のシステム時刻が読み出され、出力パラメータ RET_VAL(「outputCPUtimer」)に表示されます。



RTM: ランタイムメータ

説明

この命令を使用して、使用している CPU の 32 ビットの動作時間カウンタを設定、開始、停止、および読み出します。

保存された値が不正となる可能性があるので、ユーザープログラムの実行中にも必ず動作時間カウンタの開始および停止が行えるようにしてください。

パラメータ

以下の表に、「RTM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
NR	Input	RTM	I、Q、M、D、L、または定数	動作時間カウンタの番号 番号が 0 から始まる 使用している CPU の動作時間カウンタの番号についての情報は、技術データを参照してください。
MODE	Input	BYTE	I、Q、M、D、L、または定数	ジョブ ID: • 0: 読み出し(次にステータスが CQ に、現在値が CV に書き込まれます)。 動作時間カウンタが(2E31) -1 時間に達すると、表示可能の最大値で停止し、「オーバーフロー」エラーメッセージが出力されます。 • 1: 開始(最後のカウンタ値で) • 2: 停止 • 4: 設定(PV で指定された値に) • 5: 設定(PV で指定された値に)して開始 • 6: 設定(PV で指定された値に)して停止
PV	Input	DINT	I、Q、M、D、L、または定数	動作時間カウンタの新しい値
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
CQ	Output	BOOL	I、Q、M、D、L	動作時間カウンタのステータス(1: 実行中)
CV	Output	DINT	I、Q、M、D、L	動作時間カウンタの現在値

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
8080	動作時間カウンタの間違った番号
8081	PV パラメータに負の値が渡されました。
8082	動作時間カウンタのオーバーフロー
8091	MODE 入力パラメータに無効な値が含まれています。
一般エラー情報	関連項目: GET_ERR_ID を使用したエラー評価

例

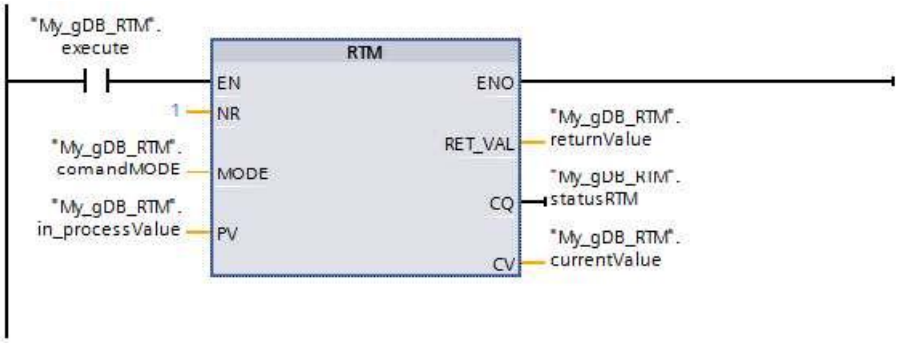
次の例では、CPU の動作時間カウンタを設定し、1 時間後に値を読み出します。

グローバルデータブロックにデータを保存するために、6 つのタグを作成します。

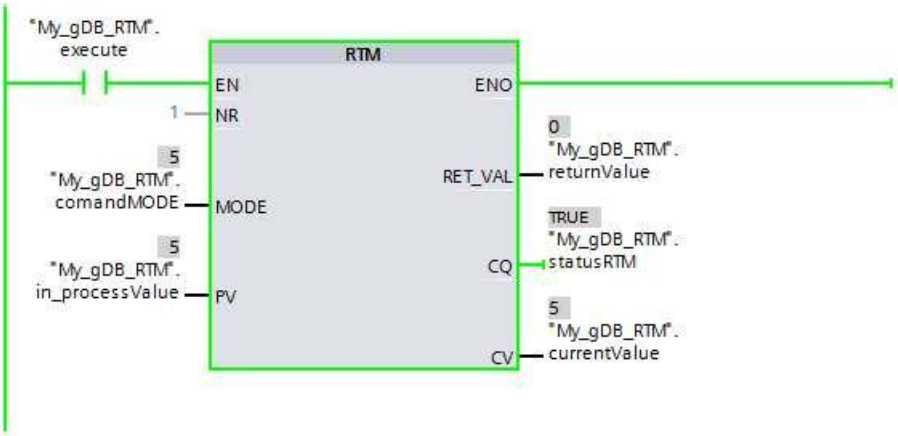
My_gDB_RTM

	Name	Data type	Start value
1	Static		
2	execute	Bool	false
3	in_processValue	DInt	5
4	returnValue	Int	0
5	statusRTM	Bool	false
6	currentValue	DInt	0
7	comandMODE	Byte	5

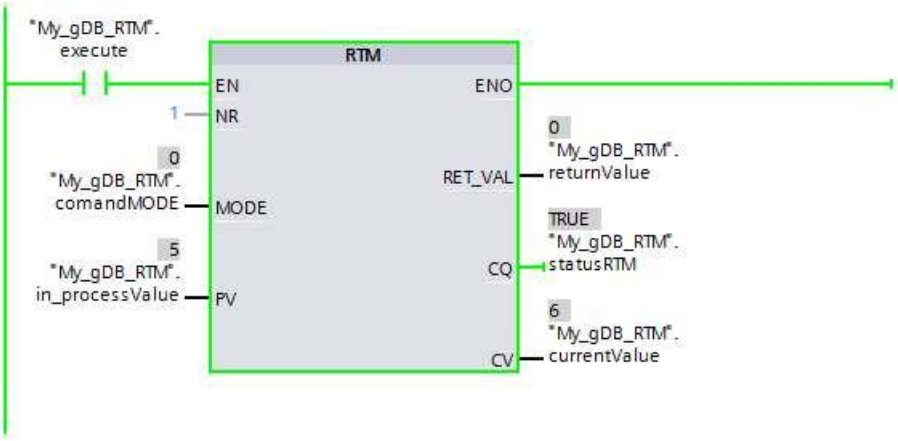
この命令のパラメータを以下のように相互接続します。 入力パラメータ NR で、CPU の動作時間カウンタの番号を指定します。



ノーマルオープン(「execute」)がシグナル状態「TRUE」を表示する場合、「RTM」命令が実行されます。 CPU の動作時間カウンタは、設定値(「in_processValue」)に設定され、開始されます。 動作時間カウンタの開始後、入力パラメータ MODE(「comandMODE」)の値を「0」に設定します。 (これを行うには、パラメータをクリックし、[オペランドの変更| 0 に設定]を選択します。) その結果、「RTM」命令は動作時間カウンタの現在値(「currentValue」)を読み出すだけで、値は変更されません。 出力パラメータ CQ(「statusRTM」)は、動作時間カウンタの開始後に動作時間カウンタが実行中であること(値は「TRUE」)を示します。 出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われていることを示します。



1 時間後、出力パラメータ CV(「currentValue」)は値「6」を示します。



String + Char

この章には下記に関する情報が記載されています：

S_MOVE: 文字列移動

説明

この命令を使用して、パラメータ IN の文字列(W)STRING の内容をパラメータ OUT で指定したデータ領域に書き込むことができます。

「MOVE_BLK」および「UMOVE_BLK」命令を使用して、データタイプ ARRAY のタグをコピーすることができます。

パラメータ

以下の表に、「S_MOVE」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	ソース文字列
OUT	Output	STRING, WSTRING	D、L	宛先文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

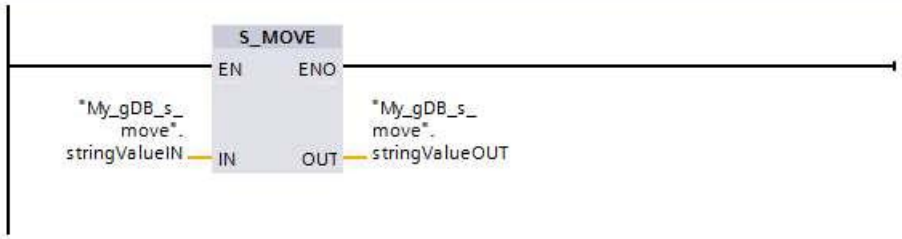
次の例では、入力パラメータ IN の文字列の内容を出力パラメータ OUT の別の文字列にコピーします。 使用されるデータタイプは STRING です。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

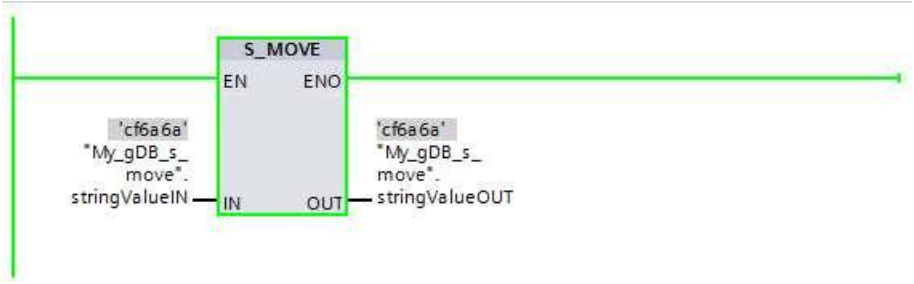
My_gDB_s_move

	Name	Data type	Start value
1	Static		
2	stringValueIN	String	'cf6a6a'
3	stringValueOUT	String	''

この命令のパラメータを以下のように相互接続します。



コピーする文字列の結果(「stringValueIN」)は、出力パラメータ OUT(「stringValueOUT」)に出力されます。



S_COMP: 文字列比較

説明

この命令は、(W)STRING フォーマットの 2 つのタグの内容を比較し、この比較結果を戻り値として出力します。 比較するタグは、IN1 および IN2 入力で相互接続されます。 入力パラメータには、シンボリックで定義されたタグのみを割り当てることができます。

命令ボックスを使用して、比較条件を選択します。 比較条件(たとえば、以上)が満たされると、出力パラメータ OUT でシグナル状態が「1」にセットされます。

次の比較オプションを使用できます。

シンボル	説明
EQ	IN1 パラメータの文字列が IN2 パラメータの文字列と同じ場合、戻り値のシグナル状態は「1」です。
NE	IN1 パラメータの文字列が IN2 パラメータの文字列と等しくない場合、戻り値のシグナル状態は「1」です。
GT (1)	IN1 パラメータの文字列が IN2 パラメータの文字列よりも大きい場合、戻り値のシグナル状態は「1」です。
LT (1)	IN1 パラメータの文字列が IN2 パラメータの文字列を超えている場合、戻り値のシグナル状態は「1」です。
GE (1)	IN1 パラメータの文字列が IN2 パラメータの文字列以上の場合、戻り値のシグナル状態は「1」です。
LE (1)	IN1 パラメータの文字列が IN2 パラメータの文字列以下の場合、戻り値のシグナル状態は「1」です。
(1)文字列を比較する際には、個々の文字はその ASCII コードによって左から比較されます(たとえば、「a」は「A」よりも大きい)。 最初に異なる文字が、比較の結果を決定します。 最初の文字が同じ場合、より長い文字列が大きくなります。	

パラメータ

以下の表に、「S_COMP」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	STRING, WSTRING*	D、L、または定数	STRING / WSTRING フォーマットの入力タグ。
IN2	Input	STRING, WSTRING*	D、L、または定数	STRING / WSTRING フォーマットの入力タグ。
OUT	Output	BOOL	I、Q、M、D、L	比較の結果
* 一時タグのインターフェース宣言でデータタイプ STRING / WSTRING を使用する場合、文字列の最大長を定義します(詳細については、データタイプの説明を参照)。				

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

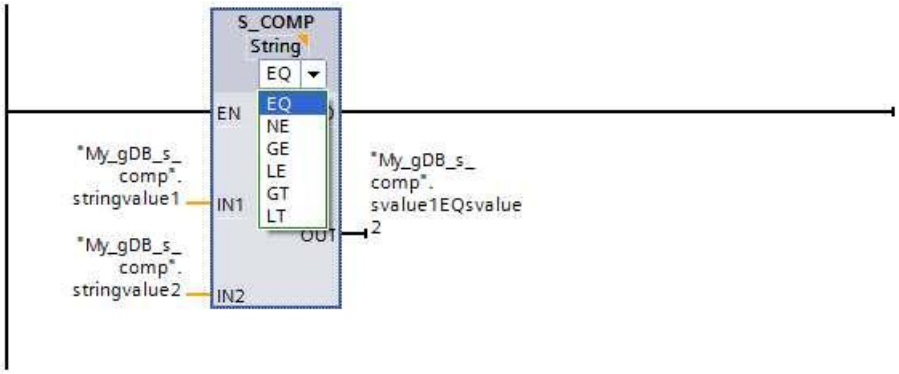
次の例では、「等しい」の比較オプションを使用して、STRING データタイプの 2 つの文字列を比較します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

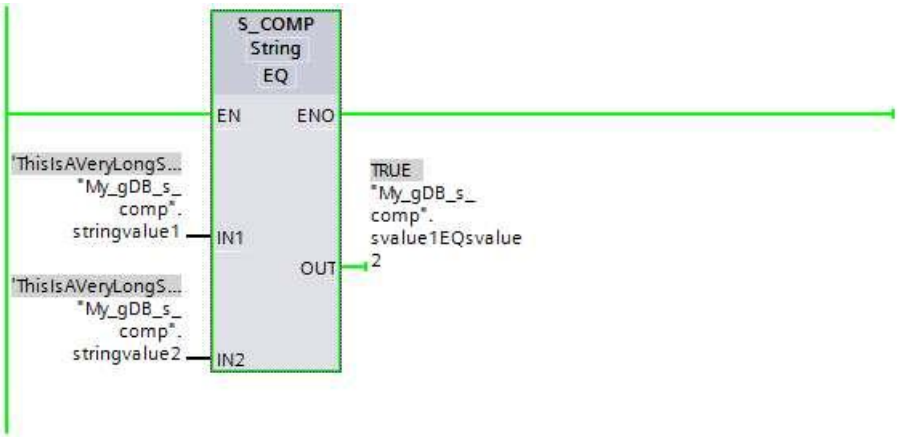
My_gDB_s_comp

	Name	Data type	Start value
1	Static		
2	stringvalue1	String	'ThisIsAVeryLongStringWithSomeNumbers646'
3	stringvalue2	String	'ThisIsAVeryLongStringWithSomeNumbers646'
4	svalue1EQsvalue2	Bool	false

この命令のパラメータを以下のように相互接続します。 STRING データタイプと EQ 比較オプションを選択します。



比較する最初の文字列(「stringvalue1」)の値が 2 番目の文字列値(「stringvalue2」)と等しいため、比較結果(「svalue1EQsvalue2」)はシグナル状態「TRUE」を示します。



S_CONV: 文字列変換

説明

この命令を使用して、IN 入力の値を OUT 出力で指定したデータフォーマットに変換します。 変換の出力フォーマットは、OUT 出力パラメータのデータタイプを選択して決定します。

以下の変換が可能です。

- 文字列を以下に変換:
  - 数値(整数または浮動小数点数)

    変換は、IN 入力パラメータで指定された文字列のすべての文字に対して行われます。 許可されている文字は、「0」～「9」の数字、小数点および正および負の符号です。 文字列の最初の文字は、有効な数または符号になります。 先頭のスペースおよび指数表記は無視されます。
  - 文字

    文字列が文字に変換されるときは、文字列の最初の文字がパラメータ OUT に転送されます。
  - 文字列
- 数値または文字を文字列へ変換:
  - 変換する数値のフォーマットは、IN 入力のデータタイプを選択して決定します。 (W)STRING データタイプの有効なタグを OUT 出力で指定する必要があります。 変換後の文字列の長さは、IN 入力の値によって異なります。
  - 変換結果は 3 番目のバイトから始まる文字列として保存されます。 文字列の最初のバイトは最大長を記録し、2 番目のバイトは文字列の実際の長さを記録します。 正の数値は符号なしで出力されます。
  - 数値 0(INT または UINT データタイプとして存在)が文字列に変換される場合(INT_TO_STRING(0)など)は、変換された文字列は 6 文字の長さになります。
  - 数値が文字列に変換される場合、文字列の最初の文字は空白文字が入力されます。 空白の数は数値の長さに応じて異なります。
  - (W)CHAR 文字が変換されるときは、この文字が文字列の最初の位置に書き込まれます。

注記

浮動小数点数からの変換中の指数表記

「S_CONV」命令のある浮動小数点数からの変換に、指数表現(「e」または「E」)を使用しないでください。 この場合、「STRG_VAL」を使用して指数表記の浮動小数点数を変換します。 この命令の FORMAT パラメータを使用して、入力書式として指数表現を選択することができます。

- 文字から文字への変換

パラメータ

次の表に、「S_CONV」命令のパラメータを変換の可能性ごとに示します。

文字列を数値に変換するためのパラメータ:

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	変換する値
OUT	Output	CHAR, WCHAR,	I、Q、M、D、L	変換の結果

		USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL		

文字列の文字列への変換のパラメータ:

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	変換する値
OUT	Output	STRING, WSTRING	D、L	変換の結果(可能な変換: STRING から WSTRING、およびその逆)

数値または文字を文字列に変換するためのパラメータ:

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	CHAR, WCHAR, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	I、Q、M、D、L、または定数	変換する値
OUT	Output	STRING, WSTRING	D、L	変換の結果

文字を文字に変換するためのパラメータ:

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	CHAR, WCHAR	I、Q、M、D、L、または定数	変換する値
OUT	Output	CHAR, WCHAR	I、Q、M、D、L	変換の結果(可能な変換: CHAR から WCHAR、およびその逆)

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

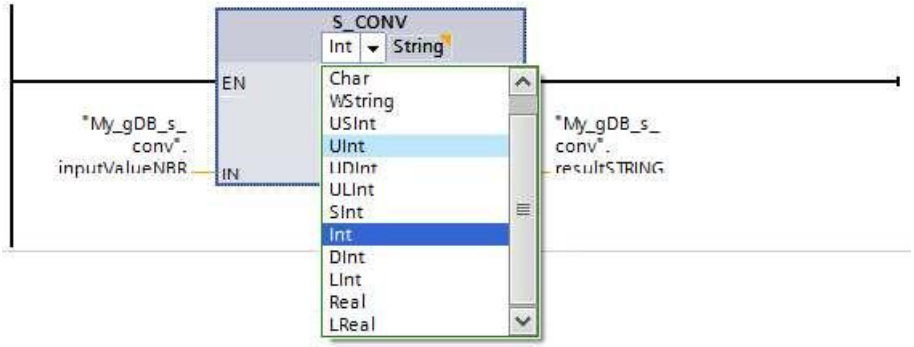
次の例では、INT データタイプの数字を STRING データタイプの文字列に変換します。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

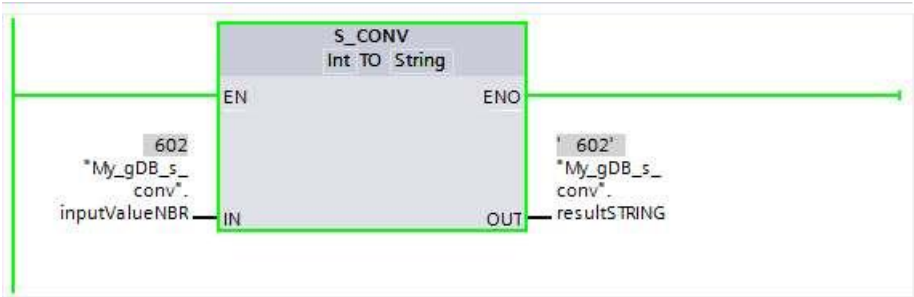
My_gDB_s_conv

	Name	Data type	Start value
1	Static		
2	inputValueNBR	Int	602
3	resultSTRING	String	''
4	<Add new>		

この命令のパラメータを以下のように相互接続します。 このために、データタイプを選択します。 最初の選択オプションを使用して、変換する値(「inputValueNBR」)のデータタイプを指定します。 2 番目の選択オプションを使用して、生成する文字列(「resultSTRING」)のデータタイプを指定します。



変換する値(「inputValueNBR」)は出力形式に変換されます。 文字列の最初の空きスペースに空白文字が書き込まれます。 変換の結果は、出力パラメータ OUT(「resultSTRING」)に文字列として出力されます。



STRG_VAL: 文字列を数値に変換

説明

「STRG_VAL」命令は、文字列を整数または浮動小数点数に変換します。

- IN 入力パラメータで変換する文字列を指定します。
- 出力値の書式は、OUT 出力パラメータのデータタイプを指定して定義します。

変換が許可されている文字は、「0」～「9」の数字、小数点、小数点のカンマ、「E」および「e」の表記、および正および負の符号です。 無効な文字が存在すると、文字列変換が中断される場合があります。

「STRG_VAL」命令は、SCL プログラミング言語によってサポートされていません。

パラメータ

以下の表に、「STRG_VAL」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	STRING, WSTRING	STRING, WSTRING	D、L、または定数	変換される数字列
FORMAT	Input	WORD	WORD	I、Q、M、D、L、P、または定数	文字の出力書式
P	Input	UINT	UINT	I、Q、M、D、L、P、または定数	変換する最初の文字への参照(最初の文字= 1、値「0」または値>文字列長は無効)
OUT	Output	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I、Q、M、D、L、P	変換の結果

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ FORMAT

FORMAT パラメータを使用し、文字列の文字の変換方法を指定します。 「STRG_VAL」命令を使用して、指数値も変換および表示できます。

次の表に、FORMAT パラメータの可能な値とその意味を示します。

値 (W#16#....)	表記法	小数点の表現
0000	小数	"."

0001		","
0002	指数	"."
0003		","
0004 終了 FFFF	無効な値	

パラメータ P

変換は、P パラメータで指定した位置の文字から開始されます。 たとえば、P パラメータで値「1」が指定されている場合、変換は指定された文字列の最初の文字から開始されます。

例

次の表に、文字列から数値への変換の例を示します。

IN (STRING)	FORMAT (W#16#....)	OUT (データタイプ)	OUT (値)	ENO ステータス
'123'	0000	INT/DINT	123	1
'-00456'	0000	INT/DINT	-456	1
'123.45'	0000	INT/DINT	123	1
'+2345'	0000	INT/DINT	2345	1
'00123AB'	0000	INT/DINT	123	1
'123'	0000	REAL	123.0	1
'-00456'	0001	REAL	-456.0	1
'+00456'	0001	REAL	456.0	1
'123.45'	0000	REAL	123.45	1
'123.45'	0001	REAL	12345.0	1
'123,45'	0000	REAL	12345.0	1
'123,45'	0001	REAL	123.45	1
'.00123AB'	0001	REAL	123.0	1
'1.23e-4'	0000	REAL	1.23	1
'1.23E-4'	0000	REAL	1.23	1
'1.23E-4'	0002	REAL	1.23E-4	1
'12,345.67'	0000	REAL	12345.67	1
'12,345.67'	0001	REAL	12.345	1
'3.4e39'	0002	REAL	W#16#7F800000	1
'-3.4e39'	0002	REAL	W#16#FF800000	1
'1.1754943e-38'	0002	REAL	0.0	1
'12345'	-/-	SINT	0	0
'A123'	-/-	-/-	0	0
''	-/-	-/-	0	0
'++123'	-/-	-/-	0	0
'+-123'	-/-	-/-	0	0

例

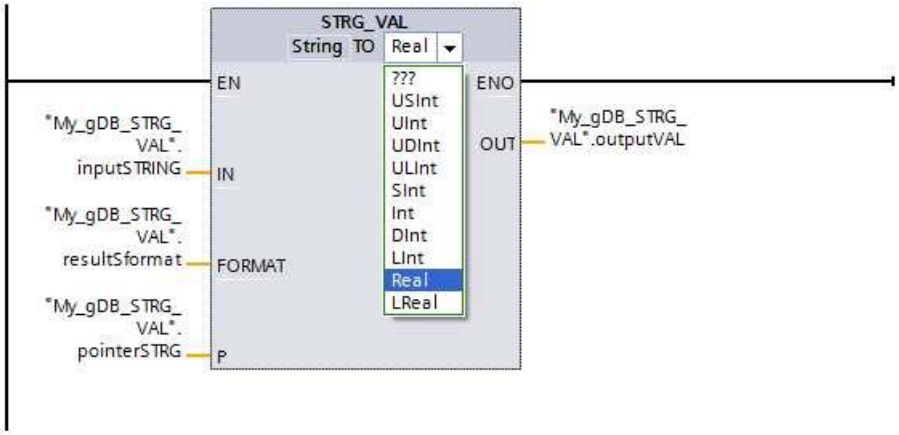
次の例では、STRING データタイプの数字列を REAL データタイプの浮動小数点数に変換します。 変換された結果は、REAL データタイプのため 32 ビットの長さで符号が付きます。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

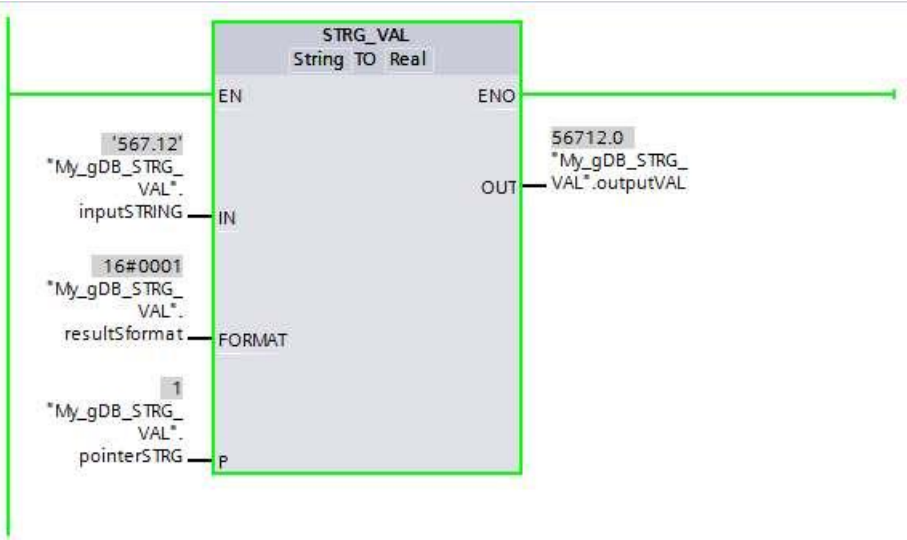
My_gDB_STRG_VAL

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'567.12'
3	resultSformat	Word	16#0001
4	pointerSTRG	UInt	1
5	outputVAL	Real	0.0

この命令のパラメータを以下のように相互接続します。 左の選択オプションを使用して、文字列のデータタイプを選択します。 右の選択オプションを使用して、浮動小数点数のデータタイプを選択します。



パラメータ P(「pointerSTRG」)の値「1」に従って、数字列は最初の文字から変換されます。 パラメータ FORMAT(「resultSformat」)の値「0001」により、数字列のピリオドは千桁の区切り文字と解釈されます。 (カンマは値「0001」では小数点の記号です。) 変換する値(「inputSTRING」)は、浮動小数点数として出力パラメータ OUT(「outputVAL」)に出力されます。



VAL_STRG: 数値を文字列に変換

説明

「VAL_STRG」命令を使用し、数値を文字列に変換します。

- 変換する値は、IN 入力パラメータで指定します。 数値のフォーマットは、データタイプを選択して決定します。
- 変換の結果は、OUT 出力パラメータで照会します。

変換が許可されている文字は、「0」～「9」の数字、小数点、小数点のカンマ、「E」および「e」の表記、および正および負の符号です。 無効な文字が存在すると、文字列変換が中断される場合があります。

「VAL_STRG」命令は、SCL プログラミング言語によってサポートされていません。

パラメータ

以下の表に、「VAL_STRG」命令のパラメータを示します。

パラメータ	宣言	データタイプ		メモリ領域	説明
		S7-1200	S7-1500		
IN	Input	USINT, SINT, UINT, INT, UDINT, DINT, REAL, LREAL	USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, LREAL	I、Q、M、D、L、P、または定数	変換する値
SIZE	Input	USINT	USINT	I、Q、M、D、L、P、または定数	文字位置の数
PREC	Input	USINT	USINT	I、Q、M、D、L、P、または定数	小数点以下の桁数
FORMAT	Input	WORD	WORD	I、Q、M、D、L、P、または定数	文字の出力書式
P	InOut	UINT	UINT	I、Q、M、D、L、P、または定数	結果の書き込みが開始する文字
OUT	Output	STRING, WSTRING	STRING, WSTRING	D、L	変換の結果

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ P

P パラメータを使用し、結果の書き込みが開始される文字列の文字を指定します。 たとえば P パラメータで値「2」が指定されている場合、変換された値は文字列の 2 番目の文字から保存されます。

SIZE および P パラメータ

SIZE パラメータを使用し、文字列の何文字が書き込まれるか指定します。 これは、P パラメータで指定された文字からカウントされます。 出力値が指定された長さよりも短い場合、結果は文字列に右揃えで書き込まれます。 空の文字位置は空白で埋められます。

パラメータ FORMAT

FORMAT パラメータを使用し、変換中に数値がどのように変換され、文字列に書き込まれるか指定します。 USINT パラメータでは、FORMAT データタイプのタグのみを指定することができます。

次の表に、FORMAT パラメータの可能な値とその意味を示します。

値 (W#16#....)	表記法	符号	小数点の表現
0000	小数	"-"	"."
0001			","
0002	指数		"."
0003			","
0004	小数	「+」と「-」	"."
0005			","
0006	指数		"."
0007			","
0008 終了 FFFF	無効な値		

パラメータ PREC

PREC パラメータを使用し、浮動小数点数の変換時の小数位を定義します。 REAL データタイプの数値では、7 つの数値の最大精度がサポートされています。 変換する値が整数の場合、PREC パラメータを使用して小数点を配置する位置を指定します。

例

次の表に、数値から文字列への変換の例を示します。

IN(値)	IN (データタイプ)	P	SIZE	FORMAT (W#16#....)	PREC	OUT (STRING)	ENO ステータス
123	UINT	16	10	0000	0	xxxxxxx123 C	1
0	UINT	16	10	0000	2	xxxxxx0.00 C	1
12345678	UDINT	16	10	0000	3	x12345.678 C	1
12345678	UDINT	16	10	0001	3	x12345.678 C	1
123	INT	16	10	0004	0	xxxxxx+123 C	1
-123	INT	16	10	0004	0	xxxxxx-123 C	1

-0.00123	REAL	16	10	0004	4	xxx-0.0012 C	1
-0.00123	REAL	16	10	0006	4	-1.2300E-3 C	1
-Inf [1]	REAL	16	10	-/-	4	xxxxxx-INF C	0
+Inf [2]	REAL	16	10	-/-	4	xxxxxx+INF C	0
NaN [3]	REAL	16	10	-/-	4	xxxxxxxNaN C	0
12345678	UDINT	16	6	-/-	3	xxxxxxxxxx C	0

「x」は空白を示します
[1]-Inf: 負の無限数を表す浮動小数点数。
[2]+Inf: 正の無限数を表す浮動小数点数。
[3]NaN: 無効な数式演算の結果として返された値。

例

次の例では、REAL データタイプの浮動小数点数を STRING データタイプの文字列に変換します。

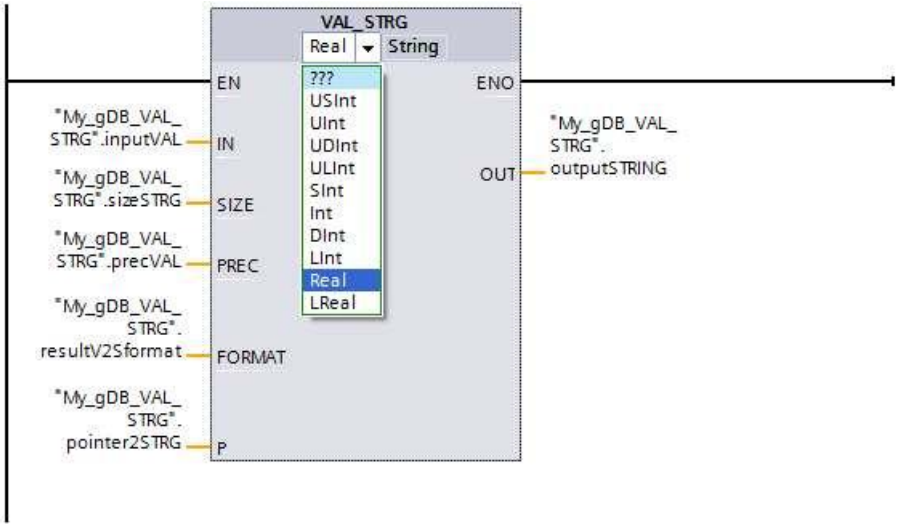
グローバルデータブロックにデータを保存するために、6 つのタグを作成します。

My_gDB_VAL_STRG

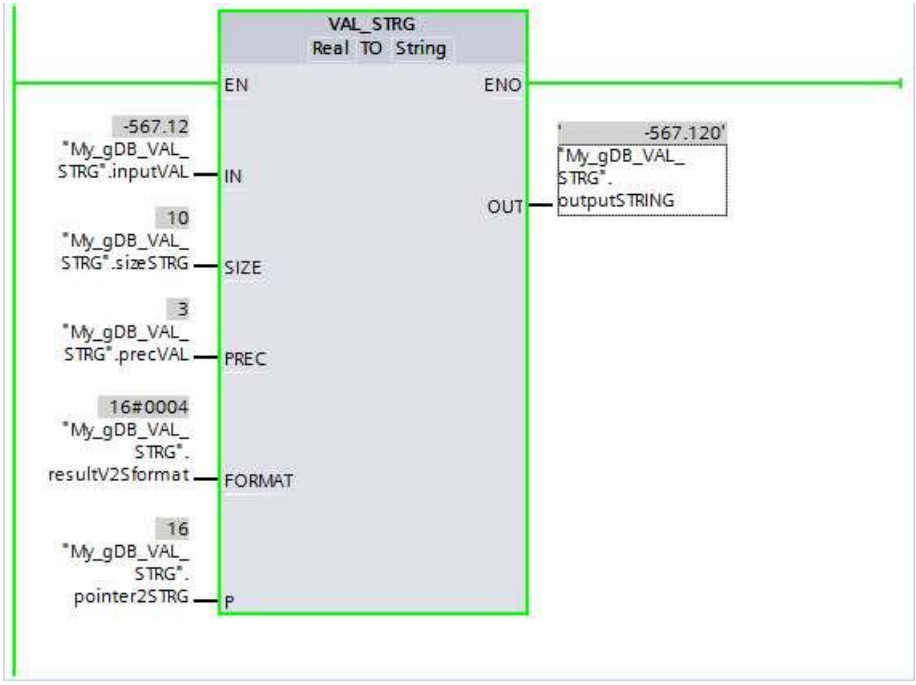
	Name	Data type	Start value
1	Static		
2	inputVAL	Real	-567.12
3	pointer2STRG	UInt	16
4	sizeSTRG	USInt	10
5	precVAL	USInt	3
6	outputSTRING	String	''
7	resultV2Sformat	Word	16#0004

この命令のパラメータを以下のように相互接続します。 データタイプを選択します。

- 左の選択オプションは、変換される値に関するものです。
- 右の選択オプションは、生成される文字列に関するものです。



パラメータ P(「pointer2STRG」)の値「16」に従って、文字列は 16 番目の文字から書き込まれます。 パラメータ SIZE(「sizeSTRG」)の値「10」に従って、文字列はこのピリオドから開始して 10 文字の長さになります。 パラメータ FORMAT(「resultV2Sformat」)の値「0004」により、変換する値のピリオド(「inputVAL」)は、小数点の記号として解釈されます。 パラメータ PREC(「precVAL」)の値「3」に従って、小数点以下の 3 桁の数が文字列に書き込まれます。 変換する値の符号は、文字列の文字として保存され、数字の前に置かれます。 文字列の残りの 10 文字は、符号の前に空白文字として書き込まれます。 文字列は、出力パラメータ OUT(「outputSTRING」)に出力されます。

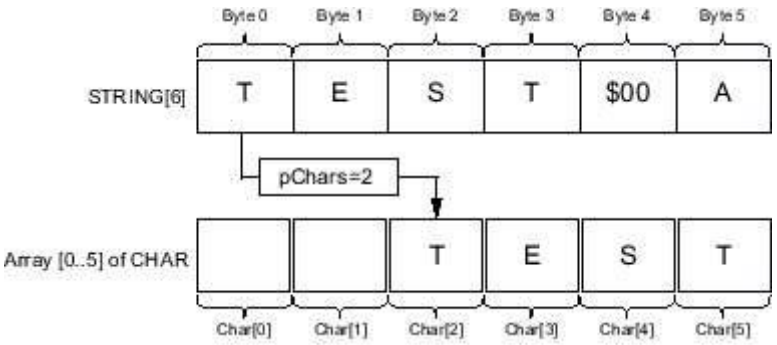


Strg_TO_Chars: 文字列を CHAR 配列に変換

説明

「Strg_TO_Chars」命令を使用して、文字列 STRING を Array of CHAR または Array of BYTE にコピーするか、または文字列 WSTRING を Array of WCHAR にコピーします。 ASCII 文字のみがコピープロセスに対して有効です。

- 入力パラメータ STRG で文字列を指定します。
- 文字は、パラメータ CHARS でデータタイプ Array of CHAR / BYTE / WCHAR に書き込まれます。
  - コピー先のフィールド内の文字数は、ソース文字列からコピーされた文字数と少なくとも同数であるべきです。
  - コピー先のフィールドに含まれる文字数がソース文字列の文字数よりも少ない場合、コピー先のフィールドの最大長まで文字が書き込まれます。
  - 文字列に「$00」または W#16#0000 文字が含まれている場合、コピー操作は対応する位置までしか実行されません(図を参照)。
  - コピーされた文字の数は、パラメータ CNT で出力されます。
- PCHARS パラメータを使用して、コピー先のフィールドで書き込みを開始する位置を指定します。
  - 例: 3 番目の位置から書き込みが行われる場合は、パラメータ PCHARS で値「2」を使用します。



  - PCHARS の既定値は「0」です。 PCHAR = 0 の場合、配列のインデックス下限値を使用します(たとえば、Array [0..5] of CHAR の場合は CHAR[0])。 これはまた、配列の下限値が負の値である場合も同様です(たとえば、Array [-5..5] of CHAR の場合は CHAR[-5])。

注記

S7-1200 V2.0 でのこの命令の使用

バージョン 2.0 までの S7-1200 では、Array [0 .. n] of CHAR / BYTE 以外はサポートされていません。 負のインデックス限界値(たとえば、 Array [-3..2] of CHAR)は許可されていません。 この制限は、ソフトウェアによってチェックされません。

パラメータ

以下の表に、「Strg_TO_Chars」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
STRG	Input	STRING, WSTRING	D、L、または定数	コピー操作のコピー元
PCHARS	Input	DINT	I、Q、M、D、L、P、または定数	文字列の文字が書き込まれる構造体 Array of (W)CHAR / BYTE 内の開始位置。
CHARS	InOut	VARIANT	D、L	コピー操作のコピー先 文字は、Array of (W)CHAR または Array of BYTE データタイプの構造体にコピーできます。
CNT	Output	UINT	I、Q、M、D、L、P	コピーされた文字の文字数。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

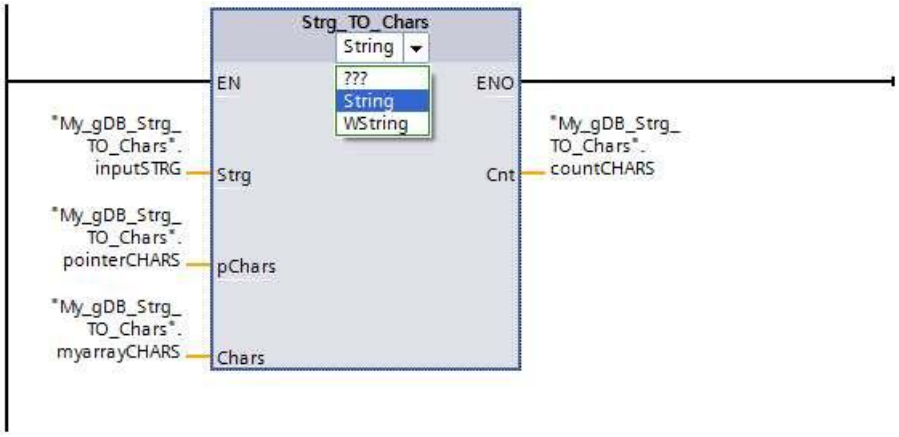
次の例では、Array of CHAR データタイプの文字列の文字を STRING データタイプの構造体にコピーします。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

My_gDB_Strg_TO_Chars

	Name	Data type	Start value
1	Static		
2	inputSTRG	String[10]	'cf7a7a#'
3	pointerCHARS	DInt	2
4	myarrayCHARS	Array[0..9] of Char	
5	myarrayCHARS[0]	Char	' '
6	myarrayCHARS[1]	Char	' '
7	myarrayCHARS[2]	Char	' '
8	myarrayCHARS[3]	Char	' '
9	myarrayCHARS[4]	Char	' '
10	myarrayCHARS[5]	Char	' '
11	myarrayCHARS[6]	Char	' '
12	myarrayCHARS[7]	Char	' '
13	myarrayCHARS[8]	Char	' '
14	myarrayCHARS[9]	Char	' '
15	countCHARS	UInt	0

この命令のパラメータを以下のように相互接続し、文字列のデータタイプを選択します。



個々の文字で構成される構造体は、Array of CHAR データタイプで作成されます。 CHARS 構造体(「myarrayCHARS」)は 10 文字の長さ(Array ~ [0..9])があります。 パラメータ PCHARS(「pointer-CHARS」)の値「2」に従って、構造体の 3 番目の文字から書き込みを開始します(「0」および「1」は空白で、「2」は文字列(「inputSTRG」の最初の文字を含みます))。 文字列(「inputSTRG」)の各文字が構造体(「myarrayCHARS」)に書き込まれると、作成される構造体の最後の文字に空白が書き込まれます。 移動する文字列の文字数は、出力パラメータ CNT(「countCHARS」)に出力されます。

My_gDB_Strg_TO_Chars

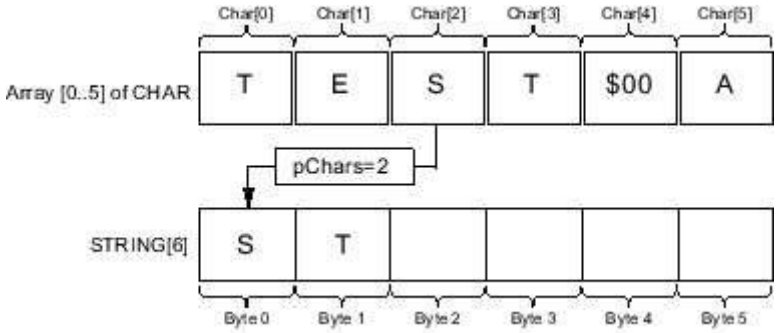
		Name	Data type	Start value	Monitor value
1		Static			
2		inputSTRG	String[10]	'cf7a7a#'	'cf7a7a#'
3		pointerCHARS	DInt	2	2
4		myarrayCHARS	Array[0..9] of Char		
5		myarrayCHARS[0]	Char	' '	' '
6		myarrayCHARS[1]	Char	' '	' '
7		myarrayCHARS[2]	Char	' '	'c'
8		myarrayCHARS[3]	Char	' '	'f'
9		myarrayCHARS[4]	Char	' '	'7'
10		myarrayCHARS[5]	Char	' '	'a'
11		myarrayCHARS[6]	Char	' '	'7'
12		myarrayCHARS[7]	Char	' '	'a'
13		myarrayCHARS[8]	Char	' '	'#'
14		myarrayCHARS[9]	Char	' '	' '
15		countCHARS	UInt	0	7

Chars_TO_Strg: CHAR 配列を文字列に変換

説明

「Chars_TO_Strg」命令を使用して、文字を Array of CHAR または Array of BYTE から STRING にコピーするか、または ARRAY of WCHAR から WSTRING.にコピーします。 コピーでは ASCII 文字のみ有効です。

- 入力パラメータ CHARS の文字列にコピーする Array of (W)CHAR / BYTE の文字を指定します。
- 文字は、パラメータ STRG で(W)STRING データタイプ に書き込まれます。
  - 文字列内の文字数は、ソースフィールドからコピーされた文字数と少なくとも同数であるべきです。
  - 文字列がソースフィールド内の文字数よりも短い場合、文字列の最大長まで文字が書き込まれます。
  - Array of (W)CHAR / BYTE に「$00」文字が含まれている場合、または Array of (W)CHAR に W#16#0000 文字が含まれている場合、コピー操作は対応する位置までしか実行されません(図を参照)。
- PCHARS パラメータを使用し、コピーする文字のソースフィールドでの開始位置を指定します。PCHARS = 0 が既定値で、負の値の場合でも常に配列のインデックス下限値を指定します。
  - 例: ソースフィールドの 3 番目の文字からコピーを開始する場合、パラメータ PCHARS で値「2」を使用します。



  - パラメータ PCHARS でインデックスが指定された場合で、かつそのインデックスがコピーソースに含まれていない場合(たとえば、Array [0..5] of CHAR で「7」)、命令は実行されません.

注記

S7-1200 V2.0 でのこの命令の使用

バージョン 2.0 までの S7-1200 では、Array [0 .. n] of CHAR / BYTE 以外はサポートされていません。 負のインデックス限界値(たとえば、 Array [-3..2] of CHAR)は許可されていません。 この制限は、ソフトウェアによってチェックされません。

パラメータ

以下の表に、「Chars_TO_Strg」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CHARS	Input	VARIANT	D、L	コピー操作のコピー元

				Array of (W)CHAR / BYTE から文字のコピーを開始。
PCHARS	Input	DINT	I、Q、M、D、L、P、または定数	文字がコピーされる Array of (W)CHAR / Array of BYTE 内の開始位置。
CNT	Input	UINT	I、Q、M、D、L、P	コピーされる文字数「0」を使用して、すべての文字をコピーします。
STRG	Output	STRING, WSTRING	D、L	コピー操作のコピー先 (W)STRING データタイプの文字列。 データタイプの最大長を遵守します。 • STRING: 254 文字 • WSTRING: 254 文字(既定) / 16382 文字(最大) WSTRING を使用する場合、254 文字を超える長さを角括弧で明示的に定義しなければならないことに注意してください (WSTRING[16382]など)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

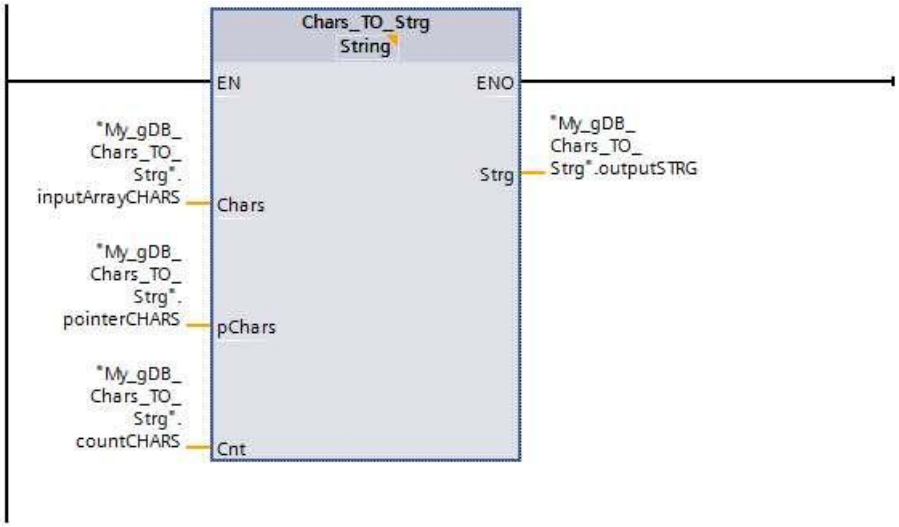
次の例では、Array of CHAR データタイプの構造体の文字を STRING データタイプの文字列にコピーします。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

My_gDB_Chars_TO_Strg

	Name	Data type	Start value
1	Static		
2	inputArrayCHARS	Array[0..9] of Char	
3	inputArrayCHARS[0]	Char	'M'
4	inputArrayCHARS[1]	Char	'y'
5	inputArrayCHARS[2]	Char	'S'
6	inputArrayCHARS[3]	Char	'7'
7	inputArrayCHARS[4]	Char	'P'
8	inputArrayCHARS[5]	Char	'L'
9	inputArrayCHARS[6]	Char	'C'
10	inputArrayCHARS[7]	Char	' '
11	inputArrayCHARS[8]	Char	' '
12	inputArrayCHARS[9]	Char	' '
13	pointerCHARS	DInt	2
14	outputSTRG	String	''
15	countCHARS	UInt	0

この命令のパラメータを以下のように相互接続し、文字列のデータタイプを選択します。



CHARS 構造体(「inputArrayCHARS」)は 10 文字の長さ(Array ~ [0..9])があります。 パラメータ PCHARS(「pointerCHARS」)の値「2」に従って、構造体の 3 番目の位置から文字列(「outputSTRG」)へ文字をコピーします。 パラメータ CNT(「countCHARS」)が値「0」を持つため、位置「2」から、構造体(「inputArrayCHARS」)のすべての文字が文字列(「outputSTRG」)にコピーされます。

My_gDB_Chars_TO_Strg

	Name	Data type	Start value	Monitor value
1	Static			
2	inputArrayCHARS	Array[0..9] of Char		
3	inputArrayCHARS[0]	Char	'M'	'M'
4	inputArrayCHARS[1]	Char	'y'	'y'
5	inputArrayCHARS[2]	Char	'S'	'S'
6	inputArrayCHARS[3]	Char	'7'	'7'
7	inputArrayCHARS[4]	Char	'P'	'P'
8	inputArrayCHARS[5]	Char	'L'	'L'
9	inputArrayCHARS[6]	Char	'C'	'C'
10	inputArrayCHARS[7]	Char	' '	' '
11	inputArrayCHARS[8]	Char	' '	' '
12	inputArrayCHARS[9]	Char	' '	' '
13	pointerCHARS	DInt	2	2
14	outputSTRG	String	''	'S7PLC '
15	countCHARS	UInt	0	0

MAX_LEN:文字列長の定義

説明

(W)STRING データタイプのタグは、2 つの長さを含みます。 最大長と現在の長さ(現在有効な文字の数)があります。

- それぞれのタグの文字列の最大長は、STRING キーワードに角括弧内で指定されます。 文字列で占められたバイトの数は 2 で、最大長を超えます。
- それぞれのタグの文字列の最大長は、WSTRING キーワードに角括弧内で指定されます。 文字列で占められたワードの数は 2 で、最大長を超えます。
- 現在の長さは、実際に使用されている文字の桁数を示します。 現在の長さは、最大長以下であることが必要です。

「MAX_LEN」命令を使用し、IN 入力パラメータで指定された文字列の最大長を照会して、この情報を OUT 出力パラメータに数値として出力します。

命令の処理中にエラーが発生すると、空の文字列が出力されます。

注記

現在の長さの読み出し

「LEN」命令を使用して、文字列の現在の長さを読み出すことも可能です。

パラメータ

以下の表に、「MAX_LEN」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	文字列
OUT	Return	INT	I、Q、M、D、L、P	最大文字数

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

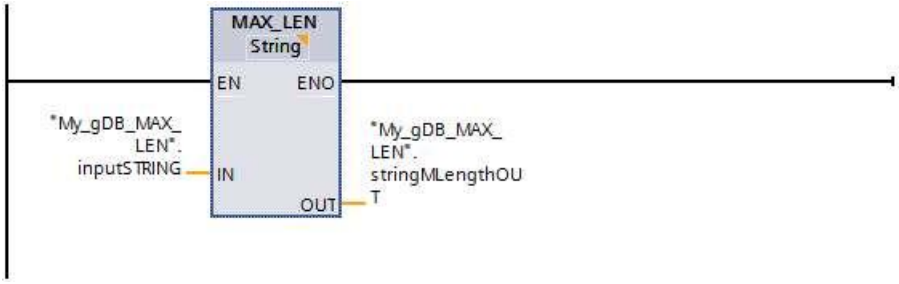
次の例では、STRING データタイプの文字列の最大長を定義します。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

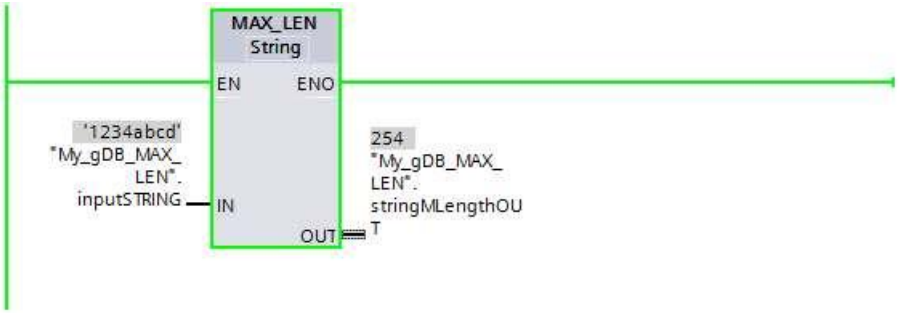
My_gDB_MAX_LEN

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	stringMLengthOUT	Int	0

この命令のパラメータを以下のように相互接続します。



指定された文字列(「inputSTRING」)の最大長が判定され、出力パラメータ「OUT」(「stringMLength-OUT」)に数値として返されます。



JOIN: 複数の文字列の結合

説明

「JOIN」命令は複数の文字列を 1 つの配列に結合します。

複数の文字列を単一の文字列に変換するために、この命令は次のファンクションを提供します。

- **形式の選択**

  Mode パラメータの先頭ビットを使用して、配列のソース文字列のシーケンスを CSV または FSR 形式に指定することができます。

  次の例では、2 つのソース文字列がテーブルの 2 つの列により指定されています。 最大文字数は、最初のソース文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

1963	1974
Miller	Jackson
John	Peter
Roadname	VeryLongRoadna

  - CSV (カンマ区切り値)を使用する場合、ソース文字列の内容が連続的に宛先配列に書き込まれ、区切り文字で区切られます (以下の例を参照)。
  - FSR (固定サイズレコード)を使用する場合、ソース文字列ごとに宛先配列での一定の文字数が定義されます。 ソース文字列の文字が宛先配列で空白文字の予約が不要な場合、対応する配列エレメントは区切り文字で埋められます。 ただし、ソース文字列の文字数が予約された空白よりも大きい場合は、対応する配列エレメントは前から埋められ、ソース文字列の超過した文字は切り捨てられます(下の例を参照)。

- **ソース文字列の区切り文字の選択**

  RecSeparator パラメータを使用して、文字列ごとに使用する区切り文字を選択できます。 SrcStruct パラメータでの入力文字列の内容に基づいて文字を選択すべきです。 たとえば、入力文字列が文字列内にカンマを含んでいる場合は、区切り文字としてカンマを使用すべきではありません。 区切り文字を配列への書き込み内容に含めるためには、区切り文字に使用するデータタイプが DstArray パラメータで宛先配列と一致する必要があります。

- **すべてのソース文字列の終わりの区切り文字の選択**

  モードパラメータの 3 番目のビットを使用して、宛先配列(DstArray パラメータ)においてコピーされた文字の終わりに区切り文字として追加文字を書き込むかどうかを指定できます。 EndSeparator パラメータで区切り文字として使用する文字を指定します。 必ず RecSeparator パラメータ(個々の文字列の区切り文字)について設定されている文字以外の文字を使用してください。 2 つの区切り文字が区別できない場合、「SPLIT」命令を使用して変換を逆転した場合に予期しない結果が生じます。

- **ソース文字列の選択**

  SrcStruct パラメータでソース文字列を指定します。 データタイプについて、Array of STRING または Array of WSTRING、あるいは STRING または WSTRING データタイプのみを含む構造体を使用できます。 これは、ユーザーデータタイプやネスト構造にも適用されます。 これらにデータタイプ STRING または WSTRING のみがそれぞれ含まれている限り、これらを使用できます。

- **結合された文字列の数の指定**

  SrcStruct パラメータ(ソース文字列)で Array of STRING または Array of WSTRING を使用する場合(ネスト構造なし)、Count パラメータを使用して、単一文字列を形成するために結合されているソース文字列の数を指定できます。 SrcStruct パラメータで Array of (W)STRING 以外のデータタ

イプを使用した場合、Count パラメータは無視されます。 これにより、大きな配列の一部のみを結合することが可能になります。

- **配列を書き込むための宛先領域の選択**

  DstArray パラメータで Array of (W)CHAR データタイプを使用します。 ここで STRING または WSTRING データタイプを使用することはできません。STRING について長さが 254 文字または 256 バイトに制限されているためです。

- **配列内の位置のインデックス(DestArray ターゲットパラメータ)**

  変換はこの位置で開始され、命令は、変換が完了する位置を決定する位置パラメータを読み出します。 これにより、配列を埋めるための命令のフォローアップコールが可能になります。

パラメータ

以下の表に、「JOIN」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
Mode	Input	DWORD	I、Q、M、D、L、または定数	文字列との結合方法を指定します(「Mode」パラメータを参照)。
RecSeparator	Input	VARIANT	I、Q、M、D、L	ソース文字列の区切り文字 • CSV を使用する場合 個々の文字列の区切り文字として使用される文字。 • FSR を使用する場合 個々の文字列のフィル文字として使用される文字。
EndSeparator	Input	VARIANT	I、Q、M、D、L、または定数	変換の終わりの区切り文字 モードパラメータについてビット 3 = 1 がセットされている場合に、文字の終わりに書き込まれる区切り文字。
SrcStruct	Input	VARIANT	I、Q、M、D、L	ソース文字列へのポインタ。
Count	Input	UDINT	I、Q、M、D、L	結合された文字列の数。 Count パラメータを使用できるのは、SrcStruct パラメータで Array of (W)STRING が使用されているときだけです。
DestArray	InOut	VARIANT	I、Q、M、D、L	変換後に文字が書き込まれる領域。 DestArray パラメータで Array [0 .. x] of CHAR/WCHAR データタイプを使用します。 SrcStruct パラメータでのソース文字列の長さに基づいて配列の長さ(x)を設定できます。
Position	InOut	UDINT	I、Q、M、D、L	合計文字列内の位置のインデックスを作成します。
Ret_Val	Return	INT	I、Q、M、D、L	命令のステータス(「RET_VAL パラメータ」の表を参照)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

Mode パラメータ

ビット	ビット値「0」	ビット値「1」	説明
0	CSV 形式(カンマ区切り値)	FSR 形式(固定サイズレコード)	形式の選択 • CSV を使用する場合、ソース文字列は宛先配列において区切り文字で区切られます。 • FSR を使用する場合、ソース文字列は、RecSeparator パラメータで定義されたフィル文字を使用して宛先配列に書き込まれます。
1	-	-	「JOIN」命令には無関係です。
2	-	-	予約済み(ビット値には無関係)
3	追加区切り文字を書き込みません。	読み出された文字の終わりに、EndSeparator パラメータによって定義された文字を書き込みます。	配列(DestArray パラメータ)において文字の終わりに区切り文字として追加文字を書き込むかどうかを選択します。
4	-	-	「JOIN」命令には無関係です。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0001	追加文字が無視されます。
8190	Mode パラメータでの選択はサポートされていません。
8x20	ソース文字列が無効です。
8x53	VARIANT が短すぎるデータ構造体をポイントしています。
8x54	無効なデータタイプ
8082	パラメータ Count が SrcStruct のソース文字列数より大きい値です。
8x84	追加文字を検出
8xB4	パラメータ SrcStruct (ソース)と DestArray (宛先)または区切り記号(パラメータ RecSeparator および EndSeparator)でデータタイプが異なります。
80B5	命令でバッファオーバーフローが発生しました。 文字がパラメータ DestArray に完全に出力されていない、またはパラメータ Position の値が DestArray の範囲外です。
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得

* エラーコードについて以下のことに注意してください。

- エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。
- リストされているエラーコードの 2 番目の位置にある「x」は、エラーの原因となったパラメータを表します。

例: 16 進のエラーコード 8352 = 3 番目のパラメータ(EndSeparator)でエラーが発生しました。パラメータの表を参照してください。

- エラーを特定のパラメータに明確に割り当てられない場合は、「0」が出力されます。

  例: CHAR データタイプが区切り文字(RecSeparator パラメータ)に使用されています。WCHAR が DestArray パラメータで配列のデータタイプとして使用されています。 この場合、エラーコード 80B4 が出力されます。

宛先配列が CSV 形式である場合の JOIN 命令の 2 つの例

- 最初の例:

  以下のソース文字列があります。
  - 1963
  - Miller
  - John
  - Roadname

  「,」を区切り文字として選択した場合、JOIN の呼び出しにより以下の宛先配列が作成されます。

  ```
  1963,Miller,John,Roadname
  ```

- 2 番目の例:

  以下のソース文字列があります。
  - 1974
  - Jackson
  - Peter
  - VeryLongRoadname

  「,」を区切り文字として選択した場合、JOIN の呼び出しにより以下の宛先配列が作成されます。

  ```
  1974,Jackson,Peter,VeryLongRoadname
  ```

宛先配列が FSR 形式である場合の JOIN 命令の 2 つの例

- 最初の例:

  以下のソース文字列があります。
  - 1963
  - Miller
  - John
  - Roadname

  宛先配列の予約文字数は、最初のソース文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

  「,」をフィル文字として選択した場合、JOIN の呼び出しにより以下の宛先配列が作成されます。

  ```
  1963Miller,,,,,,,John,,,,,,Roadname,,,,,,
  ```

- 2 番目の例:

  以下のソース文字列があります。
  - 1974
  - Jackson
  - Peter
  - VeryLongRoadname

宛先配列の予約文字数は、最初のソース文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

「,」をフィル文字として選択した場合、JOIN の呼び出しにより以下の宛先配列が作成されます。

```
1974Jackson,,,,,,Peter,,,,,VeryLongRoadna
```

SPLIT: 文字の配列を複数の文字列に分割

説明

「SPLIT」命令は、配列(Array of CHAR / WCHAR)を複数の個別の文字列(Array of STRING / WSTRING または構造体)に変換します。

配列を複数の文字列に変換するには、次の情報を指定します。

- **読み出す配列の選択**

  SrcArray パラメータで読み出す配列を指定します。 入力および出力パラメータに使用されるデータタイプが、使用されるその他のパラメータと一致することを確認してください。 たとえば、SrcArray パラメータで CHAR データタイプの配列を使用する場合は、区切り文字(Rec/EndSeparator)に CHAR データタイプを使用する必要があります。また、構造体は、DestStruct パラメータで STRING データタイプの文字列のみを含んでいる必要があります。

- **ソース配列のフォーマット選択**

  Mode パラメータの先頭ビットを使用して、読み出す配列を CSV または FSR 形式に指定します。

  - CSV (カンマ区切り値)を使用する場合、ソース配列に含まれる文字は次の文字を区切り文字として区切られています。
    ソース配列の 2 つの例:

    ```
    1963,Miller,John,CitynameA,Roadname
    1974,Jackson,Peter,CitynameB,VeryLongRoadname
    ```

  - FSR (固定サイズレコード)を使用する場合、ソース配列の論理情報の項目ごとに一定の文字数が定義されます。 情報の各項目は、定義されたスペースに合わせます。 情報が定義されたスペースを必要としない場合は、区切り文字で埋められます。

    ソース配列の 2 つの例の情報の長さは、最初の情報項目(生年)が 4 文字、2 番目(姓)が 13 文字、3 番目が 10 文字(名)、4 番目(都市)が 9 文字、5 番目(番地等)が 16 文字です。

    ```
    1963Miller,,,,,,,John,,,,,,CitynameARoad-
    name,,,,,,,,
    1974Jackson,,,,,,Peter,,,,,CitynameBVeryLongRoad-
    name
    ```

- **読み出す配列に使用される区切り文字**
  - 読み出す配列が CSV 形式を持つ場合、RecSeparator パラメータでどの区切り文字が使用されたかを指定します。
  - 読み出す配列が FSR 形式を持つ場合、RecSeparator パラメータでどのフィル文字が使用されたかを指定します。

- **合計文字列の終わりの区切り文字の選択**

  EndSeparator パラメータで、その後で配列の読み出しが停止する区切り文字を指定します。「SPLIT」命令はこの位置で停止し、検出された文字列を出力します。 EndSeparator 区切り文字は、RecSeparator 区切り文字よりも高い優先度で評価されることに注意すべきです。 EndSeparator パラメータの区切り文字が(2 つの RecSeparator 区切り文字の間で)読み出す文字列で使用された場合、EndSeparator 区切り文字の後のすべての内容は無視されます。

- **配列の読み出しが開始される位置の指定**

  変換は配列内のこの位置で開始され、命令は、変換が完了する位置を決定する位置パラメータを読み出します。 これにより、DestStruct パラメータでさまざまな文字列を埋めるための命令のフォローアップコールが可能になります。

- **読み出された文字列の数の出力**

  DestStruct パラメータで Array of STRING を使用した場合は、Count パラメータを使用して、読み出された文字列の数を出力できます。 内容を含む文字列以外はカウントされません。 DestStruct パラメータで Array of STRING 以外のデータタイプを使用した場合、Count パラメータで「0」が出力されます。

パラメータ

以下の表に、「SPLIT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
Mode	Input	DWord	I、Q、M、D、L、または定数	複数の文字列への分割方法を指定します(「Mode」パラメータを参照)。
RecSepara-tor	Input	Variant	I、Q、M、D、L	区切り文字またはフィル文字 • CSV を使用する場合 読み出す配列において個々の文字列を識別するために使用された文字。 • FSR を使用する場合 読み出す配列においてフィル文字として使用された文字。
EndSepara-tor	Input	Variant	I、Q、M、D、L	読み出す配列において合計文字列の終わりを定義するために使用される区切り文字。
SrcArray	Input	Variant	I、Q、M、D、L	読み出す配列(Array of CHAR/ WCHAR)へのポインタ
DestStruct	InOut	Variant	I、Q、M、D、L	変換された文字列(Array of STRING / WSTRING)を含む構造体。
Position	InOut	UDInt	I、Q、M、D、L	SrcArray パラメータでの配列の読み出し元となる位置。
Ret_Val	Return	Int	I、Q、M、D、L	命令実行の結果/エラーコード(「Ret_Val パラメータ」の表を参照)
Count	Output	UDInt	I、Q、M、D、L	検出された文字列の数。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ「Mode」

ビット	ビット値「0」	ビット値「1」	説明
0	CSV 形式(カンマ区切り値)	FSR 形式(固定サイズレコード)	基本モード: CSV または FSR の選択
1	• CSV を使用する場合 文字が超過すると エラーになります。	• CSV を使用する場合追加文字が無視されます。	ビット 1 を使用して、追加文字をどのように処理するかを指定します。 • CSV を使用する場合

	• FSR を使用する場合 追加のフィル文字が文字列内に残ります。	• FSR を使用する場合 追加のフィル文字が削除されます。	○ このビットがセットされた場合、宛先文字列に収まらない追加文字は無視されます。 例: 命令が長さが 16 文字の文字列(STRING[16]データタイプ)に書き込みます。 ソースには、最初の 16 文字の後に区切り文字は含まれていません。 ビット 1 がセットされた場合、追加文字は無視され、命令は配列の読み出しを続行します。 ○ このビットがセットされなかった場合、そのような場合に命令は停止し、Ret_Val パラメータでエラーメッセージを生成します。 • FSR を使用する場合 ○ ビットがセットされた場合、情報を含む文字の右側にあるフィル文字は、ソース配列が宛先文字列に転送されたときに宛先文字列に書き込まれません(例を参照)。 ○ ビットがセットされなかった場合、情報を含む文字の右側にあるフィル文字は、ソース配列が宛先文字列に転送されたときに宛先文字列には書き込まれます(例を参照)。
2	-	-	今後のバージョンで使用するために予約済み
3	区切り記号が保持されます。	区切り記号が削除されます。	合計文字列の終わりで区切り文字を削除するかどうかの指定。
4	未書き込み文字列(STRING)の長さをそのままにします。	未書き込み文字列(STRING)を長さ「0」に設定します。	DestStruct パラメータでの未使用の文字列(STRING)を長さ「0」に設定するかどうかの指定。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0001	追加文字が無視されます。
8190	Mode パラメータでの選択はサポートされていません。
8x20	ソース文字列が無効です。
8x53	VARIANT が短すぎるデータ構造体をポイントしています。
8x54	無効なデータタイプ
8x84	追加文字を検出しました。
8xB4	パラメータ SrcArray (ソース)と DestStruct (宛先)または区切り記号(パラメータ ReSeparator および EndSeparator)でデータタイプが異なります。
80B5	命令でバッファオーバーフローが発生しました。 文字列がパラメータ DestStruct に完全に出力されていません。
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードについて以下のことに注意してください。	

- エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。
- リストされているエラーコードの 2 番目の位置にある「x」は、エラーの原因となったパラメータを表します。

  例:16 進のエラーコード 8352 = 3 番目のパラメータ(EndSeparator)でエラーが発生しました。パラメータの表を参照してください。
- エラーを特定のパラメータに明確に割り当てられない場合は、「0」が出力されます。

  例:CHAR データタイプが区切り文字(RecSeparator パラメータ)に使用されています。WCHAR が DestArray パラメータで配列のデータタイプとして使用されています。この場合、エラーコード 80B4 が出力されます。

読み出す配列が CSV 形式の場合の SPLIT 命令の 2 つの例

最初のステップで、ソース配列は、区切り文字(RecSeparator パラメータ、「,」など)によって指定された文字列に分割されます。

2 番目のステップで、分割後の文字列は宛先文字列に保存されます。 DestStruct パラメータを使用して文字列の長さを指定できます。

分割で作成された文字列が宛先文字列よりも文字数が大きい場合、この後の SPLIT の動作は Mode パラメータによって異なります。 Mode=2#00 の場合、これはエラーになります。 関連するエラーコードが返され、他の宛先文字列は埋められません。 Mode=2#10 の場合、余った文字は無視され、次の宛先文字列は次の区切り文字の後でフィル文字で埋められます。

- 最初の例:

  この例では、上記の 2 つのソース配列があります。
  - `1963,Miller,John,Roadname`
  - `1974,Jackson,Peter,VeryLongRoadname`

  宛先文字列の長さは、最初の宛先文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

  Mode=2#10 の場合は、SPLIT の最初の呼び出しにより以下の宛先文字列が作成されます。
  - `1963`
  - `Miller`
  - `John`
  - `Roadname`

  Mode=2#10 の場合は、SPLIT の 2 回目の呼び出しにより以下の宛先文字列が作成されます。
  - `1974`
  - `Jackson`
  - `Peter`
  - `VeryLongRoadna`
- 2 番目の例:

  この例では、宛先配列の論理情報は予期された順序にはなりません。 上記の 2 つのソース配列があります。
  - `1963,Miller,Roadname,John`
  - `1974,Jackson,VeryLongRoadname,Peter`

  最初の例と同様に、宛先文字列の長さは、最初の宛先文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

  Mode=2#10 の場合は、SPLIT の最初の呼び出しにより以下の宛先文字列が作成されます。
  - `1963`
  - `Miller`

- Roadname
- John

Mode=2#10 の場合は、SPLIT の 2 回目の呼び出しにより以下の宛先文字列が作成されます。

- 1974
- Jackson
- VeryLongRo
- Peter

読み出す配列が FSR 形式の場合の SPLIT 命令の 2 つの例

ソース配列は、DestStruct パラメータで指定した宛先文字列の長さに従って分割されます。

フィル文字(RecSeparator パラメータ、たとえば「,」)については、SPLIT の動作は Mode パラメータによって異なります。 Mode=2#01 の場合、フィル文字は宛先文字列に入力され、Mode=2#11 の場合、フィル文字は入力されません。

- 最初の例:

  この例では、上記の 2 つのソース配列があります。

  - 1963Miller,,,,,,,John,,,,,,Roadname,,,,,,
  - 1974Jackson,,,,,,Peter,,,,,VeryLongRoadname

  宛先文字列の長さは、最初の宛先文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

  Mode=2#01 の場合は、SPLIT の最初の呼び出しにより以下の宛先文字列が作成されます。

  - 1963
  - Miller,,,,,,,
  - John,,,,,,
  - Roadname,,,,,,

  Mode=2#01 の場合は、SPLIT の 2 回目の呼び出しにより以下の宛先文字列が作成されます。

  - 1974
  - Jackson,,,,,,
  - Peter,,,,,
  - VeryLongRoadna

- 2 番目の例:

  この例では、宛先配列の論理情報は予期された順序にはなりません。 上記の 2 つのソース配列があります。

  - 1963Miller,,,,,,,Roadname,,,,,,John,,,,,,
  - 1974Jackson,,,,,,VeryLongRoadnamePeter,,,,,

  最初の例と同様に、宛先文字列の長さは、最初の宛先文字列が 4 文字、2 番目が 13 文字、3 番目が 10 文字、4 番目が 14 文字です。

  Mode=2#11 の場合は、SPLIT の最初の呼び出しにより以下の宛先文字列が作成されます。

  - 1963
  - Miller
  - Roadname
  - ,,,,John

  説明: 言ってみれば、SPLIT 命令はソース配列にテンプレートを適用します。 このテンプレートの長さは、宛先文字列の長さによって決まります。 宛先文字列への転送中に、情報内容を含む文字の右側にあるカンマのみがフィル文字として解釈されます。 Mode=2#11 の場合は、フィル文字は入力されません。最初の呼び出しで、「John」の右側にあるカンマはフィル文字です。 対応する宛先

文字列の末尾に達する前に「John」の文字があるため、「John」の左側にあるカンマはフィル文字ではありません。 したがって、「John」の左側にあるカンマは宛先文字列に表示されます。

Mode=2#11 の場合は、SPLIT の 2 回目の呼び出しにより以下の宛先文字列が作成されます。

- 1974
- Jackson
- VeryLongRo
- adnamePeter

ATH: ASCII 文字列を 16 進数に変換

説明

「ATH」命令を使用し、IN 入力パラメータで指定された ASCII 文字列を 16 進数に変換します。 変換の結果は、OUT 出力パラメータに出力されます。

- IN パラメータのポインタを使って次のデータタイプを参照することができます(ASCII)。 STRING, Array of CHAR, Array of BYTE, WSTRING, Array of WCHAR.
- OUT パラメータのポインタを使って次のデータタイプを参照することができます(16 進数)。 ビット列、整数、STRING、Array of CHAR、Array of BYTE、WSTRING、Array of WCHAR.

N パラメータを使用し、変換する ASCII 文字の数を指定します。 最大 32767 の有効な ASCII 文字を変換できます。 数字「0」~「9」、大文字「A」~「F」、および小文字「a」~「f」のみを変換できます。 その他すべての文字は、ゼロに変換されます。

ASCII 文字には 8 ビットが必要で、16 進数には 4 ビットのみが必要であるため、出力ワード長は入力ワード長の半分になります。 ASCII 文字は変換され、読み込まれた順と同じ順で出力に配置されます。奇数の ASCII 文字がある場合、その 16 進数は最後に変換された 16 進数の右側が二ブルのゼロで充当されます。

パラメータ

次の表に、「ATH」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	VARIANT	I、Q、D、L	ASCII 文字列へのポインタ
N	Input	INT	I、Q、M、D、L、または定数	変換する ASCII 文字数
RET_VAL	Return	WORD	I、Q、M、D、L	命令のステータス
OUT	Output	VARIANT	I、Q、M、D、L	16 進数

有効なデータタイプに関する追加情報は、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード (W#16#....)*	説明
0000	エラーは発生していません。
0007	無効な文字. 次の ASCII 文字のみ使用できます。 数字「0」~「9」、大文字「A」~「F」、小文字「a」~「f」。
8101	IN パラメータの無効なポインタ。たとえば、存在しないデータブロックが参照されているなどの理由。
8182	N パラメータのデータに対して入力バッファが小さすぎます。
8120	IN パラメータの書式が無効です。
8151	IN パラメータのサポートされていないデータタイプ。

8401	OUT パラメータの無効なポインタ。たとえば、存在しないデータブロックが参照されているなどの理由。
8482	N パラメータのデータに対して出力バッファが小さすぎます。
8420	OUT パラメータの書式が無効です。
8451	OUT パラメータのサポートされていないデータタイプ。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

ASCII 文字および 16 進数の番号

次の表に、ASCII 文字と対応する 16 進数の値を示します。

ASCII 文字	ASCII コード化された 16 進数の値	16 進数の数字
"0"	30	0
"1"	31	1
"2"	32	2
"3"	33	3
"4"	34	4
"5"	35	5
"6"	36	6
"7"	37	7
"8"	38	8
"9"	39	9
"A"	41	A
"B"	42	B
"C"	43	C
"D"	44	D
"E"	45	E
"F"	46	F

例

次の表に、ASCII 文字列から 16 進数の番号への変換の例を示します。

IN	N	OUT	ENO ステータス
'0123'	4	16#0123	1
'123AFx1a23'	10	16#123AF01a23	0

HTA: 16 進数を ASCII 文字列に変換

説明

「HTA」命令を使用し、IN 入力パラメータで指定された 16 進数を ASCII 文字列に変換します。 変換の結果は、OUT パラメータで指定されたアドレスに格納されます。

- IN パラメータのポインタを使用して、次のデータタイプを参照することができます(16 進数)。 ビット列、整数、STRING、Array of CHAR、Array of BYTE、WSTRING、Array of WCHAR。
- OUT パラメータのポインタを使用して、次のデータタイプを参照することができます(ASCII)。 STRING, Array of CHAR, Array of BYTE , WSTRING, Array of WCHAR.

N パラメータを使用し、変換する 16 進数のバイト数を指定します。 ASCII 文字には 8 ビットが必要で、16 進数には 4 ビットのみが必要であるため、出力値は入力値の 2 倍になります。 16 進数の 1 ニブルは 1 文字に変換され、元の順番が維持されます。

最大 32767 文字を ASCII 文字列に書き込むことができます。 変換の結果は、「0」 ~「9」までの数字と「A」 ~「F」の大文字で表示されます。

OUT パラメータの変換の結果が全部表示できない場合、結果は部分的にパラメータに書き込まれます。

パラメータ

以下の表に、「HTA」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	VARIANT	I、Q、M、D、L	16 進数の開始アドレス
N	Input	UINT	I、Q、M、D、L、または定数	変換する 16 進数のバイト数
RET_VAL	Return	WORD	I、Q、M、D、L	エラーメッセージ
OUT	Output	VARIANT	I、Q、D、L	結果が格納されるアドレス。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#....)	説明
0000	エラーは発生していません。
8101	IN パラメータの無効なポインタ。たとえば、存在しないデータブロックが参照されているなどの理由。
8182	N パラメータのデータに対して入力バッファが小さすぎます。
8120	IN パラメータの書式が無効です。
8151	IN パラメータのサポートされていないデータタイプ。

8401	OUT パラメータの無効なポインタ。たとえば、存在しないデータブロックが参照されているなどの理由。
8482	N パラメータのデータに対して出力バッファが小さすぎます。
8420	OUT パラメータの書式が無効です。
8451	OUT パラメータのサポートされていないデータタイプ。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

ASCII 文字および 16 進数の番号

次の表に、ASCII 文字と対応する 16 進数の値を示します。

16 進数の数字	ASCII コード化された 16 進数の値	ASCII 文字
0	30	"0"
1	31	"1"
2	32	"2"
3	33	"3"
4	34	"4"
5	35	"5"
6	36	"6"
7	37	"7"
8	38	"8"
9	39	"9"
A	41	"A"
B	42	"B"
C	43	"C"
D	44	"D"
E	45	"E"
F	46	"F"

例

次の表は、16 進数の ASCII 文字列への変換の例を示します。

IN	N	OUT	ENO ステータス
W#16#0123	2	'0123'	1
16#123AF01023	4	'123AF010'	0

LEN: 文字列長の定義

説明

(W)STRING データタイプのタグは、2 つの長さを含みます。 最大長と現在の長さ(現在有効な文字の数)があります。

- それぞれのタグの文字列の最大長は、STRING キーワードに角括弧内で指定されます。 文字列で占められたバイトの数は 2 で、最大長を超えます。
- それぞれのタグの文字列の最大長は、WSTRING キーワードに角括弧内で指定されます。 文字列で占められたワードの数は 2 で、最大長を超えます。
- 現在の長さは、実際に使用されている文字の桁数を示します。 現在の長さは、最大長以下であることが必要です。

「LEN」命令を使用し、IN 入力パラメータで指定された文字列の現在の長さを照会して、この情報を OUT 出力パラメータに数値として出力します。 空の文字列(")の長さはゼロです。

命令の処理中にエラーが発生すると、空の文字列が出力されます。

注記

最大長の読み出し

「MAX_LEN」命令を使用して、文字列の最大長を読み出すことが可能です。

パラメータ

以下の表に、「LEN」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	文字列
OUT	Return	INT, DINT, REAL, LREAL	I、Q、M、D、L	有効な文字の数

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

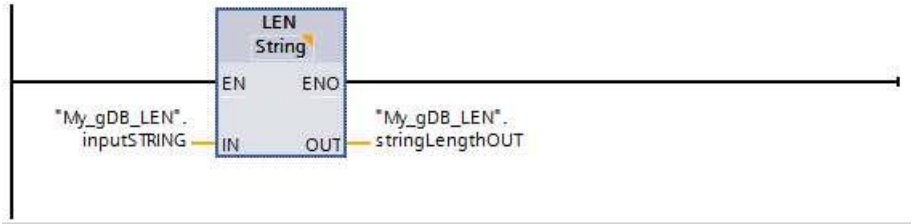
次の例では、STRING データタイプの文字列長を定義します。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

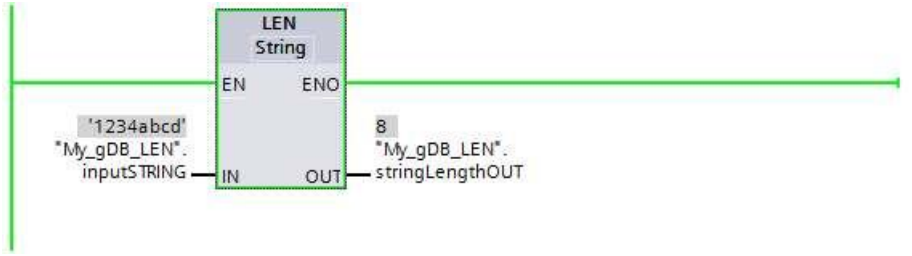
My_gDB_LEN

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	stringLengthOUT	Int	0

この命令のパラメータを以下のように相互接続します。



文字列(「inputSTRING」)の実際に占有された文字数が判定され、出力パラメータ「OUT」(「string-LengthOUT」)に数値として返されます。



CONCAT: 文字列の結合

説明

「CONCAT」命令を使用し、IN1 入力パラメータの文字列と IN2 入力パラメータの文字列を結合します。 結果は、(W)STRING フォーマットで OUT 出力パラメータに出力されます。 結果の文字列が、OUT 出力パラメータで指定されたタグよりも長い場合、結果の文字列は利用できる長さに制限されます。

命令の処理中にエラーが発生し、OUT 出力パラメータが書き込める場合、空の文字列が出力されます。

パラメータ

以下の表に、「CONCAT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	STRING, WSTRING	D、L、または定数	文字列
IN2	Input	STRING, WSTRING	D、L、または定数	文字列
OUT	Return	STRING, WSTRING	D、L	結果の文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

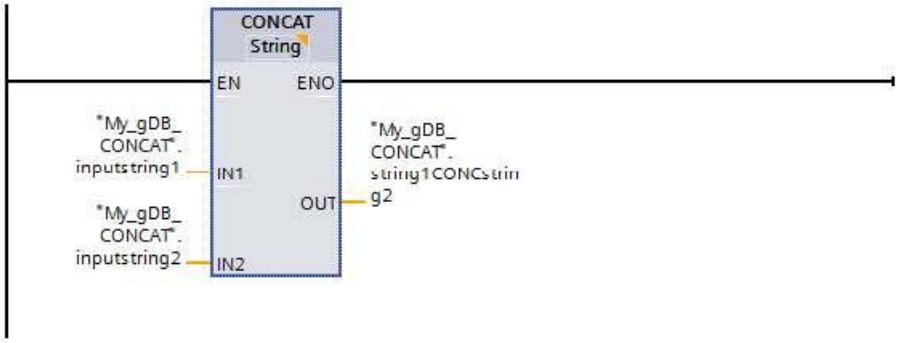
次の例では、STRING データタイプの 2 つの文字列を接続します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

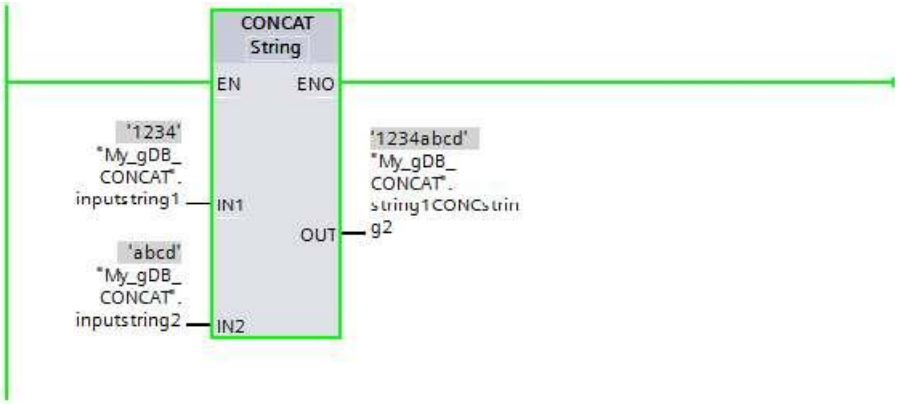
My_gDB_CONCAT

	Name	Data type	Start value
1	Static		
2	inputstring1	String	'1234'
3	inputstring2	String	'abcd'
4	string1CONCstring2	String	''

この命令のパラメータを以下のように相互接続します。



2 番目の文字列(「inputstring2」)の文字は、最初の文字列(「inputstring1」)に追加され、結果が出力パラメータ OUT(「string1CONCstring2」)に出力されます。



左: 文字列の左文字の読み出し

説明

「LEFT」命令を使用し、IN 入力パラメータの最初の文字から始まる部分的な文字列を抽出します。 抽出する文字数は、L パラメータで指定します。 抽出された文字は、(W)STRING フォーマットで OUT 出力パラメータで出力されます。

抽出する文字数が文字列の現在の長さを超える場合、OUT 出力パラメータが入力文字列を結果として返します。 L パラメータに値「0」が含まれる、または入力値が空の文字列の場合、空の文字列が返されます。 L パラメータの値が負の場合、空の文字列が出力されます。

命令の処理中にエラーが発生し、OUT 出力パラメータが書き込める場合、空の文字列が出力されます。

パラメータ

以下の表に、「LEFT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	文字列
L	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	抽出される文字の数
OUT	Return	STRING, WSTRING	D、L	抽出された部分文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

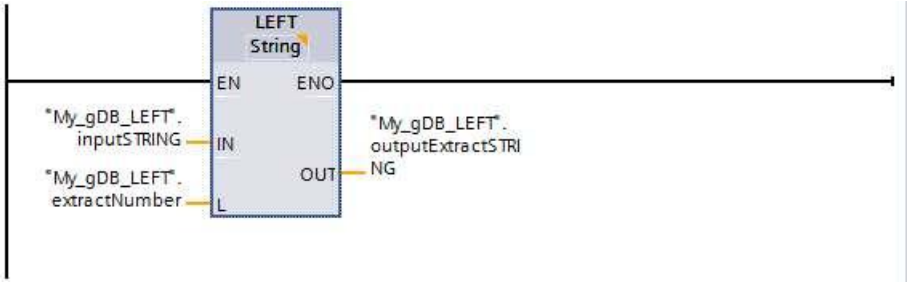
次の例では、文字列の最初の文字から STRING データタイプの部分文字列を抽出します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

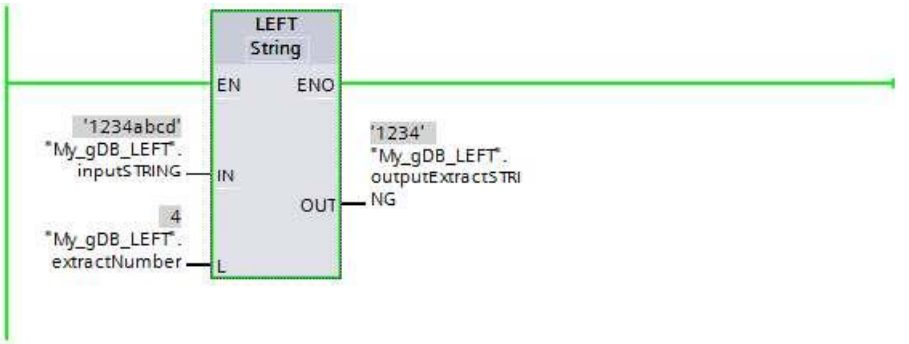
My_gDB_LEFT

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	extractNumber	Int	4
4	outputExtractSTRING	String	''

この命令のパラメータを以下のように相互接続します。



パラメータ L(「extractNumber」)の値「4」に従って、文字列(「inputSTRING」)の左側の最初の文字から始まる 4 文字の部分文字列を抽出します。 抽出された部分文字列は、出力パラメータ OUT(「outputExtractSTRING」)に出力されます。



右: 文字列の右文字の読み出し

説明

この命令を使用して、入力パラメータ IN の文字列の最後の L 文字を抽出します。 抽出する文字数は、L パラメータで指定します。 抽出された文字は、(W)STRING フォーマットで OUT 出力パラメータで出力されます。

抽出する文字数が文字列の現在の長さを超える場合、OUT 出力パラメータが入力文字列を結果として返します。 L パラメータに値「0」が含まれる、または入力値が空の文字列の場合、空の文字列が返されます。 L パラメータの値が負の場合、空の文字列が出力されます。

命令の処理中にエラーが発生し、OUT 出力パラメータが書き込める場合、空の文字列が出力されます。

パラメータ

以下の表に、「RIGHT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	文字列
L	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	抽出される文字の数
OUT	Return	STRING, WSTRING	D、L	抽出された部分文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

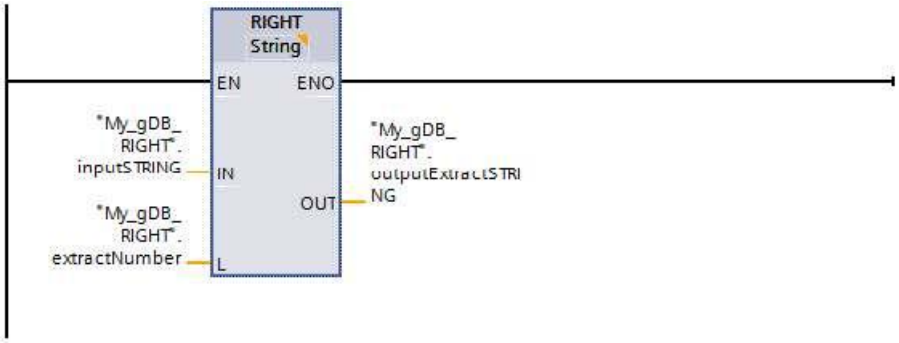
次の例では、文字列の最後の文字から STRING データタイプの部分文字列を抽出します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

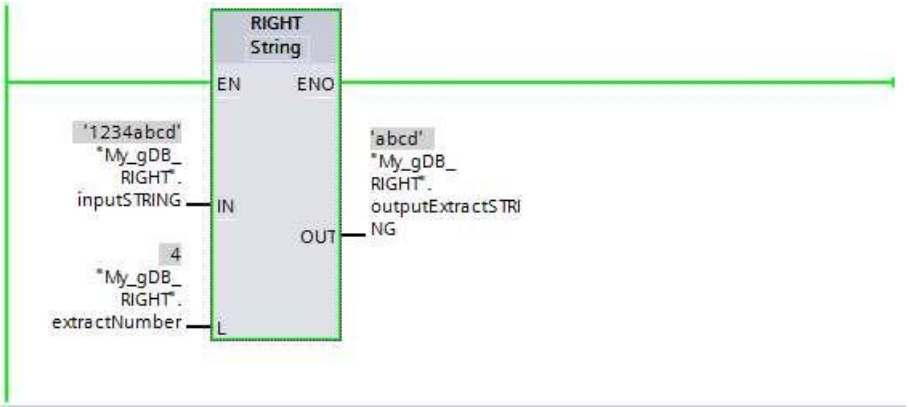
My_gDB_RIGHT

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	extractNumber	Int	4
4	outputExtractSTRING	String	''

この命令のパラメータを以下のように相互接続します。



パラメータ L(「extractNumber」)の値「4」に従って、文字列(「inputSTRING」)の右側の最初の文字から始まる 4 文字の部分文字列を抽出します。 抽出された部分文字列は、出力パラメータ OUT(「outputExtractSTRING」)に出力されます。



MID: 文字列の中央の文字の読み出し

説明

この命令を使用して、IN 入力パラメータの文字列の一部を抽出します。 P パラメータで、抽出する最初の文字の位置を指定します。 L パラメータで、抽出する文字列の長さを定義します。 抽出された部分的な文字列は、OUT 出力パラメータに出力されます。

この命令の実行時に、次のルールを守る必要があります。

- 抽出する文字数が、IN 入力パラメータの文字列の現在の長さを超えた場合、文字の位置 P から開始し、文字列の最後まで継続する部分的な文字列が出力されます。
- P パラメータで指定された文字の位置が IN 入力パラメータの現在の文字列の範囲外になる場合、OUT 出力パラメータで空の文字列が出力されます。
- P または L パラメータの値がゼロと等しいか、または負の場合、OUT 出力パラメータで空の文字列が出力されます。

命令の処理中にエラーが発生し、OUT 出力パラメータが書き込める場合、空の文字列が出力されます。

パラメータ

以下の表に、「MID」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	文字列
L	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	抽出される文字列の長さ
P	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	抽出される最初の文字の位置(最初の文字 = 1)
OUT	Return	STRING, WSTRING	D、L	抽出された部分文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

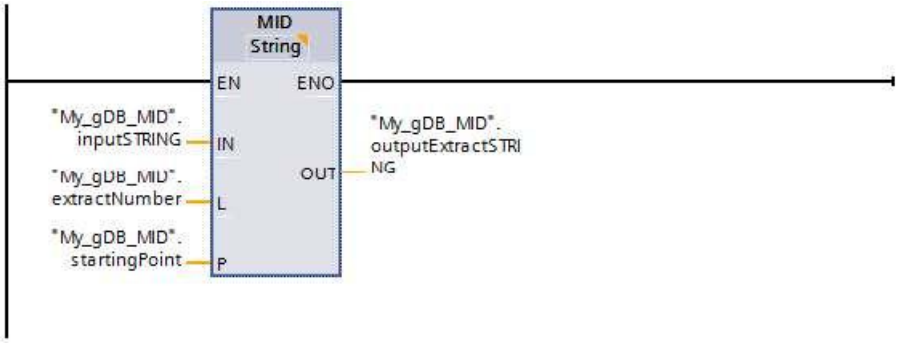
次の例では、文字列の中央から STRING データタイプの部分文字列を抽出します。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

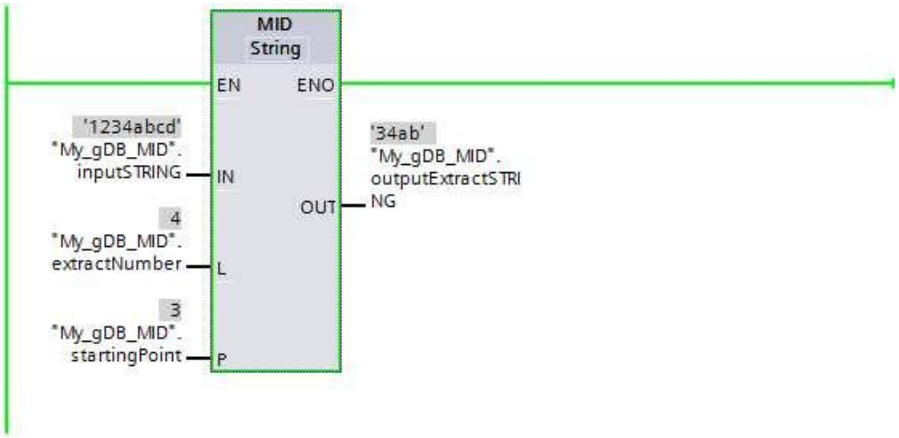
My_gDB_MID

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	startingPoint	Int	3
4	extractNumber	Int	4
5	outputExtractSTRING	String	''

この命令のパラメータを以下のように相互接続します。



パラメータ L(「extractNumber」)の値「4」に従って、文字列(「inputSTRING」)から 4 文字の部分文字列を抽出します。 抽出は文字列(「inputSTRING」)の 3 番目の文字(「startingPoint」は値「3」を持つ) から始まります。 抽出された部分文字列は、出力パラメータ OUT(「outputExtractSTRING」)に出力されます。



削除: 文字列の文字の削除

説明

この命令を使用して、IN 入力パラメータの文字列の一部を削除します。 P パラメータで、削除する最初の文字の位置を指定します。 削除する文字数は、L パラメータで指定します。 残りの部分的な文字列は、(W)STRING フォーマットで OUT 出力パラメータに出力されます。

この命令の実行時に、次のルールを守る必要があります。

- P パラメータの値がゼロ以下の場合、空の文字列が OUT 出力パラメータで出力されます。
- P パラメータの値が、IN 入力の文字列の現在の長さを超える場合、OUT 出力パラメータで入力文字列が返されます。
- L パラメータの値がゼロと等しい場合、OUT 出力パラメータで入力文字列が返されます。
- L パラメータで削除する文字数が IN 入力パラメータの文字列の長さより大きい場合、P パラメータで指定された位置から始まる文字は削除されます。 この結果の文字列が出力されます。
- L パラメータの値が負の場合、空の文字列が出力されます。

命令の処理中にエラーが発生し、OUT 出力パラメータが書き込める場合、空の文字列が出力されます。

パラメータ

以下の表に、「DELETE」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN	Input	STRING, WSTRING	D、L、または定数	文字列
L	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	削除される文字数
P	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	削除される最初の文字の位置
OUT	Return	STRING, WSTRING	D、L	結果の文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

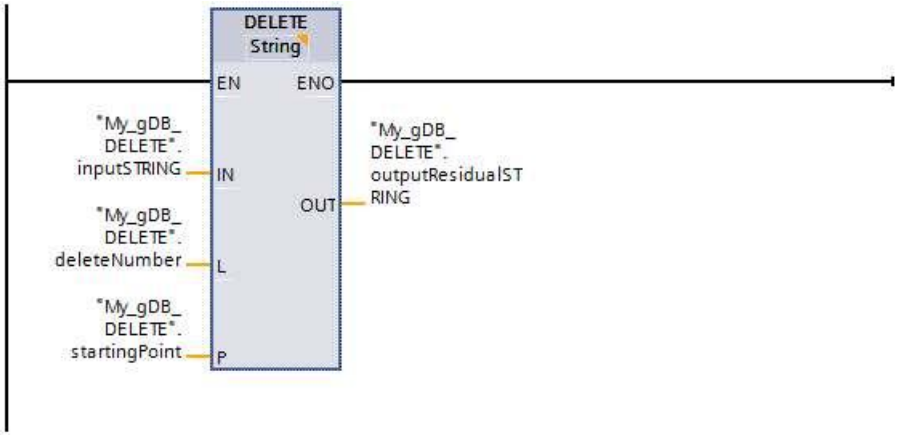
次の例では、STRING データタイプの文字列から文字を削除します。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

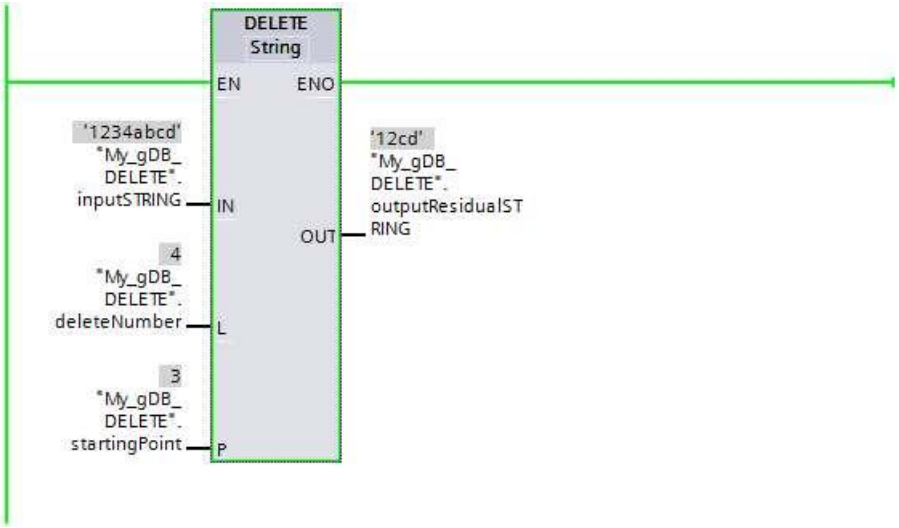
My_gDB_DELETE

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	startingPoint	Int	3
4	deleteNumber	Int	4
5	outputResidualSTRING	String	''

この命令のパラメータを以下のように相互接続します。



入力パラメータ L(「deleteNumber」)の値「4」に従って、文字列(「inputSTRING」)の 3 番目の文字(「startingPoint」は値「3」を持つ)から 4 文字を削除します。 残りの文字列は、出力パラメータ OUT(「outputResidualSTRING」)に出力されます。



INSERT: 文字列への文字の挿入

説明

この命令を使用して、IN2 入力パラメータの文字列を IN1 入力パラメータの文字列に挿入します。 P パラメータで、文字が挿入される文字列の位置を指定します。 結果は、(W)STRING フォーマットで OUT 出力パラメータに出力されます。

この命令の実行時に、次のルールを守る必要があります。

- P パラメータの値が IN1 入力パラメータの文字列の現在の長さを超える場合、IN2 入力パラメータの文字列は IN1 入力パラメータの文字列に追加されます。
- P パラメータの値がゼロの場合、IN2 パラメータの文字列と、それに続いて IN1 パラメータの文字列が、OUT 出力パラメータで出力されます。
- P パラメータの値が負の場合、OUT 出力パラメータで空の文字列が出力されます。
- 結果の文字列が、OUT 出力パラメータで指定されたタグよりも長い場合、結果の文字列は利用できる長さに制限されます。

パラメータ

以下の表に、「INSERT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	STRING, WSTRING	D、L、または定数	文字列
IN2	Input	STRING, WSTRING	D、L、または定数	挿入する文字列
P	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	挿入位置
OUT	Return	STRING, WSTRING	D、L	結果の文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

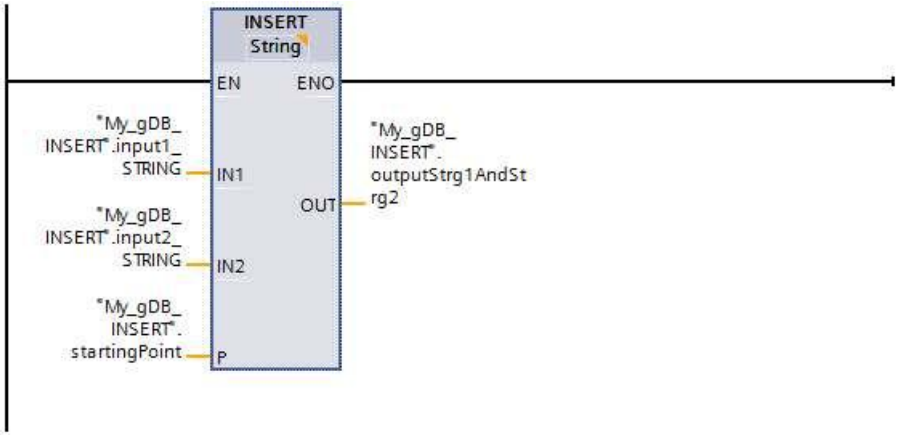
次の例では、文字列を別の文字列に挿入します。 使用されるデータタイプは STRING です。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

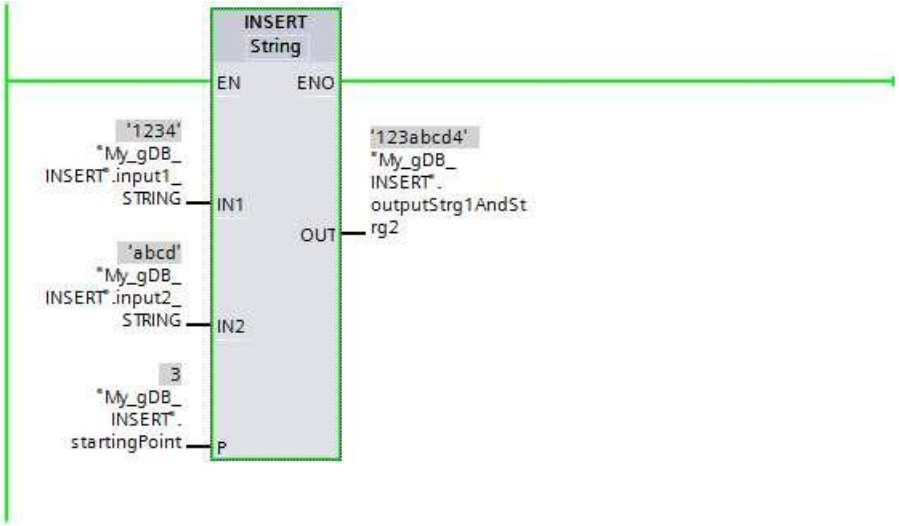
My_gDB_INSERT

	Name	Data type	Start value
1	Static		
2	input1_STRING	String	'1234'
3	input2_STRING	String	'abcd'
4	startingPoint	Int	3
5	outputStrg1AndStrg2	String	''

この命令のパラメータを以下のように相互接続します。



2 番目の文字列(「input2_STRING」)の文字が、最初の文字列(「input1_STRING」)に挿入されます。 パラメータ P(「startingPoint」)の値「3」に従って、最初の文字列(「input1_STRING」)の 3 番目の文字の後に文字が挿入されます。 結果は、出力パラメータ OUT(「outputStrg1AndStrg2」)に出力されます。



REPLACE: 文字列の文字置換

説明

この命令を使用して、IN1 入力の文字列の一部を IN2 入力の文字列に置換します。 P パラメータで、置換する最初の文字の位置を指定します。 置換する文字数は、L パラメータで指定します。 結果は、(W)STRING フォーマットで OUT 出力パラメータに出力されます。

この命令の実行時に、次のルールを守る必要があります。

- P パラメータの値がゼロ以下の場合、空の文字列が OUT 出力パラメータで出力されます。
- L パラメータの値がゼロ未満の場合、OUT 出力パラメータで空の文字列が出力されます。
- P が 1 と等しい場合、IN1 入力の文字列が最初の文字から (最初の文字を含む) 置換されます。
- P パラメータの値が IN1 入力パラメータの文字列の現在の長さを超える場合、IN2 入力パラメータの文字列は IN1 入力パラメータの文字列に追加されます。
- 結果の文字列が、OUT 出力パラメータで指定されたタグよりも長い場合、結果の文字列は利用できる長さに制限されます。
- パラメータ L の値がゼロの場合、文字は置換されずに挿入されます。 命令 INSERT を使用すると、類似の条件が適用されます。 関連項目: INSERT: 文字列への文字の挿入

パラメータ

以下の表に、「REPLACE」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	STRING, WSTRING	D、L、または定数	置換される文字のある文字列
IN2	Input	STRING, WSTRING	D、L、または定数	挿入される文字のある文字列
L	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	置換される文字数
P	Input	BYTE, INT, SINT, USINT	I、Q、M、D、L、または定数	置換される最初の文字の位置
OUT	Return	STRING, WSTRING	D、L	結果の文字列

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

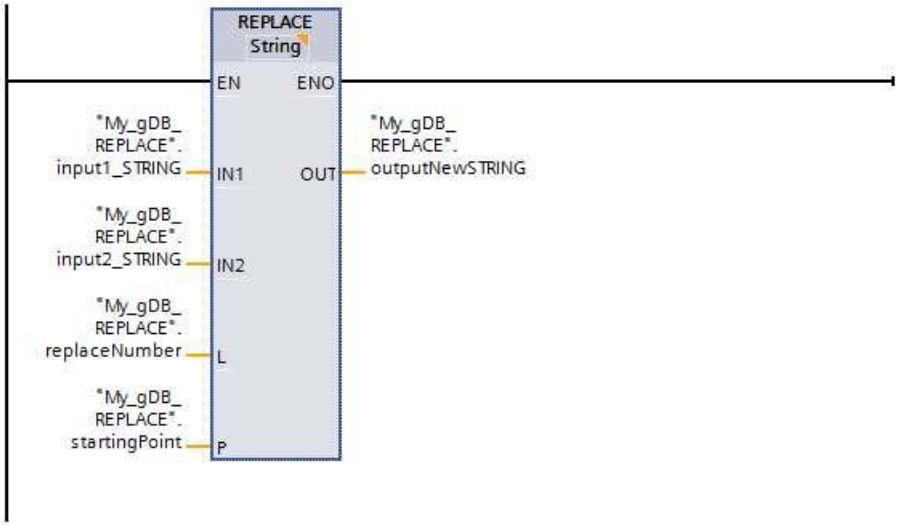
次の例では、文字列の一部を別の文字列で置換します。 使用されるデータタイプは STRING です。

グローバルデータブロックにデータを保存するために、5 つのタグを作成します。

My_gDB_REPLACE

	Name	Data type	Start value
1	Static		
2	input1_STRING	String	'1234'
3	input2_STRING	String	'abcd'
4	replaceNumber	Int	2
5	startingPoint	Int	3
6	outputNewSTRING	String	''

この命令のパラメータを以下のように相互接続します。



2 番目の文字列(「input2_STRING」)が最初の文字列(「input1_STRING」)の 3 番目の文字(「startingPoint」は値「3」を持つ)の後に追加されます。 パラメータ L(「replaceNumber」)の値「2」に従って、最初の文字列(「input1_STRING」)の 3 番目と 4 番目の文字がこのプロセスで置換されます。 結果は、出力パラメータ OUT(「outputNewSTRING」)に出力されます。

My_gDB_REPLACE

	Name	Data type	Start value	Monitor value
1	Static			
2	input1_STRING	String	'1234'	'1234'
3	input2_STRING	String	'abcd'	'abcd'
4	replaceNumber	Int	2	2
5	startingPoint	Int	3	3
6	outputNewSTRING	String	''	'12abcd'

FIND: 文字列から文字を検索

説明

この命令を使用して、IN1 入力パラメータの文字列で、特定の文字または特定の文字列を検索できます。

- IN2 入力パラメータで、検索対象の値を指定します。 検索は左から右の順に行われます。
- 最初に見つかった一致の位置が OUT 出力パラメータで出力されます。 検索で一致が返されない場合、OUT 出力パラメータで値「0」が出力されます。

IN2 パラメータで無効な文字が指定された場合や、処理中にエラーが発生した場合は、OUT パラメータで値「0」が出力されます。

パラメータ

以下の表に、「FIND」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IN1	Input	STRING, WSTRING	D、L、または定数	検索される文字列
IN2	Input	STRING, WSTRING, CHAR, WCHAR	D、L、または定数 ((W)CHAR の場合、I、Q、M も)	検索対象文字
OUT	Return	INT	I、Q、M、D、L	文字位置

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

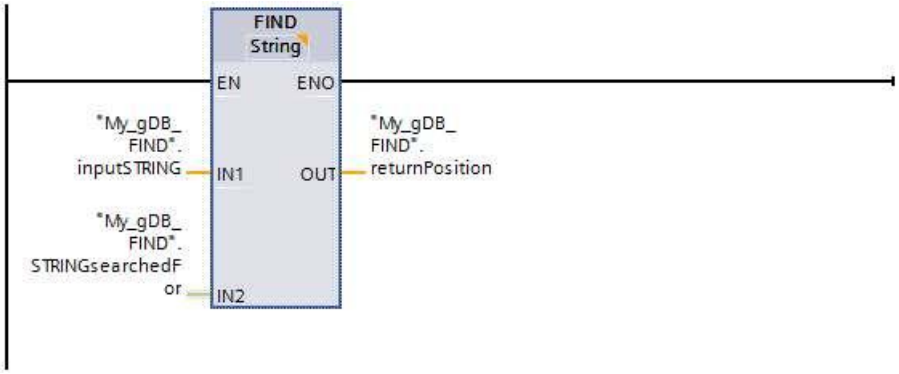
次の例では、文字列を別の文字列内で検索します。 使用されるデータタイプは STRING です。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

My_gDB_FIND

	Name	Data type	Start value
1	Static		
2	inputSTRING	String	'1234abcd'
3	STRINGsearchedFor	String	'4a'
4	returnPosition	Int	0

この命令のパラメータを以下のように相互接続します。



最初の文字列 (「inputSTRING」)は、2 番目の文字列(「STRINGsearchedFor」)の値「4a」を検索します。 検索する文字列が始まる文字の位置(「4」)は、出力パラメータ OUT(「returnPosition」)に出力されます。

My_gDB_FIND

	Name	Data type	Start value	Monitor value
1	Static			
2	inputSTRING	String	'1234abcd'	'1234abcd'
3	STRINGsearchedFor	String	'4a'	'4a'
4	returnPosition	Int	0	4

ランタイム情報

この章には下記に関する情報が記載されています：

- GetSymbolName: 入力パラメータのタグの読み出し (S7-1500)
- GetSymbolPath:入力パラメータ割り当てのクエリ複合グローバル名 (S7-1500)
- GetInstanceName: ブロックインスタンスの名前の読み出し (S7-1500)
- GetInstancePath:ブロックインスタンスのクエリ複合グローバル名 (S7-1500)
- GetBlockName: ブロックの名前の読み出し (S7-1500)

GetSymbolName: 入力パラメータのタグの読み出し

説明

「GetSymbolName」命令を使用して、ブロックの入力パラメータによってパラメータで相互接続されたタグの名前を読み出します。

ブロックがプロジェクトで複数回使用される場合で、かつ毎回異なるタグによって呼び出される場合は、「GetSymbolName」命令を使用して呼び出しタグの名前を評価できます。 この場合、タグのプロセス値は無関係です。

- VARIABLE パラメータでブロックインターフェースの入力パラメータを指定します。 このパラメータにはインターフェースパラメータのみを使用し、PLC タグやデータブロックタグは使用しないでください。
- SIZE パラメータを使用して、読み出されたタグ名の長さを制限できます。 名前が切り捨てられた場合、このことは、名前の最後に表示される文字「...」(Unicode 文字 16#2026)によって示されます。 「...」文字自体の長さは 1 であることに注意してください。たとえば、SIZE パラメータで最大長として 6 を入力した場合で、かつブロックインターフェースによってタグ名「MyPLCTag」を読み出す場合は、以下が出力されます。 "MyPLC...".
- 読み出された名前は、OUT パラメータで出力されます。

パラメータ

以下の表に、「GetSymbolName」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
VARIABLE	Input	PARAME-TER	L	ブロックインターフェースの入力パラメータ
SIZE	Input	DINT	I、Q、M、D、L	読み出された名前の長さ制限
OUT	Return	WSTRING	I、Q、M、D、L	ブロックインターフェースを経由して読み出されたタグの名前

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

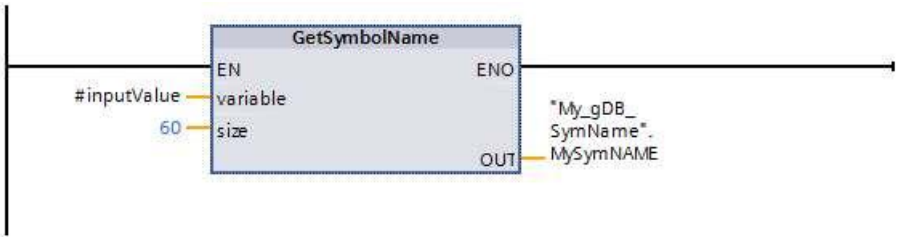
次の例では、ブロックの入力パラメータによって相互接続されたタグの名前を読み出します。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

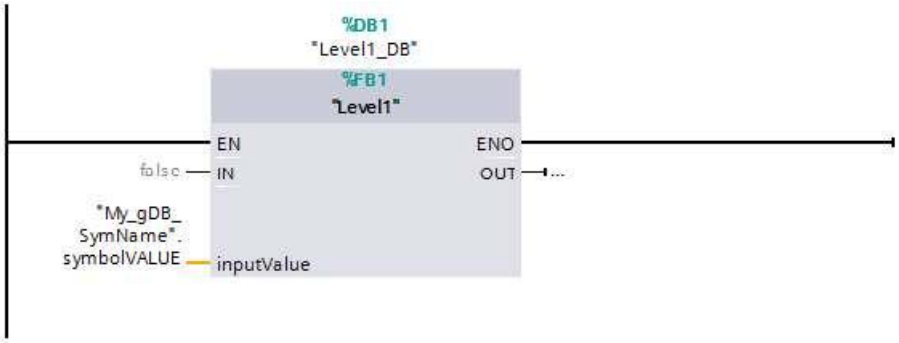
My_gDB_SymName

	Name	Data type	Start value
1	Static		
2	MySymNAME	WString	WSTRING#''
3	symbolVALUE	Byte	16#42

BYTE データタイプの入力パラメータ inputValue を「Level1」ブロックに作成します。 「Level1」ブロック内で「GetSymbolName」命令を呼び出します。 この命令のパラメータを以下のように相互接続します。



「Level1」ブロックの inputValue パラメータを次のように相互接続します。



「Level1」ブロック内で「GetSymbolName」命令が実行されます。 「Level1」ブロックの入力パラメータ inputValue は、この命令の入力パラメータ VARIABLE を使用して相互接続を検証します。 これを行うことにより、「symbolVALUE」タグが読み出され、出力パラメータ OUT(「MySymNAME」)に文字列として出力されます。 入力パラメータ SIZE の値に従って、文字列の長さは 60 文字に制限されます。

My_gDB_SymName

	Name	Data type	Start value	Monitor value
1	Static			
2	MySymNAME	WString	WSTRING#''	WSTRING#'"My_gDB_SymName".symbolVALUE'
3	symbolVALUE	Byte	16#42	16#42

GetSymbolPath:入力パラメータ割り当てのクエリ複合グローバル名

説明

「GetSymbolPath」命令を使用して、ローカルインターフェースで入力パラメータの複合グローバル名を読み出すことができます。 この名前は保存パスとタグ名で構成されています。

- この命令の VARIABLE パラメータで読み出す入力タグの名前を通じてブロックインターフェースを指定します。
  - 入力パラメータを提供するためにデータブロックのタグが使用される場合、DB の名前、格納された構造体およびタグの名前がパスとして出力されます。
  - 入力パラメータを提供するために PLC タグが使用される場合、PLC タグの名前が出力されます。
  - 入力パラメータを提供するために定数が使用される場合、定数の内容が出力されます。
- SIZE パラメータを使用して、読み出されたタグ名の長さを制限できます。 名前が切り捨てられた場合、このことは、名前の最後に表示される文字「...」(Unicode 文字 16#2026)によって示されます。 「...」文字自体が長さ 1 であることに注意してください。

  たとえば、SIZE パラメータで最大長として 6 を入力した場合で、かつブロックインターフェースを経由してタグ名「MyPLCTag」を読み出す場合は、以下が出力されます。「MyPLC...」

パラメータ

以下の表に、「GetSymbolPath」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
VARIABLE	Input	VARIANT	L	入力パラメータ供給のグローバル名を読み出すローカルインターフェースの選択。
SIZE	Input	DINT	I、Q、M、D、L、または定数	OUT パラメータで出力される文字数を制限します。 文字の出力を制限しない場合は、SIZE パラメータに「0」を入力します。
OUT	Output	WSTRING	I、Q、M、D、L	入力パラメータ供給のタグ名の出力。

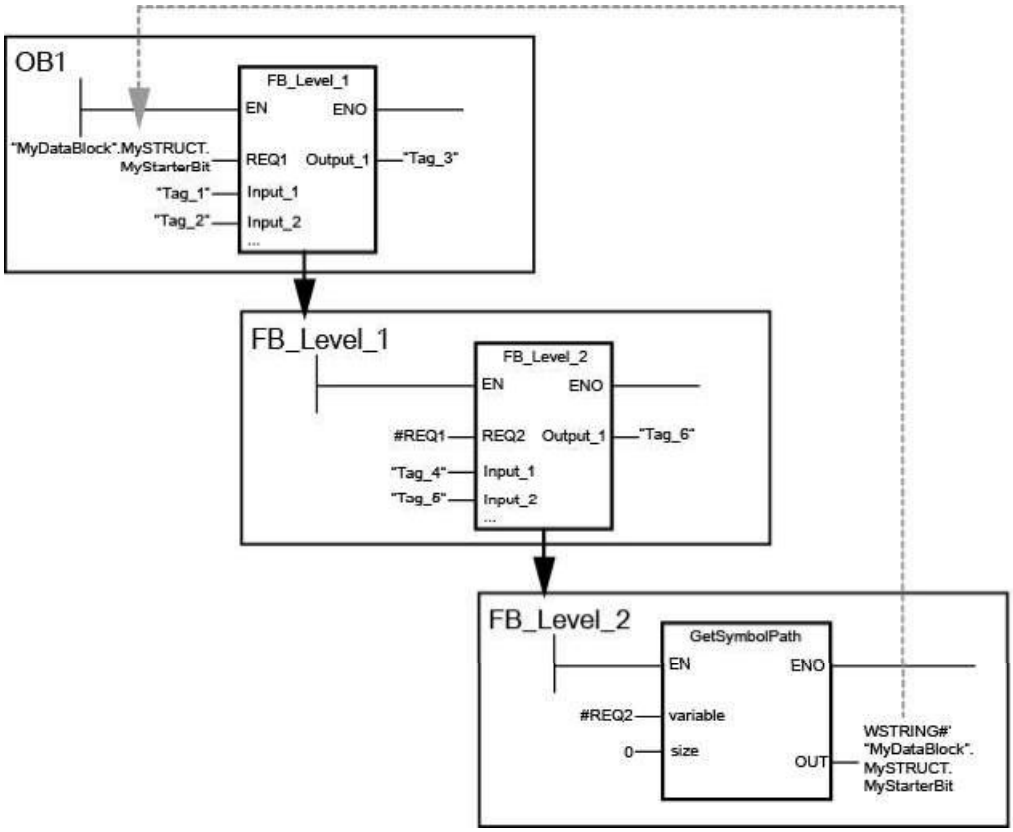
有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

次の例は、複数の呼び出しレベルでの GetSymbolPath の使用を示しています。

- FB_Level_1 ブロックはオーガニゼーションブロック OB1 から呼び出され、今度はそこから FB_Level_2 ブロックが呼び出されます。
- GetSymbolPath が FB_Level_2 ブロックで実行され、REQ2 インターフェースでパラメータのパスを読み出します。
- 今度は REQ2 が REQ1 インターフェースによって提供され、この命令が REQ1 の入力パラメータのパスを判別します。
- MyStarterBit が REQ1 パラメータでパラメータとして使用されます。 このビットは、MyDatablock データブロックの MySTRUCT 構造体にあります。

この情報は、GetSymbolPath によって読み出され、OUT パラメータに出力されます。



GetInstanceName: ブロックインスタンスの名前の読み出し

説明

「GetInstanceName」命令を使用して、ファンクションブロック内のインスタンスデータブロックの名前を読み出すことができます。

- SIZE パラメータを使用して、読み出すインスタンス名の文字数を指定できます。 SIZE に「0」値を使用した場合、名前全体が読み出されます。
- インスタンスデータブロックの名前は OUT パラメータに書き込まれます。 インスタンスデータブロックの名前が最大長(WSTRING)よりも長い場合、名前は切り捨てられます。

パラメータ

以下の表に、「GetInstanceName」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SIZE	Input	DINT	I、Q、M、D、L、または定数	インスタンスデータブロックの名前を読み出す際の最大文字数
OUT	Output	WSTRING	I、Q、M、D、L	インスタンスデータブロックの読み出された名前

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

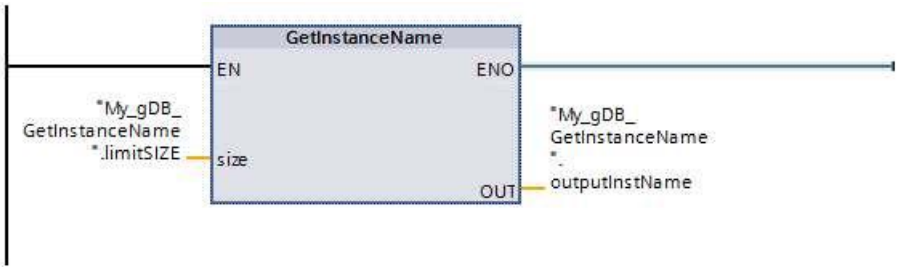
次の例では、インスタンスデータブロックの名前を読み出します。

グローバルデータブロックにデータを保存するために、2 つのタグを作成します。

My_gDB_GetInstanceName

	Name	Data type	Start value
1	Static		
2	limitSIZE	DInt	0
3	outputInstName	WString	WSTRING#''

この命令のパラメータを以下のように相互接続します。



「Level1_gin」ブロック内で「GetInstanceName」命令が実行されます。 「GetInstanceName」命令を使用して、「Level1_gin」ブロックの関連するインスタンスデータブロックを判別し、出力パラメー

タ OUT(「outputInstName」)に文字列として出力します。 パラメータ SIZE(「limitSIZE」)の値「0」に従って、文字列の長さに制限はありません。

My_gDB_GetInstanceName

	Name	Data type	Start value	Monitor value
1	Static			
2	limitSIZE	DInt	0	0
3	outputInstName	WString	WSTRING#''	WSTRING#'"Level1_DB"'

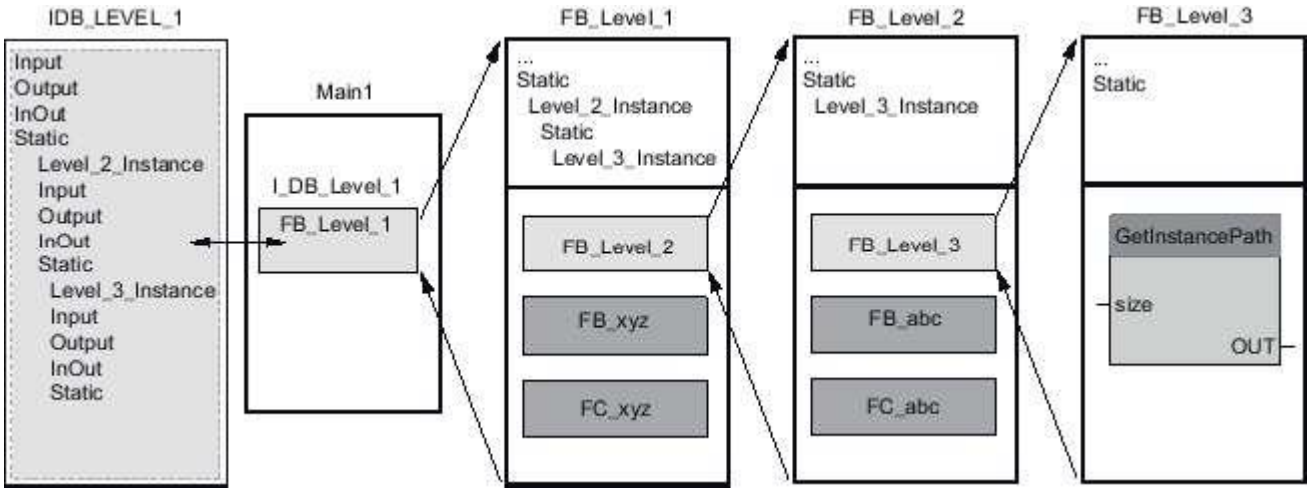
GetInstancePath:ブロックインスタンスのクエリ複合グローバル名

説明

「GetInstancePath」命令を使用して、ファンクションブロック内のブロックインスタンスの複合グローバル名を読み出すことができます。 ブロックインスタンスの複合グローバル名は、マルチインスタンスが使用される場合の階層呼び出し全体のパスを示しています。

次の例では、「GetInstancePath」命令を FB_Level_3 ファンクションブロックで呼び出します。

- FB_Level_3 ファンクションブロックは、呼び出し FB_Level_2 ファンクションブロックにデータを保存します。
- 同様に FB_Level_2 ファンクションブロックは、呼び出し FB_Level_1 ファンクションブロックにデータを保存します。
- 同様に FB_Level_1 ファンクションブロックは、インスタンスデータブロック IDB_LEVEL_1 にデータを保存します。 マルチインスタンスの使用により、FB_Level_1 のインスタンスデータブロックには 3 つのファンクションブロックのすべてのデータが含まれています。



「GetInstancePath」命令は、この例に以下のパスを返します。 '"IDB_LEVEL_1".Level_2_Instance.Level_3_Instance'

注記

シングルインスタンスのファンクションブロックでの「GetInstancePath」の使用

「GetInstancePath」を呼び出すファンクションブロックが自身のインスタンスデータブロックにデータを保存している場合、シングルインスタンスの名前がグローバル名として出力されます。 この場合、パラメータ OUT の結果は、「GetInstanceName」命令と一致します。

パラメータ

以下の表に、「GetInstancePath」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SIZE	Input	DINT	I、Q、M、D、L、または定数	ブロックインスタンスのグローバル名を読み出す際の最大文字数。 SIZE = 0 の場合、グローバル名全体が出力されます。
OUT	Output	WSTRING	I、Q、M、D、L	ブロックインスタンスのグローバル名を読み出します。 ブロックインスタンスのグローバル名が WSTRING の最大長(254 文字)よりも長い場合、名前は切り捨てられます。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

GetBlockName: ブロックの名前の読み出し

説明

「GetBlockName」命令を使用して、命令が呼び出されるブロックの名前を読み出すことができます。

- SIZE パラメータを使用して、読み出すブロック名の文字数を指定できます。 SIZE に「0」値を使用した場合、名前全体が読み出されます。
- ブロックの名前は OUT パラメータに書き込まれます。 ブロックの名前が最大長(WSTRING)よりも長い場合、名前は切り捨てられます。

パラメータ

以下の表に、「GetBlockName」命令のパラメータを示します。

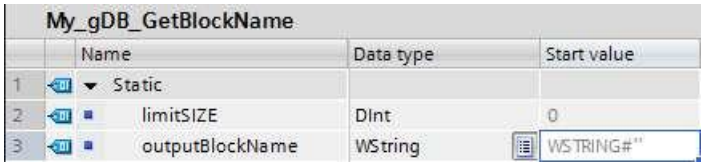
パラメータ	宣言	データタイプ	メモリ領域	説明
SIZE	Input	UINT	I、Q、M、D、L、または定数	インスタンスデータブロックの名前を読み出す際の最大文字数。
RET_VAL	Output	WSTRING	I、Q、M、D、L	インスタンスデータブロックの読み出された名前

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

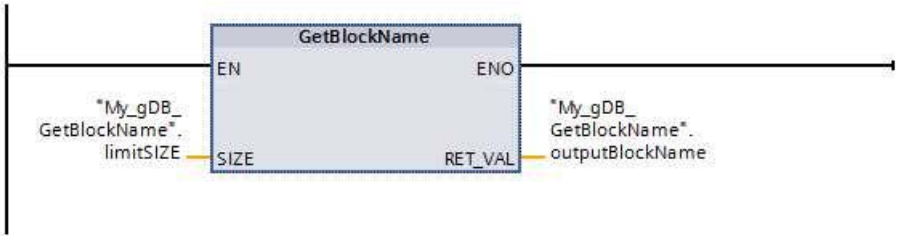
例

次の例では、ブロック名を読み出します。

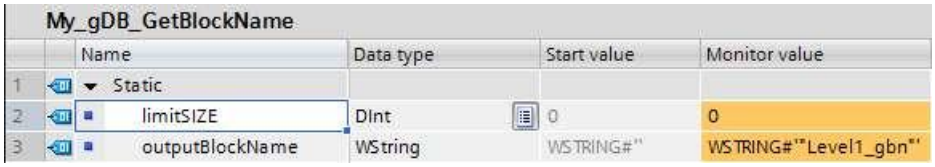
グローバルデータブロックにデータを保存するために、2 つのタグを作成します。



この命令のパラメータを以下のように相互接続します。



「Level1_gbn」ブロック内で「GetBlockName」命令が実行されます。 「GetBlockName」命令を使用して、「Level1_gbn」ブロックの名前を読み出し、出力パラメータ OUT(「outputBlockName」)に文字列として出力します。 パラメータ SIZE(「limitSIZE」)の値「0」に従って、文字列の長さに制限はありません。



My_gDB_GetBlockName

	Name	Data type	Start value	Monitor value
1	Static			
2	limitSIZE	DInt	0	0
3	outputBlockName	WString	WSTRING#''	WSTRING#'"Level1_gbn"'

プロセスイメージ

この章には下記に関する情報が記載されています：

- UPDAT_PI: プロセスイメージ入力の更新 (S7-1500)
- UPDAT_PO: プロセスイメージ出力の更新 (S7-1500)
- SYNC_PI: プロセスイメージ入力の同期 (S7-1500)
- SYNC_PO: プロセスイメージ出力の同期 (S7-1500)

UPDAT_PI: プロセスイメージ入力の更新

説明

この命令を使用して、入力の OB1 プロセスイメージ(= プロセスイメージパーティション 0)、または設定で定義された入力のプロセスイメージパーティションを更新します。

システム側のプロセスイメージ更新に I/O アクセスエラーの繰り返し通知を設定した場合、選択されたプロセスイメージが常に更新されます。

これ以外の場合、この更新は選択されたプロセスイメージパーティションがシステムによって更新されない場合のみ実行されます。

- このプロセスイメージパーティションを割り込み OB に割り当てていない場合、

  または

- 設定中にプロセスイメージパーティション 0 を選択し、OB 1 プロセスイメージパーティションの更新を無効にした場合。

> **注記**
> 設定ごとに入力のプロセスイメージパーティションに割り当てた各論理アドレスは、入力の OB 1 プロセスイメージに属さなくなります。
> 「UPDAT_PI」でプロセスイメージパーティションを更新する場合、「SYNC_PI」命令による同時更新を実行しないでください。

入力の OB 1 プロセスイメージおよび割り込み OB に割り当てた入力のプロセスイメージパーティションのシステム側の更新は、「UPDAT_PI」呼び出しとは独立して行われます。

パラメータ

次の表に、「UPDAT_PI」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PART	Input	PIP	I、Q、M、D、L、または定数	更新する入力のプロセスイメージパーティションの番号。 最大値範囲(CPU によって異なる): 0～31 (0 は OB 1 プロセスイメージを意味し、1 <= n <= 31 の n はプロセスイメージパーティション n を意味します)。
RET_VAL	Return	INT	I、Q、M、D、L、P	エラー情報
FLADDR	Output	WORD	I、Q、M、D、L、P	アクセスエラーが発生した場合にエラーの原因となった最初のバイトのアドレス。

有効なデータタイプに関する追加情報は、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	PART パラメータの無効な値
8091	指定されたプロセスイメージパーティションがまだ定義されていない、または CPU の許可されたプロセスイメージ領域にない。
8092	プロセスイメージパーティションがシステムによって更新されており、PART パラメータに使用することができません。
8093	他の OB でプロセスイメージパーティションの更新が処理されています。
80A0	I/O にアクセスしたときアクセスエラーが検出されました。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

注記

32 バイトを超える整合性領域を定義した DP 標準スレーブのプロセスイメージパーティションにこの命令を使用すると、「DPRD_DAT」命令のエラーコードも使用できます。

UPDAT_PO: プロセスイメージ出力の更新

説明

この命令を使用して、出力の OB 1 プロセスイメージ(= プロセスイメージパーティション 0)、または設定ごとに定義された出力のプロセスイメージパーティションの信号状態を出力モジュールに転送します。

選択したプロセスイメージパーティションに整合性範囲を指定している場合、対応するデータはそれぞれの I/O モジュールに整合性データとして転送されます。

注記
設定中に出力のプロセスイメージパーティションに割り当てた各論理アドレスは、出力の OB 1 プロセスイメージに属さなくなります。
「UPDAT_PO」で更新する出力は、「SYNC_PO」命令で同時に更新しないでください。

出力の OB 1 プロセスイメージおよび割り込み OB に割り当てた出力のプロセスイメージパーティションは、「UPDAT_PO」呼び出しとは関係なくシステムによって出力モジュールに転送されます。

パラメータ

次の表に、「UPDAT_PO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PART	Input	PIP	I、Q、M、D、L、または定数	転送する出力のプロセスイメージパーティションの番号 最大値範囲(CPU によって異なる): 0 ～31 (0 は OB 1 プロセスイメージ、1 <= n <= 31 の n はプロセスイメージパーティション n を意味します)
RET_VAL	Return	INT	I、Q、M、D、L、P	エラー情報
FLADDR	Output	WORD	I、Q、M、D、L、P	アクセスエラーが発生した場合にエラーの原因となった最初のバイトのアドレス。

有効なデータタイプに関する追加情報は、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	PART パラメータの無効な値
8091	指定されたプロセスイメージパーティションがまだ定義されていない、または CPU の許可されたプロセスイメージ領域にない。

8092	プロセスイメージパーティションがシステムによって更新されており、PART パラメータに使用することができません。
8093	他の OB でプロセスイメージパーティションの更新が処理されています。
80A0	I/O にアクセスしたときアクセスエラーが検出されました。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

注記
32 バイトを超える整合性領域を定義した DP 標準スレーブのプロセスイメージパーティションにこの命令を使用すると、「DPWR_DAT」命令のエラーコードも使用できます。

SYNC_PI: プロセスイメージ入力の同期

説明

「SYNC_PI」を使用して、アイソクロナスモードで入力のプロセスイメージパーティションを更新します。 DP サイクルまたは PN 送信サイクルにリンクされたユーザープログラムは、この命令を使用して入力のプロセスイメージパーティションで取得した入力データをアイソクロナス方式で整合性を保って更新できます。

呼び出し

"「SYNC_PI」は割り込み可能で、OB 61、62、63、および 64 のみで呼び出すことができます。

注記

OB 61～64 での「SYNC_PI」命令の呼び出しは、HW コンフィグレーションで影響を受けたプロセスイメージパーティションを関連する OB に割り当てている場合のみ許可されます。

「SYNC_PI」で更新するプロセスイメージパーティションは、同時に「UPDAT_PI」命令で更新しないでください。

パラメータ

次の表に、「SYNC_PI」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	値の範囲	説明
PART	Input	PIP	I、Q、M、D、L、または定数	1～31	アイソクロナス方式で更新する入力のプロセスイメージパーティションの番号。
RET_VAL	Return	INT	I、Q、M、D、L、P	-	エラー情報
FLADDR	Output	WORD	I、Q、M、D、L、P		アクセスエラーが発生した場合にエラーの原因となった最初のバイトのアドレス。

有効なデータタイプに関する追加情報は、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード (W#16#...)*	説明
0000	エラーは発生しませんでした。
0001	整合性の警告。 プロセスイメージパーティションの更新が、2 つの DP または PN サイクルに分散されました。 ただし、1 つのスレーブまたは IO デバイスのデータは整合性を保って転送されました。

8090	PART パラメータの値が無効であるか、または指定された入力のプロセスイメージパーティションの更新がこの OB では許可されていません。入力のプロセスイメージパーティションが更新されませんでした。
8091	指定されたプロセスイメージパーティションがまだ定義されていない、または CPU の許可されたプロセスイメージ領域にない。 入力のプロセスイメージパーティションが更新されませんでした。
8092	プロセスイメージパーティションがシステムによって更新されており、PART パラメータに使用することができません。
8093	他の OB でプロセスイメージパーティションの更新が処理されています。
80A0	更新中に、アクセスエラーが検出されました。 影響を受けた入力は「0」にセットされました。
80A1	更新時間が許可されたアクセス時間の後になっています。 入力のプロセスイメージパーティションが更新されませんでした。 命令を処理するために十分な時間を確保するためには、DP または PN サイクルが短すぎます。 したがって、TDP (T_DC としても知られています)、Ti および To タイマを増加する必要があります。
80A2	整合性の警告付きのアクセスエラー 指定されたプロセスイメージパーティションの更新の場合、同時整合性警告付きのアクセスエラーが検出されました。 • 不正な入力のデータは I/O で読み出されませんでした。入力のプロセスイメージパーティションで、関連する入力がゼロに設定されます。 • アクセスエラー入力データに関連しないプロセスイメージパーティションの更新は、2 つの DP および PN サイクルに分散されました。
80C1	更新時間が許可されたアクセス時間の前になっています。 入力のプロセスイメージパーティションが更新されませんでした。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

注記

32 バイトを超える整合性領域を定義した DP 標準スレーブのプロセスイメージパーティションに「SYNC_PI 命令を使用すると、「DPRD_DAT」命令のエラーコードも使用できます。

SYNC_PO: プロセスイメージ出力の同期

説明

「SYNC_PO」命令を使用して、アイソクロナスモードで出力のプロセスイメージパーティションを更新します。 DP サイクルまたは PN 送信サイクルにリンクされたアプリケーションプログラムは、この命令を使用して、出力のプロセスイメージパーティションの計算された出力データを I/O デバイスに整合性保持アイソクロナス転送することができます。

呼び出し

"「SYNC_PO」は割り込み可能で、OB 61、62、63、および 64 のみで呼び出すことができます。

注記

OB 61～64 での「SYNC_PO」命令の呼び出しは、HW コンフィグレーションで影響を受けたプロセスイメージパーティションを関連する OB に割り当てている場合のみ許可されます。

「SYNC_PO」で更新するプロセスイメージパーティションは、同時に「UPDAT_PO」命令で更新しないでください。

パラメータ

次の表に、「SYNC_PO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	値の範囲	説明
PART	Input	PIP	I、Q、M、D、L、または定数	1～31	アイソクロナス方式で更新する出力のプロセスイメージパーティションの番号。
RET_VAL	Return	INT	I、Q、M、D、P	-	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
FLADDR	Output	WORD	I、Q、M、D、L、P	-	エラーの原因となった最初のバイトのアドレス。

有効なデータタイプに関する追加情報は、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード (W#16#...)*	説明
0000	エラーは発生しませんでした。
0001	整合性の警告。 プロセスイメージパーティションの更新が、2 つの DP または PN サイクルに分散されました。 ただし、1 つのスレーブまたは IO デバイスのデータは整合性を保って転送されました。

8090	PART パラメータの値が無効であるか、または指定された出力のプロセスイメージパーティションの更新がこの OB では許可されていません。出力は I/O に転送されませんでした。出力のプロセスイメージパーティションは、変更されません。
8091	指定されたプロセスイメージパーティションがまだ定義されていない、または CPU の許可されたプロセスイメージ領域にない。 出力が I/O に転送されませんでした。出力のプロセスイメージパーティションは、変更されません。
8092	プロセスイメージパーティションがシステムによって更新されており、PART パラメータに使用することができません。
8093	他の OB でプロセスイメージパーティションの更新が処理されています。
80A0	出力の指定されたプロセスイメージパーティションの更新の場合、アクセスエラーが検出されました。 不正な出力は I/O に転送されませんでした。プロセスイメージパーティションの出力時には、これらの出力は変更されません。
80A1	整合性の警告付きのアクセスエラー 指定された出力のプロセスイメージパーティションの更新の場合、同時整合性警告付きのアクセスエラーが検出されました。 • 不正な出力のデータは I/O に転送されませんでした。プロセスイメージパーティションの出力時には、関係する出力は変更されません。 • アクセスエラー入力データに関連しないプロセスイメージパーティションの更新は、2 つの DP および PN サイクルに分散されました。
80A2	更新時間が許可されたアクセス時間の後になっています。 出力が I/O に転送されませんでした。出力のプロセスイメージパーティションは、変更されません。
80C1	更新時間が許可されたアクセス時間の前になっています。 出力が I/O に転送されませんでした。出力のプロセスイメージパーティションは、変更されません。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

注記

32 バイトを超える整合性領域を定義した DP 標準スレーブのプロセスイメージパーティションに「SYNC_PO 命令を使用すると、「DPWR_DAT」命令のエラーコードも使用できます。

リモート I/O

この章には下記に関する情報が記載されています：

RDREC:Read data record (データレコードの読み出し)

説明

命令「RDREC」を使用して、ID によってアドレス指定したモジュールから、番号 INDEX を持つデータレコードを読み出します。 これは、基本ラックのモジュール、または分散型モジュール(PROFIBUS DP または PROFINET IO)などです。

- パラメータ ID を使用して、データレコードの読み出し元のモジュール(DP/PROFINET IO)を選択します。 「ID」パラメータでは、モジュールのハードウェア識別子のみを使用します。
- INDEX パラメータを使用して、モジュールから読み出す情報を選択します。 どのデータレコード番号でどのデータレコードを読み出すことができるかは、モジュールに応じて異なります。 追加情報は、それぞれのモジュールのマニュアルを参照してください。
- 「RDREC」を使用して読み出し可能なデータレコードは、異なる長さを持ちます。 MLEN を使用して、読み出すデータレコードの最大バイト数を指定します。 MLEN パラメータで長さ「0」が選択されている場合、完全なデータレコードがパラメータ RECORD に書き込まれます。
- 宛先領域 RECORD には、最低 MLEN バイトの長さが必要です。 MLEN=0 を使用して完全なデータレコードを読み出す場合は、RECORD で、データレコードの最大長を使用します。 RECORD パラメータで使用する構造(設定、データタイプ、および長さ)も、どのモジュールによってどのデータレコードを読み出すかに応じて異なります。
- 出力パラメータ VALID の値 TRUE は、データレコードがコピー先の領域 RECORD に問題なく転送されたことを示します。 この場合、LEN 出力パラメータには、読み出したデータレコードの実際の長さがバイトで含まれます。
- データレコードの転送中にエラーが発生した場合、これは出力パラメータ ERROR で示されます。この場合、出力パラメータ STATUS にはエラー情報が含まれます。

注記
「RDREC」命令のインターフェースは、「IEC 61131-3 に基づく PROFIBUS ガイドライン PROFIBUS 通信およびプロキシファンクションブロック」で定義された「RDREC」FB と同一です。

ファンクションの説明

"RDREC」は非同期で動作する命令で、その実行は複数の呼び出しにわたって行われます。 呼び出しデータレコードの転送は、REQ=1 で「RDREC」を呼び出して開始します。

ジョブステータスは、出力パラメータ BUSY と出力パラメータ STATUS の 2 つの中央のバイトで表示されます。 STATUS の中央の 2 バイトは、非同期で動作する命令の RET_VAL 出力パラメータに対応します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

データレコードの転送は、出力パラメータ BUSY が値 FALSE を持つときに完了します。

パラメータ

以下の表に、「RDREC」命令のパラメータを示します。

パラメータ	宣言	データタイプ*	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T**、C**、または定数	REQ = 1: データレコード転送

ID	Input	HW_IO	I、Q、M、D、L、または定数	ハードウェアモジュール(DP/PROFI-NET IO)のハードウェア識別子 番号が自動的に割り当てられ、ハードウェアコンフィグレーションのモジュールまたはインターフェースのプロパティに格納されます。
INDEX	Input	BYTE, DINT, INT, SINT, UINT, USINT, WORD	I、Q、M、D、L、または定数	データレコード番号
MLEN	Input	BYTE, UINT, USINT	I、Q、M、D、L、または定数	読み出すデータレコード情報の最大長(バイト)
VALID	Output	BOOL	I、Q、M、D、L	新しいデータレコードが受け取られ、有効です。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: 読み出しプロセスが未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	ERROR = 1: 読み出しプロセス中にエラーが発生しました。
STATUS	Output	DWORD	I、Q、M、D、L	ブロックステータスまたはエラー情報
LEN	Output	UINT	I、Q、M、D、L	読み出されたデータレコード情報の長さ
RECORD	InOut	VARIANT	I、Q、M、D、L	読み出したデータレコードのコピー先の領域です。
* STL には暗黙的な変換がないため、有効なデータタイプの範囲が限定される場合があります。 STL でのプログラミング中に、各ケースで許容されるデータタイプをパラメータの操作ヒントに書き留めます。 ** S7-1500 の場合のみ。				

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記
「RDREC」を使用して PROFINET IO のデータレコードを読み出す場合、INDEX、MLEN、および LEN パラメータの負の値は符号なしの 16 ビットの整数として解釈されます。

STATUS パラメータ

STATUS パラメータの解釈については、「パラメータ STATUS」を参照してください。

WRREC:Write data record (データレコードの書き込み)

説明

「WRREC」命令を使用して、ID によってアドレス指定されたコンポーネントに RECORD データレコードを転送します。 これは、基本ラックのモジュール、または分散型コンポーネント(PROFIBUS DP または PROFINET IO)などです。

- LEN を使用して、データレコードの長さをバイトで指定します。 転送元の範囲 RECORD に選択された長さは、LEN バイト以上の長さとなるようにしてください。
- 出力パラメータ DONE の値 TRUE は、データレコードが問題なく転送されたことを示します。
- データレコードの転送中にエラーが発生した場合、これは出力パラメータ ERROR で示されます。この場合、出力パラメータ STATUS にはエラー情報が含まれます。

注記
「WRREC」命令のインターフェースは、「IEC 61131-3 に基づく PROFIBUS ガイドライン PROFIBUS 通信およびプロキシファンクションブロック」で定義された「WRREC」FB と同一です。

ファンクションの説明

「WRREC」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 呼び出しデータレコードの転送は、REQ =1 で「WRREC」を呼び出して開始します。

ジョブステータスは、出力パラメータ BUSY と出力パラメータ STATUS の 2 つの中央のバイトで表示されます。 STATUS の中央の 2 バイトは、非同期で動作する命令の RET_VAL 出力パラメータに対応します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

1 つの同じジョブに属するすべての「WRREC」呼び出しの RECORD の実際のパラメータに同じ値を割り当てなければならないことに注意してください。 LEN の実パラメータにも、これが適用されます。

データレコードの転送は、出力パラメータ BUSY が値 FALSE を持つときに完了します。

パラメータ

次の表に、命令「WRREC」のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T*、C*、または定数	REQ = 1: データレコード転送
ID	Input	HW_IO	I、Q、M、D、L、または定数	ハードウェアコンポーネント(DP/PROFINET IO)の ID 番号 番号は自動的に割り当てられ、コンポーネントのプロパティまたはハードウェアコンフィグレーションのインターフェースに保存されます。
INDEX	Input	DINT	I、Q、M、D、L、または定数	データレコード番号

LEN	Input	BYTE, UINT, USINT	I、Q、M、D、L、または定数	(非表示) 転送するデータレコードの最大長(バイト)
DONE	Output	BOOL	I、Q、M、D、L	データレコードが転送されました。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY= 1: 書き込みプロセスが未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	ERROR 1:書き込みプロセス中にエラーが発生しました。
STATUS	Output	DWORD	I、Q、M、D、L	ブロックステータスまたはエラー情報 STATUS パラメータの解釈については、「パラメータ STATUS」を参照してください。
RECORD	InOut	VARIANT	I、Q、M、D、L	データレコード
* S7-1500 の場合のみ。				

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記
「WRREC」を使用して PROFINET IO のデータレコードを書き込む場合、INDEX および LEN パラメータの負の値は、符号なしの 16 ビットの整数として解釈されます。

STATUS パラメータ

STATUS パラメータの解釈については、「パラメータ STATUS」を参照してください。

GETIO: プロセスイメージの読み出し

説明

「GETIO」命令を使用し、DP 標準スレーブ/PROFINET IO デバイスのすべての入力を連続して読み出します。「GETIO」命令は、「DPRD_DAT」命令を呼び出します。 データ転送中にエラーがなかった場合、読み出されたデータは INPUTS で示されたターゲット範囲に入力されます。

ターゲット範囲は、選択したコンポーネントに設定した長さと同じ長さを持つ必要があります。

モジュール型構成または複数の DP 識別子を持つ DP 標準スレーブから読み出しを行う場合、設定された開始アドレスでアクセスできるのは「GETIO」呼び出しあたり 1 つのコンポーネント/DP 識別子のデータのみです。

パラメータ

次の表に、「GETIO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
ID	Input	HW_SUB-MODULE	I、Q、M、D、L または定数	DP 標準スレーブ/PROFINET IO デバイスのハードウェア ID です。
STATUS	Output	DWORD	I、Q、M、D、L	「DPRD_DAT」のエラー情報を DW#16#40xxxx00 の形で含みます。
LEN	Output	INT	I、Q、M、D、L	読み出されたデータ量(バイト)
INPUTS	InOut	VARIANT	I、Q、M、D	読み出したデータのターゲット範囲です。 これは、選択した DP 標準スレーブ/PROFINET IO デバイスに設定した範囲と同じ長さであることが必要です。 BYTE データタイプのみが許可されます。

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

関連項目 DPRD_DAT: DP スレーブの整合性のあるデータ読み出し.

SETIO: プロセスイメージの転送

説明

「SETIO」命令は、パラメータ OUTPUTS によって定義されたソース範囲から、データをアドレス指定された DP 標準スレーブ/PROFINET IO デバイスに、さらに必要に応じてプロセスイメージに(DP 標準スレーブ/PROFINET IO デバイスの関連するアドレス領域をプロセスイメージの整合性のある範囲として設定している場合)、整合性を保って転送するために使用します。 "「SETIO」は「DPWR_DAT」命令を呼び出します。

ソース範囲は、選択したコンポーネントに設定した長さと同じ長さを持つ必要があります。

モジュール型構成または複数の DP 識別子を持つ DP 標準スレーブ/PROFINET IO デバイスの場合、「SETIO」呼び出しあたりアクセスできるのは 1 つの DP 識別子/コンポーネントのみです。

パラメータ

次の表に、「SETIO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
ID	Input	HW_SUB-MODULE	I、Q、M、D、L、または定数	DP 標準スレーブ/PROFINET IO デバイスのハードウェア ID です。
STATUS	Output	DWORD	I、Q、M、D、L	「DPWR_DAT」のエラー情報を DW#16#40xxxx00 の形で含みます。
OUTPUTS	InOut	VARIANT	I、Q、M、D	書き込まれるデータのソース範囲。 これは、選択した DP 標準スレーブ/PROFINET IO デバイスに設定した範囲と同じ長さであることが必要です。 BYTE データタイプのみが許可されます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ STATUS

関連項目 DPWR_DAT: DP スレーブのコンスタントデータの書き込み.

GETIO_PART: プロセスイメージ領域の読み出し

説明

「GETIO_PART」命令を使用し、IO モジュールの入力の関連部分を連続して読み出します。 "「GETIO_PART」は「DPWR_DAT」命令を呼び出します。

ID 入力パラメータを使用して、ハードウェア ID によって IO モジュールを選択します。

OFFSET および LEN パラメータを使用し、読み出されるプロセスイメージ領域の部分を指定します。 OFFSET および LEN によって指定された入力領域をモジュールが完全に認識できない場合、ブロックはエラーコード DW#16#4080B700 を返します。

ターゲット範囲の長さは、読み出されるバイト数以上である必要があります。

- データ転送中にエラーが発生しなかった場合、ERROR は値 FALSE を受信します。 読み出されるデータは、INPUTS パラメータで指定されたターゲット範囲に書き込まれます。
- データ転送中にエラーが発生しなかった場合、ERROR は値 TRUE を受信します。 STATUS パラメータは、「DPRD_DAT」からエラー情報を受信します。
- ターゲット範囲が LEN を超える場合、宛先領域の最初の LEN バイトが書き込まれます。ERROR は値 FALSE を受信します。

パラメータ

次の表に、「GETIO_PART」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
ID	Input	HW_SUBMODULE	I、Q、M、D、L、または定数	モジュールのハードウェア識別子
OFFSET	Input	INT	I、Q、M、D、L、または定数	コンポーネントのプロセスイメージで読み込まれる最初のバイトの番号(最小値: 0).
LEN	Input	INT	I、Q、M、D、L、または定数	読み出されるバイト数。
STATUS	Output	DWORD	I、Q、M、D、L	ERROR = TRUE の場合、「DPRD_DAT」のエラー情報を DW#16#40xxxx00 の形で含みます。
ERROR	Output	BOOL	I、Q、M、D、L	エラー表示: ERROR = TRUE 「DPRD_DAT」が呼び出されたときにエラーが発生した場合。
INPUTS	InOut	VARIANT	I、Q、M、D	読み出したデータのターゲット範囲です。 ターゲット範囲が LEN より大きい場合、ターゲット範囲の最初の LEN バイトが書き込まれます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ STATUS

「DPRD_DAT」命令の「RET_VAL」パラメータを参照してください。

SETIO_PART: プロセスイメージエリアの転送

説明

「SETIO_PART」命令を使用し、OUTPUTS で指定されたソース領域から IO モジュールの出力へデータを連続して書き込むことが可能です。「SETIO_PART」は「DPWR_DAT」命令を呼び出します。

ID 入力パラメータを使用して、ハードウェア ID によって IO モジュールを選択します。
OFFSET および LEN パラメータを使用し、ID によって指定されたコンポーネントの書き込まれるプロセスイメージ領域の部分を指定します。 OFFSET および LEN によって指定された出力領域をモジュールが完全に認識できない場合、ブロックはエラーコード DW#16#4080B700 を返します。

ソース範囲の長さは、書き込まれるバイト数以上である必要があります。

- データ転送中にエラーが発生しなかった場合、ERROR は値 FALSE を受信します。
- データ伝送中にエラーがあった場合、ERROR が値 TRUE を受信し、STATUS が「DPWR_DAT」のエラー情報を受信します。
- ソース範囲が LEN を超える場合、OUTPUTS からの最初の LEN バイトが転送されます。ERROR は値 FALSE を受信します。

パラメータ

次の表に、「SETIO_PART」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
ID	Input	HW_SUB-MODULE	I、Q、M、D、L または定数	IO モジュールのハードウェア識別子。
OFFSET	Input	INT	I、Q、M、D、L または定数	コンポーネントのプロセスイメージに書き込まれる最初のバイトの番号(最小値: 0)。
LEN	Input	INT	I、Q、M、D、L または定数	書き込まれるバイト数。
STATUS	Output	DWORD	I、Q、M、D、L	ERROR = TRUE の場合、「DPWR_DAT」のエラー情報を DW#16#40xxxx00 の形で含みます。
ERROR	Output	BOOL	I、Q、M、D、L	エラー表示: ERROR = TRUE「DPWR_DAT」が呼び出されたときにエラーが発生した場合。
OUTPUTS	InOut	VARIANT	I、Q、M、D	書き込まれるデータのソース範囲。 ソース範囲が LEN を超える場合、最初の LEN バイトが OUTPUTS から転送されます。

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

パラメータ STATUS および ERROR

「DPWR_DAT 」命令を参照してください。

RALRM: 割り込みの受信

この章には下記に関する情報が記載されています：

RALRM の説明

説明

この命令は、I/O モジュール(集中型構成)または DP スレーブのモジュールまたは PROFINET IO デバイスから対応する情報の割り込みを受信します。これが、その出力パラメータにこの情報を提供します。

出力パラメータの情報には、呼び出された OB の開始情報と、割り込みソースの情報が含まれます。

集中構成では、宛先領域 AINFO のデータ構造体の構成が PROFINET IO のデータ構造体と一致します。

「RALRM」は、検証する I/O 割り込みの結果として CPU のオペレーティングシステムで開始された割り込み OB 内のみで呼び出してください。

注記
開始イベントが I/O 割り込みでない OB で「RALRM」を呼び出した場合、この命令はそれに対応して少ない情報を提供します。
異なる OB で「RALRM」を呼び出す場合は、必ず異なるインスタンス DB を使用してください。 関連する割り込み OB 以外の「RALRM」呼び出しからの結果のデータを評価する場合は、OB 開始イベントごとに個別のインスタンス DB をさらに使用します。

注記
「RALRM」命令のインターフェースは、「IEC 61131-3 に基づく PROFIBUS ガイドライン PROFIBUS 通信およびプロキシファンクションブロック」で定義された「RALRM」FB と同一です。

呼び出し RALRM

"「RALRM」は、3 つの動作モード(MODE パラメータ)で呼び出すことができます。 次の表で、これらを説明します。

MODE	RALRM ...
0	... ID 出力パラメータで割り込みをトリガしたコンポーネントを示し、TRUE を NEW 出力パラメータに書き込みます。
1	... 割り込みをトリガしたコンポーネントに関係なく、すべての出力パラメータを書き込みます。
2	... F_ID 入力パラメータで指定されたコンポーネントが割り込みをトリガしたかどうかチェックします。 • トリガしていない場合、NEW = FALSE • トリガした場合、NEW = TRUE で、その他すべての出力パラメータが書き込まれます。

パラメータ

以下の表に、「RALRM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	INT	I、Q、M、D、L、または定数	モード
F_ID	Input	HW_IO	I、Q、M、D、L、または定数	モジュールのハードウェア識別子。番号は自動的に割り当てられ、コンポーネントまたはインターフェースのハードウェアコンフィグレーションのプロパティに保存されます。
MLEN	Input	UINT	I、Q、M、D、L、または定数	受信する割り込み情報の最大長(バイト)。 MLEN = 0、AINFO パラメータで指定されたすべてのデータが読み取られる場合。
NEW	Output	BOOL	I、Q、M、D、L	新規の割り込みが受信されました。
STATUS	Output	DWORD	I、Q、M、D、L	エラーコード
ID	Output	HW_IO	I、Q、M、D、L	割り込みが受信されたモジュールのハードウェア識別子。
LEN	Output	UINT	I、Q、M、D、L	受信された割り込み情報の長さ
TINFO	InOut	VARIANT	I、Q、M、D、L	OB 開始および管理情報の宛先領域
AINFO	InOut	VARIANT	I、Q、M、D、L	ヘッダー情報および追加の割り込み情報の宛先領域 AINFO の場合、少なくとも MLEN バイトの長さを提供してください。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記
短すぎるコピー先の領域 TINFO または AINFO を選択した場合、RALRM はすべての情報を入力できません。

パラメータ STATUS

説明

STATUS 出力パラメータには、エラー情報が含まれます。 ARRAY[1...4] of BYTE と解釈された場合、エラー情報は次の構造になります。

フィールドエレメント	名前	意味
STATUS[1]	Function_Num	• B#16#00、エラーがない場合 • DPV1-PDU からのファンクション ID: エラーが発生した場合、B#16#80 が出力されます(データレコード B#16#DE の読み出しおよびデータレコード B#16#DF の書き込みのエラーの場合)。 DPV1 プロトコルエレメントが使用されていない場合、B#16#C0 が出力されます。
STATUS[2]	Error_Decode	エラー ID の場所
STATUS[3]	Error_Code_1	エラー ID
STATUS[4]	Error_Code_2	製造元固有のエラー ID 拡張

フィールドエレメント STATUS[2]

STATUS[2]は、次の値を取ることができます。

Error_Decode (B#16#...)	ソース	意味
00 ~ 7F	CPU	エラーなし、または警告なし
80	DPV1	IEC 61158-6 に準じたエラー
81 ~ 8F	CPU	B#16#8x は、命令の x 番目の呼び出しパラメータのエラーを示します。
FE、FF	DP プロファイル	プロファイル固有のエラー

フィールドエレメント STATUS[3]

STATUS[3]は、次の値を取ることができます。

Error_Decode (B#16#...)	Error_Code_1 (B#16#...)	DVP1 に基づく説明	意味
00	00		エラーなし、警告なし
70	00	reserved, reject	最初の呼び出し、有効なデータレコードの転送なし
	01	reserved, reject	最初の呼び出し、データレコードの転送が開始

	02	reserved, reject	中間呼び出し、データレコードの転送が既に有効
80	81		TINFO パラメータのシステムデータタイプが、この命令の呼び出し環境に適合しません。 使用されるシステムデータタイプは、ユーザープログラムのオーガニゼーションブロックに適合する必要があります(例: 遅延割り込み OB の場合、システムデータタイプ TI_Delay が必要です)。
	90	reserved, pass	無効な論理開始アドレス
	92	reserved, pass	VARIANT ポインタで無効なタイプ
	93	reserved, pass	ID または F_ID でアドレス指定された DP コンポーネントが設定されていません。
	96		「RALRM」が OB 開始情報、管理情報、ヘッダー情報、または追加の割り込み情報を提供できません。 OB 4x、55、56、57、82 、および 83 の場合、「DPNRM_DG」命令を使用して、関連する DP スレーブの現在の診断メッセージフレームを非同期で読み出すことができます(OB 開始情報からのアドレス情報)。
	A0	read error	モジュール読み出し中の否定確認
	A1	write error	モジュールへの書き込み時の否定確認
	A2	module failure	レイヤー 2 での DP プロトコルエラー(たとえばスレーブ故障またはバスの問題)
	A3	reserved, pass	• PROFIBUS DP: ダイレクトデータリンクマッパ(DDLM)またはユーザーインターフェース/ユーザーでの DP プロトコルエラー • PROFINET IO: 一般 CM エラー
	A4	reserved, pass	PBUS+の通信が途切れました
	A5	reserved, pass	–
	A7	reserved, pass	DP スレーブまたはモジュールが占有されている(一時的なエラー
	A8	version conflict	DP スレーブまたはモジュールが非互換バージョンをレポート
	A9	feature not supported	DP スレーブまたはモジュールがファンクションをサポートしていない
	AA ~ AF	user specific	DP スレーブまたはモジュールがそのアプリケーションで製造元固有エラーをレポート。 DP スレーブまたはモジュールの製造元の文書を確認してください。
	B0	invalid index	モジュールでデータレコードが不明 無効なデータレコード番号≥ 256
	B1	write length error	長さ指定のエラー: • 「RALRM」の場合: AINFO の長さエラー • 「RDREC」の場合: MLEN の長さエラー • 「WRREC」の場合:LEN の長さエラー

	B2	invalid slot	設定されたスロットが 割り当てられていません。
	B3	type conflict	実際のモジュールタイプが指定されたモジュールタイプと一致していません。
	B4	invalid area	DP スレーブまたはモジュールが無効な領域へのアクセスをレポート
	B5	state conflict	DP スレーブまたはモジュールの準備ができていない
	B6	access denied	DP スレーブまたはモジュールがアクセス拒否
	B7	invalid range	DP スレーブまたはモジュールがパラメータまたは値の無効な範囲をレポート
	B8	invalid parame-ter	DP スレーブまたはモジュールが無効なパラメータをレポート
	B9	invalid type	DP スレーブまたはモジュールが無効タイプをレポート 「RDREC」の場合: バッファが小さすぎる(サブセットを読み出せない) 「WRREC」の場合: バッファが小さすぎる(サブセットを書き込めない)
	BA ~ BF	user specific	DP スレーブまたはモジュールがアクセス時に製造元固有エラーをレポート。 DP スレーブまたはモジュールの製造元の文書を確認してください。
	C0	read constrain conflict	「WRREC」の場合: データは、CPU が STOP モードの場合のみ書き込み可能です。 注記: つまり、ユーザープログラムによる書き込みは不可能です。 データの書き込みは、PG/PC を使ってオンラインのみで可能です。 「RDREC」の場合: モジュールはデータレコードをルーティングしますが、データが存在しないか、CPU が STOP モードの場合のみにデータを読み出すことができます。 注記: CPU が STOP モードの場合のみにデータを読み出すことができる場合、ユーザープログラムによる評価は不可能です。 この場合、データの読み出しは、PG/PC を使ってオンラインのみで可能です。
	C1	write constrain conflict	モジュールで、同じデータレコードの前の書き込みジョブのデータがモジュールによって処理されていません。
	C2	resource busy	モジュールは、現在 CPU で可能な最大数のジョブを処理しています。
	C3	resource un-available	必要な操作リソースが現在占有されています。
	C4		内部の一時的なエラー。 ジョブを実行できませんでした。 ジョブを繰り返します。 このエラーが頻発する場合は、据付けを点検して電気的な干渉の発生源を探します。

	C5		DP スレーブまたはモジュールが使用できません。
	C6		優先度クラスのキャンセルによって、データレコードの転送がキャンセルされた
	C7		DP マスタのウォームまたはコールドリスタートのため、ジョブが中止されました。
	C8 ~ CF		DP スレーブまたはモジュールが製造元固有リソースエラーをレポート。 DP スレーブまたはモジュールの製造元の文書を確認してください。
	Dx	user specific	DP スレーブ固有。 DP スレーブの説明を参照してください。
81	00 ~ FF		最初の呼び出しパラメータのエラー(「RALRM」の場合: MODE)
	00		無効な動作モード
82	00 ~ FF		2 番目の呼び出しパラメータのエラー
:	:		:
88	00 ~ FF		8 番目の呼び出しパラメータのエラー (「RALRM」の場合: TINFO)
	01		不正な構文 ID
	23		数量構造を上回ったか、またはターゲット範囲が小さすぎます。
	24		不正な範囲 ID
	32		DB/DI 番号がユーザー範囲外
	3A		領域 IDDB/DI で DB/DI 番号が Null、または指定された DB/DI が存在しない
89	00 ~ FF		9 番目の呼び出しパラメータのエラー (「RALRM」の場合: AINFO)
	01		不正な構文 ID
	23		数量構造を上回ったか、またはターゲット範囲が小さすぎます。
	24		不正な範囲 ID
	32		DB/DI 番号がユーザー範囲外
	3A		領域 IDDB/DI で DB/DI 番号が Null、または指定された DB/DI が存在しない
8A	00 ~ FF		10 番目の呼び出しパラメータのエラー
:	:		:
8F	00 ~ FF		15 番目の呼び出しパラメータのエラー
FE, FF	00 ~ FF		プロファイル固有のエラー

フィールドエレメント STATUS[4]

DPV1 エラーがある場合、DP マスタが STATUS[4] を CPU および命令に渡します。 DPV1 エラーがない場合、この値は 0 に設定されますが、「RDREC」では次が例外です。

- MLEN > RECORD からのターゲット範囲の長さの場合、STATUS[4] は、RECORD からのターゲット範囲の長さを含む
- 実際のデータレコード長 < MLEN < RECORD からのターゲット範囲の長さの場合、STATUS[4]=MLEN
- STATUS[4]> 255 を設定する必要がある場合、STATUS[4]=0

PROFINET IO で、STATUS[4] の値が「0」になっています。

パラメータ TINFO

コピー先の領域 TINFO のデータ構造体

コピー先の領域のデータ構造体 TINFO には、「RALRM」が現在呼び出されているオーガニゼーションブロックの開始情報が含まれています。

TINFO コピー先の領域には、標準アクセスまたは最適化されたアクセスを使用する開始情報を含めることができます。 TINFO 宛先領域の開始情報の形式は、対応するオーガニゼーションブロックの開始情報と必ず一致している必要があります。

- 標準アクセスの OB の開始情報は、必ずブロックインターフェースの「Temp」セクションの最初の 20 バイト内に格納されています。 これには「TI_Classic」データ構造体を使用してください。
- 最適化されたアクセスを使用する OB の開始情報は、必ず「Input」セクションに書き込まれます。これらの OB には特定の OB タイプのデータ構造体を使用してください。

ブロックアクセス(標準/最適化)を変更すると、ブロックインターフェースも変更されます。

次の表は、TINFO パラメータでオーガニゼーションブロックに応じて使用されるデータ構造体の概要を示しています。

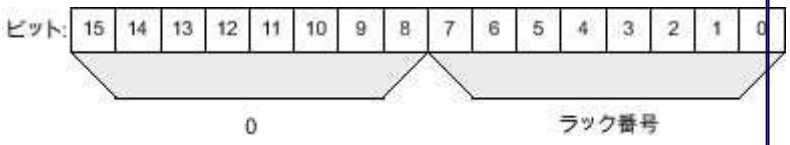
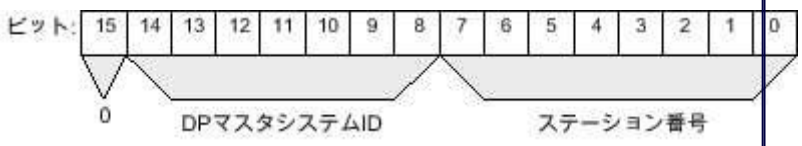
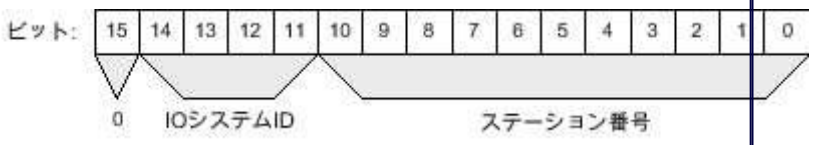
データ構造体の名前	S7-1200 CPU、以下のバージョン以降	S7-1500 CPU、以下のバージョン以降	使用されるデータ構造:
標準アクセスを使用するオーガニゼーションブロックのデータ構造			
TI_Classic	-	V1	最適化されたブロックアクセスを使用しないオーガニゼーションブロック。
最適化されたブロックアクセスを使用するオーガニゼーションブロックのデータ構造			
TI_ProgramCycle	V2	V1	サイクル OB (プログラムサイクル)
TI_Startup	V2	V1	スタートアップ OB(スタートアップ)
TI_Delay	V2	V1	遅延割り込み
TI_Cyclic	V2	V1	周期割り込み OB
TI_HWInterrupt	V2	V1	ハードウェア割り込み OB
TI_TimeError	V2	V1	タイムエラー OB
TI_DiagnosticInterrupt	V2	V1	診断エラー割り込み OB
TI_PlugPullModule	V2	V1	モジュール OB の切断/接続
TI_StationFailure	V2	V1	ラック障害 OB またはステーション障害 OB
TI_ProgIOAccessError	V2	V1	• プログラミングエラー OB • I/O アクセスエラー OB
TI_TimeOfDay	V2	V1	時刻割り込み OB
TI_SynchCycle	-	V1	同期サイクル割り込み OB
TI_Submodule	V2	V1	• ステータス割り込み OB • 割り込み OB の更新

			• 製造元またはプロファイル固有の割り込みの OB

標準アクセスを使用するオーガニゼーションブロックのデータ構造

以下に、「 TI_Classic」データ構造体を示します。

パラメータ	データタイプ	Byte (バイト)	説明
TI_Classic			最適化されたブロックアクセスを使用するオーガニゼーションブロックのデータ構造。
バイト 0 ~ 19: 「RALRM」が現在呼び出されている OB の開始情報。*			
EV_CLASS	BYTE	0	イベントクラス 例 OB1: • ビット 0 ~ 3: イベントの ID (1 = 受信イベント) • ビット 4 ~ 7: イベントクラス(1 = イベントクラス 1)
EV_NUM	BYTE	1	イベント番号(OB タイプによる) 例 OB1 (SCAN_1): • SCAN_1 = 1 (最初の呼び出しの場合) • SCAN_1 = 3 (すべての追加呼び出しの場合)
PRIORI-TY	BYTE	2	優先度クラス
NUM	BYTE	3	DB 番号
TYP2_3	BYTE	4	追加情報 使用される OB タイプに応じて、異なる情報が BYTES「TYP2_3」および「TYPE 1」に保存されます。 これらについては、オーガニゼーションブロックの文書に記載されています。 例(OB1): • TYP2_3: OB1_RESERVED_1 (予約済み) • TYP1: OB1_RESERVED_2 (予約済み)
TYP1	BYTE	5	
ZI1	WORD	6 ~ 7	追加情報 使用される OB タイプに応じて、異なる情報が「ZI1」に保存されます。 これらについては、オーガニゼーションブロックの文書に記載されています。 例(OB1): • ZI1: OB1_PREV_CYCLE (前のサイクルのランタイム(ミリ秒))
ZI2_3	DWORD	8 ~ 11	追加情報 使用される OB タイプに応じて、異なる情報が「ZI2_3」に保存されます。 これらについては、オーガニゼーションブロックの文書に記載されています。

			例(OB1): • ZI2: OB1_MIN_CYCLE (最後のスタートアップ以降の最小サイクルタイム(ミリ秒)) • ZI3: OB1_MAX_CYCLE (最後のスタートアップ以降の最大サイクルタイム(ミリ秒))
OB_DATE_TIME	DATE_AND_TIME (DT)	12 ~ 19	OB が呼び出された日付および時刻。
バイト 20 および 21: アドレス情報			
address	WORD	20 および 21	S7-300/400 CPU などのアドレス情報: • 集中構成のラック番号(0-31):  • PROFIBUS DP による分散構成の場合: ○ DP マスタシステム ID (1-31) ○ ステーション番号(0-127)。  • PROFINET IO による分散構成: ○ PROFINET IO システム ID の下 2 桁(0-15)。 完全な PROFINET IO システム ID を取得するには、これに 100 (10 進) を加算する必要があります。 ○ ステーション番号(0-2047)。 
バイト 22 ~ 31: 管理情報			
slv_prfl	BYTE	22	S7-300/400 CPU などのスレーブプロファイル: • 集中構成の場合: 0 (データレコード 0 またはデータレコード 1) • 分散構成の場合: ○ ビット 0 ~ 3: スレーブタイプ - 0000: DP (構造のデータレコード 0) - 0001: DPS7 (構造のデータレコード 0 またはデータレコード 1) - 0010: DPS7 V1 (構造のデータレコード 0 またはデータレコード 1) - 0011: DPV1 (PROFIBUS DP 標準に従った構造) - 0100 – 0111: 予約済み

			- 1000: PROFINET IO (PROFINET IO 標準に従った構造) - 1001～: 予約済み ○ ビット 4～7: プロファイルタイプ (予約済み)
intr_type	BYTE	23	S7-300/400 CPU などの割り込み情報タイプ: • 集中構成の場合: 0 • 分散構成の場合: ○ ビット 0～3: 割り込み情報タイプ - 0000: 割り込みは構成済みリモートモジュールから発生します - 0001: 割り込みは非 DPV1 スレーブ/非 IO デバイスまたは未設定のスロットから発生します - 0010: 割り込みは CPU で生成されます - 0011～: 予約済み ○ ビット 4～7: 構造体バージョン - 0000: 初期 - 0001～: 予約済み
flags1	BYTE	24	S7-300/400 CPU などの PROFIBUS DP マスタインターフェースモジュール/PROFINET IO コントローラマスタインターフェースモジュールのフラグ: • 集中構成の場合: 0 • 分散構成の場合: ○ ビット 0 = 0: 統合インターフェースモジュール(PROFINET IO または PROFIBUS DP)から発生する割り込み ○ ビット 0 = 1: 外部インターフェースモジュール(PROFINET IO または PROFIBUS DP)から発生する割り込み ○ ビット 1～7: 予約済み
flags2	BYTE	25	S7-300/400 CPU などの PROFIBUS DP マスタインターフェースモジュール/PROFINET IO コントローラマスタインターフェースモジュールのフラグ: • 集中構成の場合: 0 • PROFIBUS DP による分散構成の場合: ○ ビット 0: 診断メッセージフレームからの EXT_DIAG_FLAG、またはこのビットが割り込みに存在しない場合は 0。DP スレーブが障害の場合、このビットは 1 です。 ○ ビット 1～7: 予約済み • PROFINET IO による分散構成: ○ ビット 0: ARDiagnosisState または割り込みに情報がない場合は 0。 IO デバイス障害の場合ビットが 1。 ○ ビット 1～7: 予約済み
id	UINT	26 および 27	管理情報 • 集中構成の場合: 0 • PROFIBUS DP による分散構成の場合: PROFIBUS DP スレーブの一意の識別子としての PROFIBUS ID 番号 • PROFINET IO による分散構成: PROFINET IO デバイスの一意の識別としての PROFINET IO デバイス ID 番号

manufac-turer	UINT	28 および 29	製造元 ID (PROFINET IO による分散構成の場合のみ)。
instance	UINT	30 および 31	インスタンスの IDENT 番号(PROFINET IO による分散構成の場合のみ)。

* 開始情報は、使用される OB によって異なります。 各 OB タイプの開始情報は、インターフェースまたは OB の文書を参照してください。

最適化されたブロックアクセスを使用するオーガニゼーションブロックのデータ構造

最適化されたブロックアクセスを使用するオーガニゼーションブロックのデータ構造の形式は次の通りです。

- バイト 0 ~ 3: 呼び出される OB の開始情報、クラス、および番号の形式(すべてのデータ構造体に対して同じ構造)。
- バイト 4 ~ 19: 最適化された開始情報(構造は OB タイプによって異なる)。 バイト 4 ~ 19 のデータは、対応する OB インターフェースの構造および内容に相当します。
- バイト 20 ~ 31: さらに、特定の OB のアドレスおよび管理情報。 バイト 20 ~ 31 のデータは、TI_Classic データ構造の 20 ~ 31 バイトのデータに相当します。

データ構造体の形式については、以下の表に記載されています。

サイクル OB: データ構造 TI_ProgramCycle

パラメータ	データタイプ	Byte (バイト)	説明
TI_ProgramCycle			サイクル OB のデータ構造体
SI_Format	USINT	0	開始情報の形式 • 16#FF: なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス(=1)
OB_Nr	UINT	2	DB 番号(1...32767)。
Initial_Call	BOOL	4	=TRUE、この OB の最初の呼び出し時 • STOP または HOLD から RUN への移行 • 再ロード後
Rema-nence	BOOL	5	= TRUE、保持型データが使用可能の場合。

スタートアップ OB: データ構造 TI_Startup

パラメータ		データタイプ	Byte (バイト)	説明
TI_Startup				スタートアップ OB のデータ構造体
	SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
	OB_Class	USINT	1	OB クラス(=100)
	OB_Nr	UINT	2	DB 番号(1...32767)。
	LostRetentive	BOOL	4	=TRUE、保持型データ領域の内容が失われた場合。
	LostRTC	BOOL	5	=TRUE、リアルタイムクロックの時刻が失われた場合。

遅延割り込み OB: データ構造 TI_Delay

パラメータ		データタイプ	Byte (バイト)	説明
TI_Delay				遅延割り込み OB のデータ構造体
	SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
	OB_Class	USINT	1	OB クラス(=20)
	OB_Nr	UINT	2	DB 番号(1...32767)。
	Sign	WORD	4	ユーザー ID: 「SRT_DINT」命令の呼び出しからの入力パラメータ SIGN。

サイクリック割り込み OB: データ構造 TI_Cyclic

パラメータ		データタイプ	Byte (バイト)	説明
TI_Cyclic				サイクリック割り込み OB のデータ構造体
	SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
	OB_Class	USINT	1	OB クラス(=30)
	OB_Nr	UINT	2	DB 番号(1...32767)。
	Initial_Call	BOOL	4	=TRUE、この OB の最初の呼び出し時

			• STOP または HOLD から RUN への移行時 • 再ロード後
Event_Count	INT	6	この OB の最後の開始以降に破棄された開始イベントの数。

ハードウェア割り込み OB: データ構造 TI_HWInterrupt

パラメータ	データタイプ	Byte (バイト)	説明
TI_HWInterrupt			ハードウェア割り込み OB のデータ構造体
SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス(=40)
OB_Nr	UINT	2	DB 番号(1...32767)。
LADDR	HW_IO	4	ハードウェア割り込みをトリガするモジュールのハードウェア識別子。
USI	WORD	6	将来の拡張のための識別子(非ユーザー関連)。
IChannel	USINT	8	ハードウェア割り込みをトリガしたチャンネルの番号。
EventType	BYTE	9	割り込みをトリガするイベントに関連するイベントタイプの識別子(信号立ち上がりエッジなど)。 この ID については、それぞれのモジュールの説明を参照してください。
address	WORD	20	TI_Cassic.データ構造体については、「address」パラメータを参照してください。
slv_prfl	BYTE	22	TI_Cassic データ構造体については、「slv_prfl」パラメータを参照してください。
intr_type	BYTE	23	TI_Cassic データ構造体については、「intr_type」パラメータを参照してください。
flags1	BYTE	24	TI_Cassic データ構造体については、「flags1」パラメータを参照してください。
flags2	BYTE	25	TI_Cassic データ構造体については、「flags2」パラメータを参照してください。
id	UNIT	26	TI_Cassic データ構造体については、「id」パラメータを参照してください。
manufacturer	UNIT	28	TI_Cassic データ構造体については、「manufacturer」パラメータを参照してください。
instance	UNIT	30	TI_Cassic データ構造体については、「instance」パラメータを参照してください。

タイムエラー OB: データ構造 TI_TimeError

パラメータ		データタイプ	Byte (バイト)	説明
TI_TimeError				タイムエラー OB のデータ構造体
	SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
	OB_Class	USINT	1	OB クラス(=80)
	OB_Nr	UINT	2	DB 番号(1...32767)。
	Csg_OBnr	OB_ANY	4	タイムエラーが発生したときに実行されていた OB の番号。
	Fault_ID	BYTE	7	エラーコード。 次の値が返されます。 • B#16#01: サイクルタイムを超えています • B#16#02: 要求された OB はまだ処理中です。 • B#16#05: タイムジャンプによって期限が切れた時刻割り込み。 • B#16#06: HOLD 後に再び RUN になったことによって期限が切れた時刻割り込み。 • B#16#07: 現在の優先度クラスに対する OB 要求バッファのオーバーフロー。 • B#16#08: アイソクロナスモードの割り込み - タイムエラー。 • B#16#09: 高い割り込み負荷による割り込み損失 • B#16#0B: テクノロジー同期割り込み - タイムエラー
	Csg_Prio	UNIT	8	タイムエラーが発生したときに実行されていた OB の優先度。

診断割り込み OB: データ構造 TI_DiagnosticInterrupt

パラメータ		データタイプ	Byte (バイト)	説明
TI_DiagnosticInterrupt				診断割り込み OB のデータ構造体
	SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
	OB_Class	USINT	1	OB クラス(=82)
	OB_Nr	UINT	2	DB 番号(1...32767)。
	LADDR	HW_ANY	4	診断割り込みをトリガしたハードウェアオブジェクトのハードウェア識別子。
	IO_State	WORD	6	ハードウェアオブジェクトのステータス: • ビット 0: 良好 • ビット 1: 無効 • ビット 2: メンテナンス必須

			• ビット 3: メンテナンス要求 • ビット 4: エラー • ビット 5: アクセス不可能 • ビット 6: 限定 • ビット 7: 使用不可
Channel	UINT	8	チャンネル番号
MultiError	BOOL	10	=TRUE、複数のエラーが存在する場合。
address	WORD	20	TI_Cassic.データ構造体については、「address」パラメータを参照してください。
slv_prfl	BYTE	22	TI_Cassic データ構造体については、「slv_prfl」パラメータを参照してください。
intr_type	BYTE	23	TI_Cassic データ構造体については、「intr_type」パラメータを参照してください。
flags1	BYTE	24	TI_Cassic データ構造体については、「flags1」パラメータを参照してください。
flags2	BYTE	25	TI_Cassic データ構造体については、「flags2」パラメータを参照してください。
id	UINT	26	TI_Cassic データ構造体については、「id」パラメータを参照してください。
manufactur-er	UINT	28	TI_Cassic データ構造体については、「manufacturer」パラメータを参照してください。
instance	UINT	30	TI_Cassic データ構造体については、「instance」パラメータを参照してください。

切断/接続割り込み OB: データ構造 TI_PlugPullModule

パラメータ	データタイプ	Byte (バイト)	説明
TI_PlugPullModule			切断/接続割り込み OB のデータ構造体
SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス(=83)
OB_Nr	UINT	2	DB 番号(1...32767)。
LADDR	HW_IO	4	影響を受けるモジュールまたはサブモジュールのハードウェア識別子
Event_Class	BYTE	6	• B#16#38: (サブ)モジュール差し込み済み • B#16#39: (サブ)モジュール引き出し済みまたは応答なし
Fault_ID	BYTE	7	エラーコード 次の表に、切断/接続割り込み OB の開始の原因になるイベントを示します。

			• Event_Class = B#16#38 - モジュール/サブモジュール差し込み済みの場合: ○ B#16#54: サブモジュールが差し込まれ、設定されたサブモジュールと一致しています。 ○ B#16#55: サブモジュールが差し込まれましたが、設定されたサブモジュールと一致していません。 ○ B#16#56: サブモジュールが差し込まれましたが、モジュールパラメータ割り当てでエラーが生じました。 ○ B#16#57: サブモジュールまたはモジュールが差し込まれましたが、故障があるかメンテナンス中です ○ B#16#58: サブモジュールアクセスエラーが修正されました。 ○ B#16#61: モジュール差し込み済み、モジュールタイプ OK ○ B#16#63: モジュールが差し込まれましたが、モジュールタイプが不正です ○ B#16#64: モジュールが差し込まれましたが、障害があります(モジュール ID を読み出せない) ○ B#16#65: モジュールが差し込まれましたが、モジュールパラメータ割り当てでエラーが生じました。 ○ B#16#66: モジュールが再び応答するようになりました。負荷電圧エラーが修正されました。 • Event_Class = B#16#39 - モジュール/サブモジュール引き出し済みまたは応答なしの場合: ○ B#16#51:モジュール引き出し済み ○ B#16#54:サブモジュール引き出し済み ○ B#16#61: モジュール引き出し済みまたは応答なし ○ B#16#66:モジュールが応答していません、負荷電圧エラー
address	WORD	20	TI_Cassic.データ構造体については、「address」パラメータを参照してください。
slv_prfl	BYTE	22	TI_Cassic データ構造体については、「slv_prfl」パラメータを参照してください。
intr_type	BYTE	23	TI_Cassic データ構造体については、「intr_type」パラメータを参照してください。
flags1	BYTE	24	TI_Cassic データ構造体については、「flags1」パラメータを参照してください。
flags2	BYTE	25	TI_Cassic データ構造体については、「flags2」パラメータを参照してください。
id	UINT	26	TI_Cassic データ構造体については、「id」パラメータを参照してください。
manufacturer	UINT	28	TI_Cassic データ構造体については、「manufacturer」パラメータを参照してください。
instance	UINT	30	TI_Cassic データ構造体については、「instance」パラメータを参照してください。

ラックエラー OB: データ構造 TI_StationFailure

パラメータ	データタイプ	Byte (バイト)	説明
TI_StationFailure			ラックエラー OB のデータ構造体
SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス(=86)
OB_Nr	UINT	2	DB 番号(1...32767)。
LADDR	HW_Device	4	不完全なハードウェアオブジェクトのハードウェア識別子。
Event_Class	BYTE	6	• B#16#32:「D_ACT_DP」命令を使用したステーションの有効化 • B#16#33:「D_ACT_DP」命令を使用したステーションの無効化 • B#16#38:発信イベント • B#16#39:着信イベント
Fault_ID	BYTE	7	エラーコード エラーコードは、ラックエラー OB の開始を引き起こすイベントを出力するために使用します。 • Event_Class = B#16#39, FAULT_ID = C1: 拡張ユニットの障害 • Event_Class = B#16#38, FAULT_ID = C1: 拡張ユニットの戻り値 • Event_Class = B#16#38, FAULT_ID = C2: 拡張ユニットの戻り値(予期される設定と実際の設定の不一致による発信中の拡張ユニットの障害) • Event_Class = B#16#39, FAULT_ID = C3:リモート I/O:DP マスタシステムの障害 • Event_Class = B#16#38, FAULT_ID = C4: DP ステーションの障害 • Event_Class = B#16#38, FAULT_ID = C5: DP ステーションの戻り値、ただしステーションに障害があります • Event_Class = B#16#38, FAULT_ID = C6:拡張ユニットの戻り値、ただしモジュールパラメータ割り当てにエラーがあります • Event_Class = B#16#38, FAULT_ID = C7: DP ステーションの戻り値、ただしモジュールパラメータ割り当てにエラーがあります • Event_Class = B#16#38, FAULT_ID = C8: DP ステーションの戻り値、ただし予期される設定と実際の設定の間に不一致があります • Event_Class = B#16#32/33, FAULT_ID = C9:「D_ACT_DP」を使用した DP スレーブの有効化/無効化 • Event_Class = B#16#39, FAULT_ID = CA:PROFINET IO システム障害 • Event_Class = B#16#39/38, FAULT_ID = CB: PROFINET IO ステーションの障害/ステーションの戻り値 • Event_Class = B#16#38, FAULT_ID = CC:PROFINET IO ステーションの戻り値、問題があるかメンテナンス中です • Event_Class = B#16#38, FAULT_ID = CD: PROFINET IO ステーションの戻り値、予期される設定と実際の設定の相違

			• Event_Class = B#16#38, FAULT_ID = CE:PROFINET IO ステーションの戻り値、モジュールパラメータ割り当てにエラーがあります • Event_Class = B#16#32/33, FAULT_ID = CF: 「D_ACT_DP」命令を使用した PROFINET IO デバイスの有効化/無効化 • Event_Class = B#16#39/38, FAULT_ID = F8:PROFINET I デバイスの一部のサブモジュールの障害/戻り値 • Event_Class = B#16#38, FAULT_ID = F9:PROFINET I デバイスの一部のサブモジュールの戻り値、デバイス構成に差異があります
address	WORD	20	TI_Cassic.データ構造体については、「address」パラメータを参照してください。
slv_prfl	BYTE	22	TI_Cassic データ構造体については、「slv_prfl」パラメータを参照してください。
intr_type	BYTE	23	TI_Cassic データ構造体については、「intr_type」パラメータを参照してください。
flags1	BYTE	24	TI_Cassic データ構造体については、「flags1」パラメータを参照してください。
flags2	BYTE	25	TI_Cassic データ構造体については、「flags2」パラメータを参照してください。
id	UINT	26	TI_Cassic データ構造体については、「id」パラメータを参照してください。
manufactur-er	UINT	28	TI_Cassic データ構造体については、「manufacturer」パラメータを参照してください。
instance	UINT	30	TI_Cassic データ構造体については、「instance」パラメータを参照してください。

プログラミングエラー OB/I/O アクセスエラー OB: データ構造 TI_ProgIOAccessError

パラメータ	データタイプ	Byte (バイト)	説明*
TI_ProgIOAccessError			プログラミングエラー OB および I/O アクセスエラー OB のデータ構造
SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス: • =121 (プログラミングエラー OB の場合) • =122 (I/O アクセスエラー OB の場合)
OB_Nr	UINT	2	DB 番号(1...32767)。
BlockNr	UINT	4	プログラミングエラーが発生したブロックの番号。
Reaction	USINT	6	• 0: エラーを無視 • 1: 不良な値を置換 • 2: コマンドをスキップ

Fault_ID	BYTE	7	エラーコード • B#16#21: BCD 変換エラー • B#16#22: 読み出し時の領域の長さエラー • B#16#23: 書き込み時の領域の長さエラー • B#16#24: 読み出し時の範囲エラー • B#16#25: 書き込み時の範囲エラー • B#16#26: タイマ番号のエラー • B#16#27: カウンタ番号のエラー • B#16#28: ビットアドレスがゼロでないポインタによるバイト、ワード、またはダブルワードへの読み取りアクセス • B#16#29: ビットアドレスがゼロでないポインタによるバイト、ワード、またはダブルワードへの書き込みアクセス • B#16#:30: 書き込み保護されているグローバル DB への書き込みアクセス • B#16#31: 書き込み保護されているインスタンス DB への書き込みアクセス • B#16#32: グローバル DB にアクセスしたときの DB 番号エラー • B#16#33: インスタンス DB にアクセスしたときの DB 番号エラー • B#16#34: FC 呼び出し中の番号エラー • B#16#35: FB 呼び出し中の番号エラー • B#16#42: I/O アクセスエラー、読み取り • B#16#43: I/O アクセスエラー、書き込み • B#16#3A: ロードされていない DB へのアクセス、DB 番号は許可済み領域にあります。 • B#16#3C: ロードされていない FC へのアクセス、FC 番号は許可されている領域にあります • B#16#3D: ロードされていない命令(SFC)へのアクセス、SFC 番号は許可されている領域にあります • B#16#3E: ロードされていない FB へのアクセス、FB 番号は許可されている領域にあります • B#16#3F: 使用不可の SFB へのアクセス、SFB 番号は許可されている領域にあります
BlockType	USINT	8	エラーが発生したブロックのタイプ: • OB: B#16#88 • FC: B#16#8C • FB: B#16#8E
Area	USINT	9	不正なアクセスが発生したメモリ領域: • ローカルデータ: B#16#40～4E、86、87、8E、8F、C0～CE • プロセスイメージ入力: B#16#01 • プロセスイメージ出力: B#16#02 • テクニカル DB: B#16#04 • I: B#16#81 • Q: B#16#82 • M: B#16#83

			• DB: B#16#84、85、8A、8B
DBNr	DB_ANY	10	DB 番号、AREA = DB の場合(グローバル DB またはインスタンス DB) プログラミングエラー OB の場合のみ。
Csg_OBNr	OB_ANY	12	OB 番号: • 121: プログラミングエラー OB • 122: I/O アクセスエラー OB
Csg_Prio	USINT	14	OB 優先度
Width	USINT	15	エラーが発生したアクセスのタイプ: • ビット: B#16#00 • バイト: B#16#01 • ワード: B#16#02 • DWORD: B#16#03 • LWORD: B#16#04
* 情報が読み取られる OB (I/O アクセスエラー OB またはプログラミングエラー OB)に応じて、特定の出力値のみが可能です。			

時刻割り込み OB: データ構造 TI_TimeOfDay

パラメータ	データタイプ	Byte (バイト)	説明
TI_TimeOfDay			時刻割り込み OB のデータ構造体
SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス(=10)
OB_Nr	UINT	2	DB 番号(1...32767)。
CaughtUp	BOOL	4	=TRUE、時刻が前へ調整されたため、その後に OB 呼び出しが実行された場合
Second-Time	BOOL	5	=TRUE、時刻が後ろへ調整されたため、OB が二度目に呼び出された場合 注記: SecondTime は、時刻が後ろへ設定される状況で 1 回のみ設定されます。

アイソクロナスモード割り込み OB: データ構造 TI_SynchCycle

パラメータ	データタイプ	Byte (バイト)	説明
TI_SynchCycle			アイソクロナスモード割り込み OB のデータ構造体

SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス(=61)
OB_Nr	UINT	2	DB 番号(1...32767)。
Initial_Call	BOOL	4	=TRUE、この OB の最初の呼び出し時 • STOP または HOLD から RUN への移行 • 再ロード後
IO _System	USINT	5	割り込みをトリガするリモート I/O システムの番号
Event_Count	INT	6	• = n: 失われたサイクルの数 • = -1: 不明な数のサイクルが失われました(周期が変わったなど)。
PIP_Input	BOOL	10	=TRUE: 入力の関連するプロセスイメージは最新です
PIP_Output	BOOL	11	=TRUE: 出力の関連するプロセスイメージは、最後のサイクル後まもなく、出力に転送されました
SyncCycle-Time	LTIME	16	計算されたサイクルタイム

製造元またはプロファイル固有の割り込みのステータス割り込み OB/更新割り込み OB: データ構造 TI_Submodule

パラメータ	データタイプ	Byte (バイト)	説明
TI_Submodule			製造元またはプロファイル固有の割り込みのステータス割り込み OB/更新割り込み OB のデータ構造体
SI_Format	USINT	0	開始情報の形式 • 16#FF:なし • 16#FE: 最適化開始情報
OB_Class	USINT	1	OB クラス • =55 (ステータス割り込み OB の場合) • =56 (更新割り込み OB の場合) • =57 (製造元またはプロファイル固有の割り込みの OB の場合)
OB_Nr	UINT	2	DB 番号(1...32767)。
LADDR	HW_IO	4	割り込みをトリガするコンポーネントのハードウェアアドレス
Slot	UINT	6	割り込みをトリガするコンポーネントのスロット番号
Specifier	WORD	8	割り込みフレームの割り込み指定子
address	WORD	20	TI_Classic.データ構造体については、「address」パラメータを参照してください。
slv_prfl	BYTE	22	TI_Classic データ構造体については、「slv_prfl」パラメータを参照してください。

intr_type	BYTE	23	TI_Cassic データ構造体については、「intr_type」パラメータを参照してください。
flags1	BYTE	24	TI_Cassic データ構造体については、「flags1」パラメータを参照してください。
flags2	BYTE	25	TI_Cassic データ構造体については、「flags2」パラメータを参照してください。
id	UINT	26	TI_Cassic データ構造体については、「id」パラメータを参照してください。
manufactur-er	UINT	28	TI_Cassic データ構造体については、「manufacturer」パラメータを参照してください。
instance	UINT	30	TI_Cassic データ構造体については、「instance」パラメータを参照してください。

パラメータ AINFO

PROFIBUS DP からの割り込みがあるコピー先の領域 AINFO のデータ構造体

バイト	意味	
0～3	ヘッダ情報、正確な説明については、次を参照してください。	
4～63	追加の割り込み情報: それぞれの割り込みのデータ:	
	分散型:	ARRAY[0] ～ ARRAY[59]

PROFIBUS DP からの割り込みがあるヘッダ情報の構造体

バイト	データタイプ	意味		
0	BYTE	受信された割り込み情報の長さ(バイト)		
		中央: 分散型:	4～224 4～63	
1	BYTE	中央:	予備	
		分散型:	割り込みの ID	
			1: 2: 3: 4: 5: 6: 31	診断割り込み プロセス割り込み 切断割り込み 接続割り込み ステータス割り込み 更新割り込み 拡張デバイスの障害、DP マスタシステムまたは DP ステーション
			32～126:	製造元固有の割り込み
2	BYTE	割り込みをトリガするコンポーネントのスロット番号		
3	BYTE	中央:	予備	
		分散型:	指定子	
			ビット 0 および 1:	0: 追加情報がありません、 1: 受信イベント、スロットの故障 2: 送信イベント、スロットの故障除去済み 3: 送信イベント、スロットの故障が継続中
			ビット 2:	Add_Ack
			ビット 3～7:	シーケンス番号

PROFINET IO または集中 I/O デバイスからの割り込みがあるコピー先の領域 AINFO のデータ構造体

バイト	意味
0～25	ヘッダ情報、正確な説明については、次を参照してください。
26～1431	追加の割り込み情報: 各割り込みの正規化された診断データ: ARRAY[0] ～ ARRAY[1405] 注記: 追加の割り込み情報も省略できます。

PROFINET IO または集中 I/O デバイスからの割り込みがあるヘッダー情報の構造体

バイト	データタイプ	意味
0 および 1	WORD	• ビット 0～7: ブロックタイプ • ビット 8～15: 予備
2 および 3	WORD	ブロック長
4 および 5	WORD	バージョン: • ビット 0～7: 下位バイト: • ビット 8～15: 上位バイト:
6 および 7	WORD	割り込みタイプの ID: • 1: 診断割り込み(着信) • 2: プロセス割り込み • 3: モジュール割り込みの削除 • 4: モジュール割り込みの挿入 • 5: ステータス割り込み • 6: 更新割り込み • 7: 冗長化割り込み • 8: スーパーバイザによる制御 • 9: スーパーバイザによるリリース • 10: 設定されたモジュールが挿入されていません • 11: サブモジュールの戻り値 • 12: 診断割り込み(発信) • 13: スレーブツースレーブ接続アラーム • 14: 近傍変更アラーム • 15: クロック同期メッセージ(バス側) • 16: クロック同期アラーム(デバイス側) • 17: ネットワークコンポーネントアラーム • 18: 時刻同期アラーム(バス側) • 19～31: 予備 • 32～127: 製造元固有の割り込み • 128～65535: 予備
8～11	DWORD	API (アプリケーションプロセス識別子)
12～13	WORD	割り込みをトリガするコンポーネントのスロット番号(値の範囲 0～65535)

14～15	WORD	割り込みをトリガするコンポーネントのサブモジュールのスロット番号(値の範囲 0～65535)
16～19	DWORD	モジュール識別、割り込みのソースの固有情報
20～23	DWORD	サブモジュール識別、割り込みのソースの固有情報
24～25	WORD	割り込み指定子。 • ビット 0～10:シーケンス番号(値の範囲 0～2047) • ビット 11:チャンネル診断: 0:使用できるチャンネル診断なし 1: チャンネル診断情報存在します • ビット 12:製造元固有診断のステータス: 0:使用できる製造元固有のステータス情報なし 1: 製造元固有のステータス情報入手可能 • ビット 13:サブモジュールの診断のステータス: 0:ステータス情報なし、すべてのエラーが修正済み 1: チャンネル診断および/またはステータス情報の少なくとも 1 つのアイテムが使用できます。 • ビット 14: 予備 • ビット 15: アプリケーション関係の診断状態: ○ 0: このアプリケーション関係内で設定されたモジュールのいずれも診断情報を報告しません ○ 1: この AR 内で構成されたモジュールの少なくとも 1 つが、診断情報を報告する。

PROFINET IO または集中 I/O からの割り込みに対する追加割り込み情報の構造体

PROFINET IO の追加の割り込み情報は、書式識別子によって異なります。 これは、同じまたは異なる書式識別子付きの複数のデータブロックで構成することができます。 次の書式識別子が使用可能です。

- W#16#0000 ～W#16#7FFF: 製造元固有の診断

バイト	データタイプ	意味
0～1	WORD	追加の割り込み情報としての役割を果たす次のデータ構造の書式識別子 W#16#0000 ～W#16#7FFF: 製造元固有の診断
2～n	BYTE	製造元のマニュアルを参照してください。

- W#16#8000: チャンネル診断
  チャンネル診断は、各 6 バイトのブロックで出力されます。 追加の割り込み情報(書式識別子なし)は、途切れたチャンネルのみについて出力されます。

バイト	データタイプ	意味
0～1	WORD	追加の割り込み情報としての役割を果たす次のデータ構造の書式識別子 W#16#8000: チャンネル診断
2～3	WORD	割り込みをトリガするコンポーネントのチャンネル番号(値の範囲: 0～65535):

<table>
<tr><td></td><td></td><td colspan="2">○ W#16#0000 ～W#16#7FFF: インターフェースモジュール/サブモジュールのチャンネル番号
○ W#16#8000:サブモジュール全体の汎用代替値
○ W#16#8001～W#16#FFFF: 予備</td></tr>
<tr><td rowspan="3">4</td><td rowspan="3">BYTE</td><td>ビット 0～2:</td><td>予備</td></tr>
<tr><td>ビット 3～4:</td><td>エラーのタイプ:
○ 0: 予備
○ 1: 着信エラー
○ 2: 発信エラー
○ 3: 発信エラー、他のエラーあり</td></tr>
<tr><td>ビット 5～7:</td><td>チャンネルのタイプ:
○ 0: 予備
○ 1: 入力チャンネル
○ 2: 出力チャンネル
○ 3: 入力/出力チャンネル</td></tr>
<tr><td>5</td><td>BYTE</td><td colspan="2">データ形式:
○ B#16#00:任意のデータ形式
○ B#16#01:ビット
○ B#16#02: 2 ビット
○ B#16#03: 4 ビット
○ B#16#04:バイト
○ B#16#05:ワード
○ B#16#06:ダブルワード
○ B#16#07: 2 ダブルワード
○ B#16#08 ～B#16#FF: 予備</td></tr>
<tr><td>6～7</td><td>WORD</td><td colspan="2">エラーのタイプ:
○ W#16#0000: 予備
○ W#16#0001:短絡
○ W#16#0002:電圧不足
○ W#16#0003:過電圧
○ W#16#0004:過負荷
○ W#16#0005:過熱
○ W#16#0006:断線
○ W#16#0007:上限値超過
○ W#16#0008:下限値超過
○ W#16#0009: エラー
○ W#16#000A ～W#16#000F: 予備
○ W#16#0010 ～W#16#001F: 製造元固有
○ W#16#0020 ～W#16#00FF: 予備
○ W#16#0100 ～w#16#7FFF: 製造元固有
○ W#16#8000:デバイス診断使用可能
○ W#16#8001 ～W#16#FFFF: 予備
すべてのエラータイプをサポートしていないチャンネルもあります。詳細情報については、特定のデバイスの診断データの説明を参照してください。</td></tr>
</table>

注記
「チャンネル番号」から「エラーのタイプ」までのセクションは、0～n 回発生します。

- W#16#8001: MULTIPLE (異なるタイプの診断情報が送信されます)。 この場合、追加の割り込み情報は可変長のブロックとして転送されます。

バイト	データタイプ	意味
0～1	WORD	追加の割り込み情報としての役割を果たす次のデータ構造の書式識別子 W#16#8001: 製造元固有の診断および/またはチャンネル診断
2～3	WORD	ブロックタイプ
4～5	WORD	ブロック長
6	BYTE	バージョン: 上位バイト:
7	BYTE	バージョン: 下位バイト:
8～11	DWORD	API (バージョンの下位バイト= 1 の場合のみ)
12～13	WORD	スロット番号
14～15	WORD	サブスロット番号
16～17	WORD	チャンネル番号
18～19	WORD	チャンネルプロパティ
20～21	WORD	書式識別子: ○ W#16#0000 ～W#16#7FFF: 製造元固有の診断 ○ W#16#8000:チャンネル診断 ○ W#16#8002:拡張チャンネル診断 ○ W#16#8003:段階的拡張チャンネル診断 ○ W#16#8004 ～W#16#80FF: 予備
22～n	BYTE	書式識別子に基づくデータ

注記
「ブロックタイプ」から始まるセクションは、1～n 回発生します。

- W#16#8002:拡張チャンネル診断

バイト	意味
0～1	書式識別子 W#16#8002
2～3	チャンネル番号
4～5	チャンネルプロパティ
6～7	エラータイプ
8～9	追加のエラー値
10～13	追加のエラー情報

- W#16#8003:段階的拡張チャンネル診断

バイト	意味
0～1	書式識別子 W#16#8003
2～3	チャンネル番号
4～5	チャンネルプロパティ
6～7	エラータイプ
8～9	追加のエラー値
10～13	追加のエラー情報

14～17	限定チャンネル修飾子

- W#16#8100:保守情報

バイト	意味
0～1	書式識別子 W#16#8100
2～3	ブロックタイプ
4～5	ブロック長
6～7	ブロック バージョン
8～9	予備
10～13	メンテナンス状態

注記
追加のアラーム情報の構造体の詳細情報については、*PROFIBUS DP から PROFINET IO までの SIMATIC PROFINET IO プログラミングマニュアル*および IEC 61158-6-10-1 現行バージョンを参照してください。

コピー先の領域 TINFO および AINFO

コピー先の領域 TINFO および AINFO

「RALRM」が呼び出されたそれぞれの OB によって、コピー先の領域 TINFO および AINFO は、部分的に書き込まれます。 次の表を参照して、入力される情報を特定してください。

割り込みタイプ	OB クラス	TINFO OB の開始情報	TINFO 管理情報	AINFO ヘッダ情報	AINFO 追加の割り込み情報	
プロセス割り込み	4x	可能	可能	可能	中央:	不可
					分散型:	PROFIBUS DP スレーブ/PROFI-NET IO デバイスでの提供による
ステータス割り込み	55	可能	可能	可能	可能	可能
更新割り込み	56	可能	可能	可能	可能	可能
製造元固有の割り込み	57	可能	可能	可能	可能	可能
I/O 冗長エラー	70	あり	あり	なし	なし	なし
診断割り込み	82	可能	可能	可能	中央:	PROFINET IO 標準に従った構造
					分散型:	PROFIBUS DP スレーブ/PROFI-NET IO デバイスでの提供による
割り込みの挿入/削除	83	可能	可能	可能	中央:	不可
					分散型:	PROFIBUS DP スレーブ/PROFI-NET IO デバイスでの提供による
モジュール削除割り込みの特殊な形態: スーパーバイザによる制御	83	可能	可能	可能	PROFINET IO のみ	
モジュール挿入割り込みの特殊な形態: スーパーバイザによる有効化	83	可能	可能	可能	PROFINET IO のみ	
未設定のモジュールの挿入	83	可能	可能	可能	PROFINET IO のみ	

ラック障害/ ステーション障害	86	あり	あり	なし	なし
... その他すべての OB		あり	なし	なし	なし

D_ACT_DP: DP スレーブの有効化/無効化

説明

「D_ACT_DP」命令を使用して、設定した DP スレーブ/PROFINET IO デバイスを個別に無効および有効にします。 さらに、それぞれの割り当てられた DP スレーブまたは PROFINET IO デバイスが現在有効か無効かを特定することができます。

この命令を使用してゲートウェイの IE/PB Link PN IO タイプを無効にした場合、すべての接続された PROFIBUS DP スレーブも、機能を停止します。 これらの障害は報告されます。

この命令は、DP/PA リンクで DP マスタシステムに接続されている PROFIBUS PA フィールドデバイスでは使用できません。

注記
1 つ以上の「D_ACT_DP」ジョブが有効である限り、プログラミングデバイスから CPU へ変更した設定を読み込めません(CiR の範囲内)。
継続している動作(CiR)中に、変更された設定がプログラミングデバイスから CPU に読み込まれると、CPU は「D_ACT_DP」ジョブの有効化を拒否します。

ジョブの無効化または有効化には、サイクルコントロールポイントを複数回通過する必要があります。 したがって、プログラミングループでこのようなジョブの終了を待つことはできません。

ファンクションの説明

"D_ACT_DP"は非同期で動作する命令で、その実行は複数の呼び出しにわたって行われます。 REQ=1 で「D_ACT_DP」を呼び出してジョブを開始します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

用途

実際には存在しない、または現在必要とされていない CPU で DP スレーブ/PROFINET IO デバイスを設定した場合でも、CPU は定間隔でこれらの DP スレーブ/PROFINET IO デバイスへのアクセスを継続します。 スレーブが無効になった後は、CPU のアクセスも停止します。 PROFIBUS DP では、こうすることで可能な限り最速の DP バスサイクルを実現でき、対応する エラーイベントが発生しなくなります。

例

機械 OEM の視点では、機械の量産では多数のデバイスのオプションがあります。 しかし、それぞれの納入された機械には、選択されたオプションの 1 つの組み合わせしかありません。

これらの可能な機械のオプションのそれぞれが、共通のユーザープログラムですべての可能なオプションを作成し維持するため、製造元によって DP スレーブ/PROFINET IO デバイスとして設定されます。 「D_ACT_DP」を使用して、機械のスタートアップ時に存在しなかったすべての DP スレーブ/PROFINET IO デバイスを無効にします。

同様に、機械ツールには多数のツールオプションがあるものの、同時に使用するのはそのうちのわずかであることです。 これらのツールは、DP スレーブ/PROFINET IO デバイスとして実装されます。「D_ACT_DP」を使用して、ユーザープログラムが現在必要なツールを有効にし、後に必要なツールを無効にします。

ジョブの識別

無効化または有効化ジョブを開始し、ジョブが完了する前に再度「D_ACT_DP」を呼び出すと、命令の動作は新規の呼び出しに同じジョブが含まれるかどうかによって異なります。 入力パラメータ LADDR が一致した場合、呼び出しは後続の呼び出しとみなされます。

DP スレーブ/PROFINET IO デバイスの無効化

「D_ACT_DP」で DP スレーブ/PROFINET IO デバイスを無効にした場合、そのプロセス出力は設定した代替値または 0 (セーフ状態)に設定されます。 割り当て済み DP マスタ/PROFINET IO コントローラは、このコンポーネントのアドレス指定を継続します。 無効になった DP スレーブ/PROFINET IO デバイスは、DP マスタ/PROFINET IO コントローラのエラー LED または CPU によって故障または不明である場合には識別されません。

無効になった DP スレーブ/PROFINET IO デバイスの入力のプロセスイメージは、0 で更新されます。つまり、これは障害の発生した DP スレーブ/PROFINET IO デバイスと同様に扱われます。

使用しているプログラムを使用して、無効になった DP スレーブ/PROFINET IO デバイスのユーザーデータに直接アクセスすると、I/O アクセスエラー OB が呼び出され、対応するイベントの開始が診断バッファに入力されます。 命令(「RD_REC」など)を使用して、無効になった DP スレーブ/PROFINET IO デバイスにアクセスしようとすると、RET_VAL で使用できない DP スレーブ/PROFINET IO デバイスと同じエラー情報を受け取ります。

DP スレーブ/PROFINET IO デバイスを無効にしても、その入力または出力が更新されるシステム側のプロセスイメージに属する場合でも、プログラムエラー OB は開始されません。 また、診断バッファにはエントリはありません。

「D_ACT_DP」で無効にした後、DP ステーション/PNIO ステーションに障害が発生すると、オペレーティングシステムはこの障害を検出しません。

PROFIBUS DP に適用: スレーブツースレーブ通信でトランスミッタとして機能している DP スレーブを無効にする場合、最初にどの入力データをトランスミッタがその DP マスタに転送しているかを検出するレシーバ(リスナー)を無効にすることを推奨します。 トランスミッタは、必ずこのステップを実行した後に無効にしてください。

DP スレーブ/PROFINET IO デバイスの有効化

「D_ACT_DP」で DP スレーブ/PROFINET IO デバイスを再度有効にする場合、このコンポーネントは、関連する DP マスタ/PROFINET IO コントローラによって設定され、パラメータを割り当てられます(障害が発生した DP ステーション/PROFINET IO ステーションの戻り値と同様)。 この有効化は、コンポーネントがユーザーデータを転送できるようになると完了します。

DP スレーブ/PROFINET IO デバイスを有効にしても、その入力または出力が更新されるシステム側のプロセスイメージに属する場合でも、プログラムエラー OB は開始されません。 また、診断バッファにはエントリはありません。

アクセスできない「D_ACT_DP」のある DP スレーブ/PROFINET IO デバイス(たとえば、バスから物理的に分離されたため)を有効にしようとする場合、リモート I/O の設定済みパラメータ割り当て時間の経過後に、この命令により、エラーコード W#16#80A7 が返されます。DP スレーブ/PROFINET IO デバイスが有効になり、有効になった DP スレーブ/PROFINET IO デバイスにアクセスできないことが、対応する表示をシステム診断にもたらします。

DP スレーブ/PROFINET IO デバイスが後で再度アクセス可能な場合、標準的なシステムの動作(たとえば、このために設定された OB の呼び出し)が実行されます。

注記
DP スレーブ/PROFINET IO デバイスの有効化には、時間がかかる場合があります。 現在の有効化ジョブをキャンセルする場合は、LADDR の同じ値と MODE = 2 で「D_ACT_DP」を再度開始します。RET_VAL = 0 で有効化ジョブのキャンセルの成功が示されるまで、MODE = 2 で「D_ACT_DP」の呼び出しを継続します。

スレーブツースレーブ通信を行う DP スレーブを有効にする場合、最初にトランスミッタを有効にし、その後にレシーバ(リスナー)を有効にすることを推奨します。

パラメータ

以下の表に、「D_ACT_DP」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	レベルトリガの制御パラメータ REQ=1: 有効化または無効化を実行
MODE	Input	USINT	I、Q、M、D、L、または定数	ジョブ識別子 可能な値: • 0: アドレス指定されたコンポーネントが有効または無効になったかどうかについての情報を要求 (RET_VAL パラメータ経由の出力) • 1: DP スレーブ/PROFINET IO デバイスを有効化 • 2: DP スレーブ/PROFINET IO デバイスを無効化
LADDR	Input	HW_DEVICE	I、Q、M、D、L、または定数	DP スレーブ(HW_DPSlave)/PROFINET IO デバイス(HW_Device)のハードウェア識別子 この数は、ネットワークビューの DP スレーブ/PROFINET IO デバイスのプロパティから、または標準タグテーブルの[システム定数]タブから取得することができます。 そこで、デバイス診断用の ID と動作ステータス移行用の ID の両方が指定されている場合、デバイス診断用のコードを使用する必要があります。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
BUSY	Output	BOOL	I、Q、M、D、L	有効コード: • BUSY=1: ジョブがまだ有効です。 • BUSY=0: ジョブが終了しました。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	ジョブがエラーなしで完了済み。

0001	DP スレーブ/PROFINET IO デバイスが有効(このエラーコードは MODE = 0 のみで使用可)。
0002	DP スレーブ/PROFINET IO デバイスが無効(このエラーコードは MODE = 0 のみで使用可)。
7000	REQ=0 による最初の呼び出し。LADDR で指定されたジョブが有効ではありません。BUSY の値は「0」です。
7001	REQ=1 による最初の呼び出し。LADDR で指定されたジョブが開始されました。BUSY の値は「1」です。
7002	中間呼び出し(REQ は対象外)。 有効にしたジョブがまだ有効です。BUSY の値は「1」です。
8090	• LADDR で指定されたアドレスでモジュールを設定していません。 • 使用している CPU を I スレーブとして操作し、LADDR でこの I スレーブのアドレスを指定しました。
8092	現在アドレス指定された DP スレーブ/PROFINET IO デバイスの無効化(MODE=2)は、有効化(MODE=1)によってキャンセルできません。 後でコンポーネントを有効にしてください。
8093	有効化または無効化が可能な DP スレーブ/PROFINET IO デバイスが、LADDR で指定されたアドレスに割り当てられています。
8094	ツール交換ポートの潜在的なパートナーであるデバイスを有効にしようとしました。しかし、現在このツール交換ポートで他のデバイスが有効になっています。 有効になったデバイスは、有効のままになります。
80A0	CPU と IO コントローラ間の通信中にエラーが発生しました。
80A1	アドレス指定されたコンポーネントにパラメータを割り当てられませんでした。 (このエラーコードは、MODE = 1 の場合のみ使用可能です。) 注記: "D_ACT_DP」は、このコンポーネントの有効になったスレーブ/デバイスにパラメータ割り当て中に再度障害が発生した場合のみ、このエラー情報を返します。単一のモジュールのパラメータ割り当てに失敗した場合、「D_ACT_DP」はエラー情報 W#16#0000 を返します。
80A2	アドレス指定された DP スレーブは、確認を返しません。 (このエラー情報は、PROFINET IO デバイスでは使用できません。 PROFINET では、プロセスは時間をモニタされていません。)
80A3	関係する DP マスタ/PROFINET IO コントローラがこのファンクションをサポートしていません。
80A4	CPU が外部 DP マスタ/PROFINET IO コントローラでこのファンクションをサポートしていません。
80A6	DP スレーブ/PROFINET IO デバイスのスロットエラー。アクセスできないユーザーデータがあります(このエラーコードは、MODE=1 の場合のみ使用できます)。 注記: "「D_ACT_DP」は、パラメータ割り当て後、「D_ACT_DP」の最後よりも前に有効になったコンポーネントに再度障害が発生した場合のみ、このエラー情報を返します。 単一のモジュールのみが使用できない場合は、「D_ACT_DP」はエラー情報 W#16#0000 を返します。
80A7	アクセス不可能なデバイスの有効化。
80AA	DP スレーブ/PROFINET IO デバイスでのエラーを有効化 コンフィグレーションによる違い
80AB	DP スレーブ/PROFINET IO デバイスでのエラーを有効化 パラメータ割り当てエラー
80AC	DP スレーブ/PROFINET IO デバイスでのエラーを有効化 メンテナンス必須

80C1	"「D_ACT_DP」が開始され、他のアドレスで続行されています(このエラーコードは MODE=1、および MODE=2 の場合に出力されることがあります)。
80C3	• 一時的なリソースエラー: CPU が現在、最大可能な有効化および無効化ジョブ(8)を処理中です。 (このエラーコードは、MODE = 1 および MODE = 2 の場合のみ使用可能です。) • 修正された設定の受信のために CPU が使用中です。 現在、DP スレーブ/PROFINET IO デバイスの有効化/無効化を行えません。
80C5	DP: ユーザーによって収集されていないジョブは再起動で破棄されます。
80C6	PROFINET: ユーザーによって収集されていないジョブは再起動で破棄されます。
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

ReconfigIOSystem: IO システムの再構成

説明

「ReconfigIOSystem」命令を使用して、データレコードを CPU の PROFINET インターフェースに転送します。 このデータレコード (「CTRLREC」パラメータ) は、設定を制御するための次の情報を含みます。

- 有効にするオプションの IO デバイスのリスト
- 指定するパートナーポートのリスト、IO デバイスのポートプロパティに[パートナーはユーザープログラムに設定]オプションを設定している場合。

注記

「ReconfigIOSystem」命令は、MODE1 および MODE3 の内部で「D_ACT_DP」命令を使用して IO デバイスを有効/無効にします。 したがって、この命令の説明に記載されている規則と注記を遵守してください。

「D_ACT_DP: リモート I/O デバイスの無効化/有効化」も参照してください。

ファンクションの説明

「ReconfigIOSystem」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 REQ=1 で「ReconfigIOSystem」を呼び出してジョブを開始します。

出力パラメータ STATUS および BUSY は、ジョブのステータスを示します。

パラメータ

以下の表に、「ReconfigIOSystem」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	入力	BOOL	I、Q、M、D、L、または定数	エッジトリガの制御パラメータ REQ = 1: データ転送の実行
MODE	入力	UINT	I、Q、M、D、L、または定数	MODE パラメータを使用して命令の動作方法を制御します。 下記のテーブルに機能の詳細な説明が記載されています。 次の値が返されます。 • 1: 変換フェーズで IO システムのすべての IO デバイスを無効にします。 • 2: データレコードの設定に従って IO システムを再構成します (CTRLREC)。 • 3: 再構成後に IO システムのすべての IO デバイスを有効にします。
LADDR	入力	HW_INTER-FACE	I、Q、M、D、L、または定数	PROFINET インターフェース(IO コントローラ)のハードウェア識別子

CTRLREC	入力	VARIANT	I、Q、M、D、L	IO システムの実際の設定を制御するためのデータレコード
DONE	出力	BOOL	I、Q、M、D、L	0: 命令がまだ完了していません。 1: 命令が完了しました。
BUSY	出力	BOOL	I、Q、M、D、L	有効コード: 0: 命令が有効 1: 命令が完了しました。
ERROR	出力	BOOL	I、Q、M、D、L	0: 命令がエラーなしで完了 1: 命令がエラーありで完了
STATUS	出力	DWORD	I、Q、M、D、L	結果/エラーコード
ERR_INFO	出力	WORD	I、Q、M、D、L	直前にエラーが発生した IO デバイスの HW ID

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

パラメータ MODE

MODE パラメータは、以下の値を取ることができます。

MODE	説明
1	モード 1 で命令を呼び出すと、IO システムのすべての IO デバイスを無効にできます。「ReconfigIOSystem」命令は、内部的に「D_ACT_DP"」命令を使用します。"「ReconfigIOSystem」は、次の出力パラメータの D_ACT_DP によって検出されるエラーを返します。 • STATUS (エラーコード) • ERR_INFO (エラーが発生した IO デバイスのハードウェア識別子)。 STATUS および ERR_INFO で、CPU は最後に特定されたエラー/HW 識別子を入力し、それにより既存のエラーコードを上書きします。 そのため、入力されたエラー以外にもエラーが存在する可能性があります。
2	IO システムの実際の設定を制御するため、命令はデータレコードを PROFINET インターフェースに転送します。PROFINET インターフェースは、LADDR を使用してアドレス指定されます。 データレコードの構造に関する情報は、次のセクションを参照してください。 エラー分析のための STATUS パラメータについては、次のセクションを参照してください。
3	IO システムのオプションなしのすべての IO デバイス、および CTRLREC に記載されているオプションの IO デバイス制御データレコードは有効です。 CTRLREC データレコードに記載されて**いない**オプションの IO デバイスは無効です。 ドッキングユニットの一部を構成する IO デバイスが CTRLREC 制御データレコードに記載されている場合、PN IO システムは次のように応答します。 • **ドッキングユニット**の IO デバイスは、ReconfigIOSystem がモード 3 で呼び出される場合は無効です。 この応答は、構成によって制御される IO デバイスが存在しない構成の応答に対応しています。 ドッキングユニットの IO デバイスは既定で無効に設定されているため、ユーザープログラムで有効にする必要があります。

制御データレコードの構造体

制御データレコード(「CTRLREC」パラメータ)を使用して、オプションで構成される IO デバイスが実際の IO システムで使用可能で、ポートの相互接続が設定される CPU の PROFINET インターフェースに通知することができます。

これには、IO システムの構成で調整することが可能な構成が必要となります。

- 制御データレコードに記載されている IO デバイスは、「オプションの IO デバイス」として有効にする必要があります(IO デバイスのプロパティ: PROFINET インターフェース[X1]|詳細オプション|インターフェースオプション)。
- 制御データレコードに記載されているポートの相互接続は、パートナーポートが適切なポートとして「パートナーはユーザープログラムによって設定」されている場合のみ可能です。

パラメータ CTRLREC のデータタイプは「VARIANT」です。 「CTRLREC」制御データレコードは、以下の構造を持っています。

- Word データタイプのエレメントを持つ配列

下記に「CTRLREC」の基本的な構造が Word エレメントデータタイプを使用して記載されています。

ネットワークビューまたはデバイスビューの[システム定数]タブで必要なハードウェア識別子を直接読み出します。 これを行うには、ネットワークビュー(IO デバイス)またはデバイスビュー(PROFINET インターフェース)でオブジェクトを選択します。

推奨事項: HW 識別子の名前を使用して、HW 識別子の記号アドレス指定を使用してください。

名前	データタイプ	コメント
Version_High, Version_Low	Word	制御データレコードのバージョン(上位バイト: 01 下位バイト: 00
Number_of_opt_Devices_used	Word	実際の IO システムの構成で使用されるオプションの IO デバイスの番号。 下記のリストに記載されていないオプションの IO デバイスは、無効になっています。
Activate_opt_Device_1	Word / Hw_Device	実際の構成で存在するオプションの IO デバイスは、ハードウェア識別子と共にリストに記載する必要があります。 IO デバイスオブジェクトのシステム定数を使用します。 例: IO デバイスオブジェクトの名前は「IO-Device_4~IODevice」であり、タイプは「Hw_Device」です。
Activate_opt_Device_2	Word / Hw_Device	2 番目のオプションの IO デバイス(値が 262 など)
...	...	...
Activate_opt_Device_n	Word / Hw_Device	n 番目のオプションの IO デバイス(値が 282 など)
Number_of_Port_Interconnections_used	Word	下記のリストに記載されたポート相互接続の番号。 ポート相互接続を指定しない場合は、「0」を入力します。

		「パートナーはユーザープログラムによって設定」され、下記のリストに記載されていないすべてのポートでは、CPU は「任意のパートナー」設定を使用します。
Port_Interconnection_1_Local	Word / Hw_Interface	1 番目のポートの相互接続、ローカルポートの HW 識別子 ポートオブジェクトのシステム定数を使用します。 例: ポートオブジェクトの名前は「IO-Device_2~PROFINET_interface~Port_2」であり、タイプは「Hw_Interface」です。
Port_Interconnection_1_Remote	Word / Hw_Interface	1 番目のポートの相互接続、パートナーポートの HW 識別子
Port_Interconnection_2_Local	Word / Hw_Interface	2 番目のポートの相互接続、ローカルポートの HW 識別子
Port_Interconnection_2_Remote	Word / Hw_Interface	2 番目のポートの相互接続、パートナーポートの HW 識別子
...	...	...
Port_Interconnection_n_Local	Word / Hw_Interface	n 番目のポートの相互接続、ローカルポートの HW 識別子
Port_Interconnectio_n_Remote	Word / Hw_Interface	n 番目のポートの相互接続、パートナーポートの HW 識別子

STATUS パラメータ

STATUS 出力パラメータには、エラー情報が含まれます。 考えられるエラーコードの詳細なリストが次のセクションにあります。 STATUS が ARRAY[1...4] of BYTE と解釈された場合、エラー情報は次の構造になります。

フィールドエレメント	名前	意味
STATUS[1]	Function_Num	B#16#DF: データレコードの書き込みエラー、それ以外の場合は B#16#00。
STATUS[2]	Error_Decode	エラーの場合は B#16#80 が出力されます(データレコードの読み出し/書き込みの状況では IEC 61158-6 に対応します)。
STATUS[3]	Error_Code_1	B#16#AA データレコードの構造体のエラーの場合
STATUS[4]	Error_Code_2	製造元固有のエラー ID 拡張: • B#16#00 データレコードのエラー(バイト値の不正なパディング(0 でない場合) • B#16#01 予備 • B#16#02 データレコードの IO デバイスの少なくとも 1 つのステーション番号が無効です(未設定、オプションなしの IO デバイスを指定、または IO コントローラの値が 0)。 • B#16#03 データレコードで指定された少なくとも 1 つのパートナーポートが無効です。 例: ○ パートナーポートのサブスロットアドレスが使用不可

		○ パートナーポートの設定が不正(正しい設定:「パートナーはユーザープログラムによって設定」) ○ パートナーポートが無効な IO デバイスに属しています。 • B#16#10 CTRLREC 制御データレコードのバージョンが無効(指定されたバージョンはサポートされていません) • B#16#11 CTRLREC 制御データレコードで有効にするオプションの IO デバイスの番号がサポートされていません。 • B#16#12 CTRLREC 制御データレコードで指定された相互接続(「パートナーをユーザープログラムで設定」)の番号がサポートされていません。 • B#16#13 ハードウェア識別子からデバイス番号への内部変換が失敗しました。 ERR_INFO 出力パラメータには、エラーが発生したデバイスのハードウェア識別子が含まれています。 • B#16#14 整合性エラー: CTRLREC 制御データレコードの長さが、制御データレコードの情報を一致しません。例: 20 のオプション IO デバイスが指定されていますが、制御データレコードの長さが 10 バイトしかありません。

エラーコード(STATUS パラメータ)

エラーコード	説明
16#0000_0000	ジョブがエラーなしで完了済み
16#0070_0000	有効なジョブなし
16#0070_0100	命令の最初の呼び出し
16#0070_0200	次の命令呼び出し(命令がまだ実行中、BUSY = 1)
16#0080_8000	MODE がサポートされていません
16#0084_5100	CRTLREC データレコードのデータタイプが不正です。 Word の配列を使用してください。
16#0080_9100	LADDR パラメータが PROFINET インターフェースをアドレス指定していません(存在しない、または間違ったタイプ、PROFIBUS インターフェースなど)。 PROFINET インターフェースは IO システムの構成制御をサポートしていません。
16#0080_Cx00	リソースの一時的な不足などによる一時的なエラー
16#DF80_AAxy	データレコードの構造体のエラー(MODE 2)。「xy」の意味については、上記の STATUS[4] Error_Code_2 の定義を参照してください。
16#DF80_B600	オプションの IO デバイスが構成されておらず、「パートナーをユーザープログラムで設定」によってポートが割り当てられていないため、構成制御が不可能。 この構成は命令呼び出しの必要条件です。
16#0080_9400 16#0080_A000 16#0080_A700 16#0080_AA00 16#0080_AB00 16#0080_AC00	内部的に呼び出された D_ACT_DP 命令のエラーコードが転送されました。 これらのエラーコードの意味については、「D_ACT_DP」を参照してください。 エラーが発生した IO デバイスのハードウェア識別子は、ERR_INFO(入力は後続のエラーによって連続して上書きされます)に入力されます。 複数の IO デバイスが関連する場合、STEP 7 によるオンライン診断を推奨します。 IRT 構成の場合: IO デバイスのデバイス番号は、相互接続トポロジの昇順で IO コントローラの後に続きます。詳細は、ここを参照してください。

その他

この章には下記に関する情報が記載されています：

- RD_REC: I/O からデータレコードの読み出し (S7-1500)
- WR_REC: データレコードを I/O に書き込み (S7-1500)
- DPRD_DAT: DP スレーブの整合性のあるデータ読み出し (S7-1200, S7-1500)
- DPWR_DAT: DP スレーブのコンスタントデータの書き込み (S7-1200, S7-1500)
- iDevice/iSlave (S7-1500)
- PROFIBUS (S7-1200, S7-1500)
- ASI (S7-1200, S7-1500)

RD_REC: I/O からデータレコードの読み出し

説明

この命令を使用して、アドレス指定されたモジュールから番号 RECNUM を持つデータレコードを読み出します。 読み出しプロセスを開始するには、呼び出し中に入力パラメータ REQ に値「1」を割り当てます。 読み出しプロセスを直ちに実行できる場合、この命令が出力パラメータ BUSY で値「0」を返します。 BUSY が値「1」の場合、読み出しが未完了です。

関連項目:非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味. データ転送にエラーがなかった場合、読み出されたデータレコードは、RECORD パラメータで範囲が決められたコピー先の領域に入力されます。

注記

1997 年 2 月以前に購入した FM または CP (以下「旧モジュール」と呼びます)から 1 を超える番号のデータレコードを読み出す場合、「RD_REC」は新型モジュールの場合とは異なる応答をします。 この特殊な状況については、「データレコード番号 >1 の旧型 S7-300 FM および CP の使用」セクションで説明しています(以下参照)。

DPV1 スレーブを GSD ファイル(GSD rev. 3 以降)を使用して設定し、DP マスタの DP インターフェースが「S7 互換」に設定されている場合、「RD_REC」を使用してユーザープログラムの I/O モジュールからデータレコードを読み出すことができない場合があります。 この場合、DP マスタが誤ったスロットをアドレス指定しています(設定したスロット + 3)。

対策: DP マスタのインターフェースを「DPV1」にセットします。

パラメータ

以下の表に、「RD_REC」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: 読み出し要求
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	モジュールのハードウェア識別子。
RECNUM	Input	BYTE	I、Q、M、D、L、または定数	データレコード番号(許可される値: 0 ～240)
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。 追加事項: 実際に転送されたデータレコード長(バイト) (許可される値: +1 ～+240、コピー先の領域が転送されたデータレコードよりも大きく、転送中にエラーが発生しなかった場合)。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: 読み出しプロセスが未完了です。
RECORD	Output	ANY	I、Q、M、D、L	読み出したデータレコードのコピー先の領域です。 「RD_REC」の非同期実行の場合、すべての呼び出しで RECORD の実際のパラメータが同じ長さ

				の情報を持つことを確認してください。 BYTE データタイプのみが許可されます。 注記: S7-300 CPU の場合、パラメータ RECORD には、DB パラメータを完全に指定する必要があることに注意してください(たとえば、P#DB13.DBX0.0 のバイト 100)。 明示的な DB 番号の省略は、S7-300 CPU のみで許可され、ユーザープログラムでエラーメッセージが表示されます。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RECORD パラメータ

注記
データレコード全体が必ず読み出されるようにするには、長さ 241 バイトのコピー先の領域を選択します。 データ転送にエラーがない場合、RET_VAL に実際のデータレコード長が含まれます。

データレコード番号 > 1 の旧型の S7-300 FM および CP の使用

旧型 S7-300 FM または旧型 S7-300 CP から 1 よりも大きな番号のデータレコードを読み出すために「RD_REC」命令を使用する場合、次の点に注意してください。

- コピー先の領域が必要なデータレコードの実際の長さよりも大きい場合、RECORD にはデータが入力されません。 RET_VAL が W#16#80B1 で書き込まれます。
- コピー先の領域が必要なデータレコードの長さよりも小さい場合、RECORD の長さ情報で指定されたバイト数だけ CPU がレコードの先頭から読み出しを行い、このバイト数を RECORD に入力します。RET_VAL の値は「0」になります。
- RECORD で指定された長さが必要なデータレコードの長さと同じ場合、CPU がデータレコードを読み出し、RECORD に入力します。「0」で RET_VAL が書き込まれます。

RET_VAL パラメータ

- ファンクションの実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
- 転送中にエラーが発生しなかった場合、RET_VAL には次が含まれます。
  - コピー先の領域全体が選択されたデータレコードからのデータで埋められた場合(データレコードが不完全な場合もあります)は 0。
  - 実際に転送されたデータレコード長(バイト) (許可される値:+1 ~ +240、コピー先の領域が転送されたデータレコードよりも大きい場合)

    注記
    一般エラー W#16#8745 が発生した場合、これはプロセスイメージの少なくとも 1 バイトへのアクセスがブロックされたことのみを示します。 データレコードはモジュールに正しく読み出され、I/O メモリ領域に書き込まれています。

次の表で「本当の」エラー情報(エラーコード W#16#8xyz)を参照する場合、2 つの場合を分けて考えます。

- 一時的なエラー(エラーコード W#16#80A2 ~ 80A3、80Cx):

  このタイプのエラーは、ユーザーの操作なしで解決する可能性があります。つまり、命令を再度呼び出してみると便利です(必要に応じて、複数の呼び出し)。

  一時的なエラーの例: 必要なリソースが現在使用中である場合(W#16#80C3)。

- 永続的なエラー(エラーコード W#16#809x、80A0、80A1、80Bx):

  このタイプのエラーは、自然に解決しません。 エラーを解決するまで、命令を新たに呼び出しても成功しません。 永続的なエラーの例: RECORD での不正な長さ指定(W#16#80B1)。

注記

「WR_REC」を使用して DPV1 スレーブにデータレコードを転送した場合、または RD_REC を使って DPV1 スレーブからデータレコードを読み出している場合でこの DPV1 スレーブが DPV1 モードで作動している場合、DP マスタはスレーブから受信したエラー情報を次のように評価します。

エラー情報が W#16#8000～W#16#80FF、または W#16#F000～W#16#FFFF の範囲にある場合、DP マスタはエラー情報を命令に渡します。 エラー情報がこの範囲外の場合、DP マスタは値 W#16#80A2 を命令にし、スレーブを保留します。

DPV1 スレーブからのエラー情報の説明については、「STATUS[3] パラメータ STATUS」を参照してください。

WR_REC と RD_REC のパラメータ RET_VAL

エラーコード* (W#16#...)	説明	制限事項
0000	エラーは発生していません。	-
7000	REQ=0 による最初の呼び出し: 有効なデータ転送がありません。BUSY の値は 0 です。	-
7001	REQ=1 による最初の呼び出し: データ転送が開始されました。BUSY の値は 1 です。	リモート I/O
7002	中間呼び出し(REQ は対象外): データ転送が既に有効です。BUSY の値は 1 です。	リモート I/O
8090	ADDR パラメータで指定されたアドレスが無効です。	-
8092	BYTE 以外のタイプが ANY 参照で指定されています。	-
8093	この命令は、LADDR および IOID を使用して選択したモジュールでは許可されません。	-
80A0	モジュールからの読み出し時の否定確認: 読み出しプロセス中にモジュールが取り外されたか、または故障しています。	「RD_REC」のみ
80A1	モジュールへの書き込み時の否定確認: 書き込みプロセス中にモジュールが取り外されたか、故障しています。	「WR_REC」のみ
80A2	• レイヤー 2 での DP プロトコルエラー(たとえば、スレーブ故障またはバスの問題) • ET 200S の場合、DPV0 モードでデータレコードが読み出せません。	リモート I/O
80A3	ユーザーインターフェース/ユーザーによる DP プロトコルエラー	リモート I/O
80A4	PROFIBUS の通信が中断されました。	-
80B0	• モジュールタイプで命令が使用できません。 • モジュールがデータレコードを認識しません。 • データレコード番号 241 は許可されていません。 • 「WR_REC」の場合、データレコード 0 および 1 は許可されていません。	-

80B1	RECORD パラメータで指定された長さが不正です。	• 「WR_REC」の場合: 長さが不正 • 「RD_REC」の場合(旧型の S7-300 FM および S7-300 CP を使用する場合のみ): 指定長さ > データレコード長さ • DPNRM_DG の場合: 指定された長さ < データレコード長
80B2	設定されたスロットが割り当てられていません。	-
80B3	実際のモジュールタイプが指定されたモジュールタイプと一致していません。	-
80B5	DP スレーブまたはモジュールの準備ができていません。	-
80B7	DP スレーブまたはモジュールがパラメータまたは値の無効な範囲を報告します。	「RD_REC」のみ
80C0	「WR_REC」の場合: データは、CPU が STOP モードの場合のみ書き込み可能です。 注記: つまり、ユーザープログラムによる書き込みは不可能です。 データの書き込みは、PG/PC を使用したオンラインのみで可能です。 「RD_REC」の場合: モジュールはデータレコードをルーティングしますが、データが存在しないか、CPU が STOP モードの場合のみにデータを読み出すことができます。 注記: CPU が STOP モードの場合のみにデータを読み出すことができる場合、ユーザープログラムによる評価は不可能です。 この場合、データの読み出しは、PG/PC を使用したオンラインのみで可能です。 「DPNRM_DG」の場合: 使用可能な診断データがありません。	「WR_REC」または「RD_REC」または「DPNRM_DG」の場合
80C1	モジュールで、同じデータレコードの前の書き込みジョブのデータがモジュールによって処理されていません。	-
80C2	モジュールは、現在 CPU で可能な最大数のジョブを処理しています。	-
80C3	必要なリソース(メモリなど)が現在占有されています。	-
80C4	内部の一時的なエラー。 ジョブを実行できませんでした。 ジョブを繰り返します。 このエラーが頻発する場合は、据付けを点検して電気的な干渉の発生源を探します。	-
80C5	リモート I/O が使用不可です。	リモート I/O
80C6	優先度クラス中止(再起動またはバックグラウンド)のため、データレコードの転送が停止されました。	リモート I/O
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得	-
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

WR_REC: データレコードを I/O に書き込み

説明

「WR_REC」命令を使用して、アドレス指定されたモジュールにデータレコード RECORD を転送します。

転送するデータは、最初の呼び出し中に RECORD パラメータから読み出されます。 データレコードの転送に 1 回の呼び出しよりも長い時間がかかる場合、RECORD パラメータの内容は後の命令呼び出しで関連性が失われます(同じジョブの場合)。

書き込みプロセスを開始するには、呼び出し中に入力パラメータ REQ に値「1」を割り当てます。 書き込みプロセスを直ちに実行できる場合、この命令が出力パラメータ BUSY で値「0」を返します。 BUSY が値「1」の場合、書き込みが未完了です。

パラメータ

次の表に、「WR_REC」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数	REQ = 1: 書き込み要求
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	モジュールのハードウェア識別子。
RECNUM	Input	BYTE	I、Q、M、D、L、または定数	データレコード番号
RECORD	Input	VARIANT	I、Q、M、D、L	データレコード
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: 書き込みプロセスが未完了です。

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

関連項目 RD_REC: I/O からデータレコードの読み出し

注記
一般エラー W#16#8544 が発生した場合、データレコードを含む I/O メモリ領域の少なくとも 1 バイトへのアクセスが拒否されたことのみを示します。 データ転送は継続されました。

DPRD_DAT: DP スレーブの整合性のあるデータ読み出し

説明

命令「DPRD_DAT」を使用して、I/O モジュールから矛盾のないデータを読み出します。

この命令は、基本モジュール、DP 標準スレーブ、および PROFINET IO デバイスの場合に使用できます。

I/O またはプロセスイメージ入力テーブルにアクセスする読み込みコマンドを使って読み出せるのは最大で連続する 4 バイトであるため、「DPRD_DAT」が必要です。 必要に応じて、入力のプロセスイメージを使って整合性のあるデータを読み出すこともできます。 使用している CPU がこの機能をサポートするかの確認については、関連する文書を参照してください。 DP 標準スレーブ/PROFINET IO デバイスの整合性のあるデータの追加情報については、セクション「データの一貫性」を参照してください。

必要に応じて、「DPRD_DAT」命令を 1 倍と以上のデータ領域に使用することも可能です。 データの最大長に関する情報については、お使いの CPU のマニュアルを参照してください(たとえば、S7-1214 では 64 バイト)。

- LADDR パラメータを使用して、DP 標準スレーブ/PROFINET IO デバイスのモジュールを選択します。 アクセスエラーが発生すると、エラーコード W#16#8090 が出力されます。
- RECORD パラメータを使用し、読み出されたデータのターゲット範囲を定義します。
  - ターゲット範囲は、少なくとも選択したモジュールの入力と同じ長さであることが必要です。 入力のみが転送され、他のバイトは考慮されません。 モジュール型構成または複数の DP 識別子を持つ DP 標準スレーブから読み出しを行う場合、アクセスできるのは 1 回の「DPRD_DAT」呼び出しごとに設定されたハードウェア識別子のモジュールのデータのみです。 選択したターゲット範囲が小さすぎる場合、エラーコード 80B1 が RET_VAL パラメータに出力されます。
  - すべてのビット列およびすべての整数がデータタイプとして使用できます。 これらのデータタイプを ARRAY タイプのデータ構造内で使用することも可能です。 データタイプ STRING はサポートされません。
- データ転送中にエラーがなかった場合、読み出されたデータは RECORD パラメータで定義されたターゲット範囲に入力されます。

パラメータ

次の表に、「DPRD_DAT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IO	I、Q、M、L、または定数	データの読み出し元のモジュールのハードウェア ID。 このハードウェア ID は、デバイスビューのモジュールのプロパティ、または、システム定数に存在します。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
RECORD	Output	VARIANT	I、Q、M、D、L	読み出されたユーザーデータ用の宛先領域。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	• 指定したハードウェア識別子のモジュールを設定していない、または • 整合性のあるデータの長さに関する制限を無視したか、または • LADDR パラメータでハードウェア識別子をアドレスとして指定していません。
8092	配列の(Array of) ビット列または整数以外のデータタイプが RECORD パラメータで指定されています。
8093	LADDR で指定されている整合性のあるデータを読み出す DP モジュール/PROFINET IO デバイスがありません。 このエラーコードは、LADDR によってアドレス指定されたモジュールに入力がない場合も発生します。
80A0	I/O にアクセスしたときアクセスエラーが検出されました。
80B1	RECORD パラメータで指定されたターゲット範囲の長さが、設定したユーザーデータの長さよりも短くなっています。
80C0	データがまだ読み出されていません。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

DPWR_DAT: DP スレーブのコンスタントデータの書き込み

説明

「DPWR_DAT」命令は、パラメータ RECORD のデータを基本モジュールまたは DP 標準スレーブ/PROFINET IO デバイスのアドレス指定したモジュールに、そして必要に応じて、プロセスイメージに(すなわち、DP 標準スレーブの関連するアドレス範囲をプロセスイメージの整合性のある範囲として投影している場合)、整合性を保って転送するために使用します。

I/O またはプロセスイメージ出力にアクセスする転送コマンドを使って書き込めるのは最大で連続する 4 バイトであるため、「DPWR_DAT」が必要です。 必要に応じて、プロセスイメージ出力を使って整合性のあるデータを書き込むこともできます。 使用している CPU がこの機能をサポートするかの確認については、関連する文書を参照してください。 整合性のあるデータを書き込む場合、両方の方法を同時に使用しないでください。 「DPWR_DAT」を使用するか、プロセスイメージ出力テーブルを使って書き込みます。 DP 標準スレーブ/PROFINET IO デバイスの整合性のあるデータの追加情報については、セクション「データの一貫性」を参照してください。 DP 標準スレーブがモジュール型設計となっている場合、DP スレーブの 1 つのモジュールのみにアクセスできます。

注意
I/O アクセス 「DPWR_DAT」を使用する場合、OB6x 接続(アイソクロナスモード割り込み)が割り当てられたプロセスイメージパーティションがある I/O 領域へのアクセスは避けてください。

必要に応じて、「DPRD_DAT」命令を 1 倍と以上のデータ領域に使用することも可能です。 データの最大長に関する情報については、お使いの CPU のマニュアルを参照してください(たとえば、S7-1214 では 64 バイト)。

- LADDR パラメータを使用し、DP 標準スレーブ/PROFINET IO デバイスを選択します。 アドレス指定されたモジュールでアクセスエラーが発生すると、エラーコード 8090 が出力されます。
- RECORD パラメータを使用し、書き出されるデータのソース範囲を定義します。
  - ソース範囲は、少なくとも選択したモジュールの出力と同じ長さであることが必要です。 出力のみが転送され、他のバイトは考慮されません。 RECORD パラメータのソース範囲が、設定したモジュールの出力の長さよりも長い場合、出力の最大長さまでのデータのみが転送されます。 設定されたモジュールの出力よりもパラメータ RECORD のソース範囲が短い場合は、エラーコード 80B1 が出力されます。
  - 使用可能なデータタイプ: BYTE、CHAR、WORD、LWORD、DWORD、INT、UINT、USINT、SINT、LINT、ULINT、DINT、UDINT。 タイプ ARRAY または STRUCT のデータ構造体でのこれらのデータタイプの使用も許可されています。
  - データタイプ STRING はサポートされません。

データは同期して転送されます。つまり、命令が完了すると書き込みプロセスが完了します。

パラメータ

次の表に、「DPWR_DAT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IO	I、Q、M、L、または定数	データの書き込み先のモジュールのハードウェア ID。

				このハードウェア ID は、デバイスビューのモジュールのプロパティ、または、システム定数に存在します。
RECORD	Input	VARIANT	I、Q、M、D、L	書き込まれるユーザーデータのソース領域。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	• 指定したハードウェア識別子のモジュールを設定していない、または • 整合性のあるデータの長さに関する制限を無視したか、または • LADDR パラメータでハードウェア識別子を指定していません。
8092	(Array of)ビット列または整数以外のデータタイプが RECORD パラメータで指定されています。
8093	LADDR で指定された HW ID に、整合性のあるデータを書き込める DP モジュール/PROFINET IO デバイスが存在しません。 このエラーコードは、LADDR によってアドレス指定された DP 標準スレーブ/PROFINET IO デバイスに出力がない場合も発生します。
80A1	I/O デバイスへのアクセス中にアクセスエラーが検出されました。
80B1	設定された DP 標準スレーブ/PROFINET IO デバイスの出力よりも長さの短いソース範囲が RECORD パラメータで指定されています。
80C1	前の書き込みジョブのデータが DP 標準スレーブ/PROFINET IO デバイスによって処理されていません。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

iDevice/iSlave

この章には下記に関する情報が記載されています：

- RCVREC: データレコードの受信 (S7-1500)
- PRVREC: データレコードを使用可能にする (S7-1500)

RCVREC: データレコードの受信

説明

I デバイスは、上位コントローラからデータレコードを受信することができます。 受信は、「RCVREC」(receive record)命令を使用してユーザープログラムで行われます。

この命令には、次の動作モードがあります。

- I デバイスにデータレコード受信の要求があるか確認します。
- 出力パラメータでデータレコードを利用できるようにします。
- 上位コントローラに応答を送信します。

入力パラメータ MODE を使用し、命令によって実行される動作モードを特定することができます(以下参照)。

I デバイスが RUN または STARTUP モードである必要があります。

MLEN を使って、受信する最大バイト数を指定します。 ターゲット範囲 RECORD に選択された長さは、MLEN バイト以上の長さとなるようにしてください。

データレコードが受信されると(MODE=1 または MODE=2)、出力パラメータ NEW がデータレコードが RECORD に保存されたことを示します。 RECORD が十分な長さであることに注意してください。 出力パラメータ LEN には、受信したデータレコードの実際の長さがバイトで含まれます。

上位コントローラへの肯定応答のため、CODE1 および CODE2 をゼロに設定します。 受信したデータレコードを拒否する場合は、CODE1 の Error Code 1 および CODE2 の Error Code 2 で否定応答を上位コントローラに入力します。

注記
I デバイスがデータレコード受信の要求を受信した場合、特定の時間内にこの要求を受け取ったことを確認する必要があります。 この確認後、この時間内に上位コントローラに応答を送信する必要があります。 これを遵守しない場合、I デバイスがタイムアウトエラーとなり、I デバイスのオペレーティングシステムが上位コントローラに否定応答を送信する原因となります。 時間の値についての情報は、使用している CPU の仕様を参照してください。

出力パラメータ STATUS は、エラー発生後にエラー情報を受信します。

動作モード

「RCVREC」命令の動作モードは、入力パラメータ MODE で決定できます。 このステップを次の表で説明します。

MODE	意味
0	データレコード受信の要求があるかを確認 I デバイスに上位コントローラからのデータレコードが存在する場合、この命令は出力パラメータ NEW、SLOT、SUBSLOT, INDEX および LEN のみを書き込みます。 MODE=0 で複数回この命令を呼び出した場合、出力パラメータは 1 つの同じ要求のみを参照します。
1	I デバイスのいずれかのサブスロットのデータレコードを受信 I デバイスにいずれかの I デバイスのサブスロットに対する上位コントローラからのデータレコードが存在する場合、この命令は出力パラメータを書き込み、データレコードをパラメータ RECORD に転送します。

2	I デバイスの特定のサブスロットのデータレコードを受信 I デバイスに特定の I デバイスのサブスロットに対する上位コントローラからのデータレコードが存在する場合、この命令は出力パラメータを書き込み、データレコードをパラメータ RECORD に転送します。
3	上位コントローラに肯定応答を送信 この命令は、上位コントローラのデータレコード受信の要求を確認し、既存のデータレコードを受け入れ、上位コントローラに肯定確認を送信します。
4	上位コントローラに否定応答を送信 この命令は、上位コントローラのデータレコード受信の要求を確認し、既存のデータレコードを拒否し、上位コントローラに否定確認を送信します。 拒否の理由を入力パラメータ CODE1 および CODE2 に入力します。

注記
データレコード(NEW=1)の受信後、「RCVREC」を 2 回呼び出して処理を完了する必要があります。 これは、次の順で行う必要があります。

- MODE=1 または MODE=2 による最初の呼び出し
- MODE=3 または MODE=4 による 2 回目の呼び出し

パラメータ

次の表に、「RCVREC」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	INT	I、Q、M、D、L、または定数	モード
F_ID	Input	HW_SUB-MODULE	I、Q、M、D、L、または定数	受信するデータレコード用の I デバイスの転送領域にあるサブスロット (MODE=2 の場合のみ)。 high word は常にゼロに設定されます。
MLEN	Input	INT*	I、Q、M、D、L、または定数	受信するデータレコードの最大長(バイト)
CODE1	Input	BYTE	I、Q、M、D、L、または定数	ゼロ(MODE=3 の場合)および/または Error Code 1 (MODE=4 の場合)
CODE2	Input	BYTE	I、Q、M、D、L、または定数	ゼロ(MODE=3 の場合)および/または Error Code 2 (MODE=4 の場合)
NEW	Output	BOOL	I、Q、M、D、L	• MODE=0:新規データレコードが受信されました。 • MODE=1 または 2: データレコードが RECORD に転送されました。
STATUS	Output	DWORD	I、Q、M、D、L	エラー情報
SLOT	Output	HW_SUB-MODULE	I、Q、M、D、L	F_ID と同一
SUBSLOT	Output	HW_SUB-MODULE	I、Q、M、D、L	F_ID と同一
INDEX	Output	UINT	I、Q、M、D、L	受信したデータレコードの番号です。
LEN	Output	UINT	I、Q、M、D、L	受信したデータレコードの長さです。

RECORD	InOut	VARIANT	I、Q、M、D、L	受け取ったデータレコードのターゲット範囲です。 注記: S7-300 CPU の場合、パラメータ RECORD には、DB パラメータを完全に指定する必要があることに注意してください(たとえば、P#DB13.DBX0.0 Byte 100)。 明示的な DB 番号の省略は、S7-300 CPU では許可されず、ユーザープログラムでエラーメッセージが表示されます。
* STL プログラミング言語では、データタイプ「UINT」を使用します。				

有効なデータタイプの追加情報については、「**有効なデータタイプの概要**」を参照してください。

STATUS パラメータ

STATUS パラメータの解釈については、次のセクションを参照してください。 **パラメータ STATUS**

PRVREC: データレコードを使用可能にする

説明

I デバイスは、データレコードを使用可能にするための上位コントローラからの要求を受信できます。I デバイスは、「PRVREC」(provide record)命令でユーザープログラムでデータレコードが使用できるようにします。

この命令には、次の動作モードがあります。

- I デバイスにデータレコードを使用可能にする要求があるか確認します。
- 要求されたデータレコードを上位コントローラに転送します。
- 上位コントローラに応答を送信します。

入力パラメータ MODE を使用し、命令によって実行される動作モードを特定することができます(以下参照)。

I デバイスが RUN または STARTUP モードである必要があります。

LEN で、送信されるデータレコードの最大バイト数を入力します。 ターゲット範囲 RECORD に選択された長さは、LEN バイト以上の長さとなるようにしてください。

データレコードを使用可能にする要求が存在する場合(MODE=0)、出力パラメータ NEW が TRUE に設定されます。

データレコードを使用可能にする要求が受け入れる場合は、上位コントローラに要求されたデータレコードと RECORD が肯定応答のために書き込まれ、CODE1 と CODE2 にゼロが書き込まれます。データレコードを使用可能にする要求を拒否する場合は、CODE1 の Error Code 1 および CODE2 の Error Code 2 で否定応答を上位コントローラに入力します。

注記
I デバイスがデータレコードを使用可能にする要求を受信した場合、特定の時間内にこの要求を受け取ったことを確認する必要があります。 この確認後、この時間内に上位コントローラに応答を送信する必要があります。 これを遵守しない場合、I デバイスがタイムアウトエラーとなり、I デバイスのオペレーティングシステムが上位コントローラに否定応答を送信する原因となります。 時間の値についての情報は、使用している CPU の仕様を参照してください。

出力パラメータ STATUS は、エラー発生後にエラー情報を受信します。

動作モード

「PRVREC」命令の動作モードは、入力パラメータ MODE で特定できます。 このステップを次の表で説明します。

MODE	意味
0	データレコードを使用可能にする要求があるかを確認 I デバイスに上位コントローラからのデータレコードを使用可能にする要求が存在する場合、この命令は出力パラメータ NEW、SLOT、SUBSLOT, INDEX および RLEN のみを書き込みます。 MODE=0 で複数回この命令を呼び出した場合、出力パラメータは 1 つの同じ要求のみを参照します。
1	I デバイスのいずれかのサブスロットのデータレコードを使用可能にする要求を受信

	I デバイスのいずれかのサブスロットに上位コントローラからのこの要求が存在する場合、この命令は出力パラメータを書き込みます。
2	I デバイスの特定のサブスロットのデータレコードを使用可能にする要求を受信 I デバイスに上位コントローラからのこの要求が存在する場合、この命令は出力パラメータを書き込みます。
3	データレコードを使用可能にし上位コントローラに肯定応答を送信 この命令は、上位コントローラのデータレコードを利用可能にする要求があるかチェックし、要求されたデータレコードを RECORD で利用可能とし、上位コントローラに肯定確認を送信します。
4	上位コントローラに否定応答を送信 この命令は、上位コントローラのデータレコードを利用可能にする要求を確認し、この要求を拒否して、上位コントローラに否定確認を送信します。 拒否の理由を入力パラメータ CODE1 および CODE2 に入力します。

注記
要求(NEW=1)の受信後、命令を 2 回呼び出して処理を完了する必要があります。 これは、次の順で行う必要があります。

- MODE=1 または MODE=2 による最初の呼び出し
- MODE=3 または MODE=4 による 2 回目の呼び出し

パラメータ

次の表に、「PRVREC」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	INT	I、Q、M、D、L、または定数	モード
F_ID	Input	HW_SUB-MODULE	I、Q、M、D、L、または定数	送信するデータレコード用の I デバイスの転送領域にあるサブスロット (MODE=2 の場合のみ)。 high word は常にゼロに設定されます。
CODE1	Input	BYTE	I、Q、M、D、L、または定数	ゼロ(MODE=3 の場合)および/または Error Code 1 (MODE=4 の場合)
CODE2	Input	BYTE	I、Q、M、D、L、または定数	ゼロ(MODE=3 の場合)および/または Error Code 2 (MODE=4 の場合)
LEN	Input	UINT	I、Q、M、D、L、または定数	送信するデータレコードの最大長(バイト)
NEW	Output	BOOL	I、Q、M、D、L	上位コントローラによって新規のデータレコードが要求されました。
STATUS	Output	DWORD	I、Q、M、D、L	エラー情報
SLOT	Output	HW_SUB-MODULE	I、Q、M、D、L	F_ID と同一
SUBSLOT	Output	HW_SUB-MODULE	I、Q、M、D、L	F_ID と同一
INDEX	Output	UINT	I、Q、M、D、L	送信するデータレコードの番号
RLEN	Output	UINT	I、Q、M、D、L	送信するデータレコードの長さ

RECORD	InOut	VARIANT	I、Q、M、D、L	使用可能となったデータレコード 注記: S7-300 CPU の場合、パラメータ RECORD には、DB パラメータを完全に指定する必要があることに注意してください(たとえば、P#DB13.DBX0.0 Byte 100)。 明示的な DB 番号の省略は、S7-300 CPU では許可されず、ユーザープログラムでエラーメッセージが表示されます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

STATUS パラメータの解釈については、次のセクションを参照してください。 パラメータ STATUS

PROFIBUS

この章には下記に関する情報が記載されています：

- DPSYC_FR: DP スレーブの同期/入力値の固定 (S7-1500)
- DPNRM_DG: 診断データの DP スレーブからの読み出し (S7-1200, S7-1500)
- DP_TOPOL: DP マスタシステムのトポロジの特定 (S7-1500)

DPSYC_FR: DP スレーブの同期/入力値の固定

説明

この命令を使用して、DP スレーブの 1 つ以上のグループを同期します。

このファンクションを使用して、関連グループに次の制御コマンドのいずれか、またはそれらを組み合わせて送信します。

- SYNC (同時出力および DP スレーブの出力状態のフリーズ)
- UNSYNC (SYNC 制御コマンドのキャンセル)
- FREEZE (DP スレーブの入力状態のフリーズおよびフリーズされた入力の読み込み)
- UNFREEZE (FREEZE 制御コマンドのキャンセル)

上記の制御コマンドを送信する前に、DP スレーブを設定ごとのグループに割り当てる必要があります。 どの DP スレーブがどのグループに、どの番号で割り当てられているか、また、さまざまなグループの SYNC/FREEZE への応答を把握しておくことが必要です。

注記
制御コマンド SYNC および FREEZE は、ウォーム/コールドリスタートを実行する場合にも有効なままであることに注意してください。

また、1 つの SYNC/UNSYNC ジョブのみ、または 1 つの FREEZE/UNFREEZE ジョブのみを開始できることに注意してください。

ファンクションの説明

"「DPSYC_FR」は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 REQ=1 で「DPSYC_FR」を呼び出してジョブを開始します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

ジョブの識別

SYNC/FREEZE ジョブをトリガし、最初のジョブの完了前に「DPSYC_FR」を再度呼び出すと、新規の呼び出しが同じジョブに対するものかどうかで命令の応答が異なります。 入力パラメータ LADDR、GROUP および MODE が一致した場合、呼び出しは後続の呼び出しとみなされます。

DP モジュールの出力の書き込み

DP モジュールの出力の書き込みは、次のようにトリガされます。

- DP I/O への転送コマンドによる
- 出力のプロセスイメージをモジュールに書き込むことによる(OB 1 の終了時にオペレーティングシステムによって、または「UPDAT_PO」命令の呼び出しによる)
- 「DPWR_DAT」命令の呼び出し。

通常の動作では、DP マスタが出力バイトを DP スレーブの出力に周期的に(PROFIBUS DP バスサイクル内で)転送します。

プロセスへの出力に特定の出力データ(複数のスレーブに分散されている場合も)が完全に同時に適用されるようにする場合、「DPSYC_FR」命令を使用して SYNC 制御コマンドを関連する DP マスタに送信することができます。

SYNC の効果

SYNC 制御コマンドを使用して、選択したグループの DP スレーブを SYNC モードに切り替えます。すなわち、DP マスタは現在の出力データを転送し、関係する DP スレーブにその出力をフリーズするように指示します。 次の出力メッセージフレームを使用して、DP スレーブによって出力データが内部バッファに入力され、出力の状態は変更されません。

それぞれの SYNC 制御コマンドに続き、選択されたグループの DP スレーブがそれらの内部バッファの出力データをプロセスの出力に適用します。

これらの出力は、「DPSYC_FR」命令を使用して UNSYNC 制御コマンドを送信した場合のみ、再度周期的に更新されます。

注記
選択したグループの DP スレーブが現在ネットワークに接続されていないか、制御コマンドが送信されたときにエラーが発生した場合、SYNC モードに切り替わりません。 この情報は、命令の戻り値では通信されません。

DP モジュールの入力データの読み出し

DP モジュールの入力データは、次のように読み出されます。

- DP I/O への読み込みコマンドの使用
- 入力のプロセスイメージが更新されたとき(OB 1 の開始時にオペレーティングシステムによって、または「UPDAT_PI」命令の呼び出しによる)
- 「DPRD_DAT」命令の呼び出しによる。

通常の動作では、DP マスタは DP スレーブからこの入力データを周期的に受信し(PROFIBUS DP バスの周期内で)、それらが CPU で使用できるようにします。

特定の入力データ (複数のスレーブに分散されている場合も)が完全に同時にプロセスから読み出されるようにするには、「DPSYC_FR」命令を使用して関連する DP マスタに FREEZE 制御コマンドを送信します。

FREEZE の効果

FREEZE 制御コマンドを使用して、関係する DP スレーブが FREEZE モードに切り替えられます。すなわち、DP マスタが DP スレーブに入力の現在の状態をフリーズするように命令します。 その後、フリーズされたデータは CPU の入力領域に転送されます。

それぞれの FREEZE 制御コマンドの後、DP スレーブが入力の状態を再度フリーズします。

DP マスタは、DPSYC_FR 命令で UNFREEZE 制御コマンドを送信した後のみに、入力の現在の状態を周期的に受信します。

注記
選択したグループの DP スレーブが現在ネットワークに接続されていないか、制御コマンドが送信されたときにエラーが発生した場合、FREEZE モードに切り替わりません。 この情報は、命令の戻り値では通信されません。

データの整合性

DPSYC_FR ファンクションは非同期で、より高い優先度クラスで割り込むことができるため、「DPSYC_FR」命令を使用する場合にプロセスイメージが I/O の実際の入力および出力と一致していることを確認する必要があります。

これは、次の整合性の規則に従った場合は保証されます。

- 「SYNC 出力」および「FREEZE 入力」に適切なプロセスイメージパーティションを定義します (S7-400 のみで可能)。 SYNC ジョブのそれぞれの最初の呼び出しの直前に、「UPDAT_PO」命令を呼び出します。 FREEZE ジョブの最後の呼び出しの直後に、「UPDAT_PI」命令を呼び出します。

- 別の方法: SYNC ジョブに関係する出力および FREEZE ジョブに関係する入力にはダイレクト I/O のみを使用します。 SYNC ジョブが有効な場合はこれらの出力に書き込まないでください。また、FREEZE ジョブが有効な場合は、これらの入力を読み出さないでください。

DPWR_DAT および DPRD_DAT の使用

「DPWR_DAT」命令を使用する場合は、関係する出力に SYNC ジョブを送信する前に完了している必要があります。

「DPRD_DAT」命令を使用する場合は、関係する入力に FREEZE ジョブを送信する前に完了している必要があります。

スタートアップおよび「DPSYC_FR」

必ずユーザーがスタートアップ OB の SYNC および FREEZE 制御コマンドを送信する必要があります。

ユーザープログラムが起動したときに 1 つ以上のグループの出力を SYNC モードにする場合、スタートアップ中にこれらの出力を初期化し、SYNC 制御コマンドで「DPSYC_FR」命令を完全に実行する必要があります。

ユーザープログラムが起動したときに 1 つ以上のグループの入力を FREEZE モードにする場合、起動中にこれらの入力に対して FREEZE 制御コマンドで「DPSYC_FR」命令を完全に実行する必要があります。

パラメータ

以下の表に、「DPSYC_FR」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	レベルトリガの制御パラメータ REQ=1: SYNC/FREEZE ジョブのトリガ
LADDR	Input	HW_DPMASTER	I、Q、M、D、L、または定数	DP マスタインターフェースのハードウェア識別子 この数は、ネットワークビューの DP マスタインターフェースのプロパティから、または標準タグテーブルの[システム定数]タブから取得することができます。
GROUP	Input	BYTE	I、Q、M、D、L、または定数	グループ選択 ビット 0 = 1: グループ 1 が選択済み ビット 1 = 1: グループ 2 が選択済み : ビット 7 = 1: グループ 8 が選択済み ひとつのジョブで複数のグループを選択できます。 値 B#16#0 が無効です。
MODE	Input	BYTE	I、Q、M、D、L、または定数	ジョブ ID (コード化は EN 50 170 Volume 2, PROFIBUS に準拠) ビット 0: 予約済み(値 0) ビット 1: 予約済み(値 0) ビット 2:

				• = 1: UNFREEZE が実行されます。 • = 0: 意味なし ビット 3: • = 1: FREEZE が実行されます。 • = 0: 意味なし ビット 4: • = 1: UNSYNC が実行されます。 • = 0: 意味なし ビット 5: • = 1: SYNC が実行されます。 • = 0: 意味なし ビット 6: 予約済み(値 0) ビット 7: 予約済み(値 0) 次の値が返されます。 • 1 つのジョブあたり 1 つの ID: ○ B#16#04 (UNFREEZE) ○ B#16#08 (FREEZE) ○ B#16#10 (UNSYNC) ○ B#16#20 (SYNC) • 1 つのジョブあたり複数の ID: ○ B#16#14 (UNSYNC, UN-FREEZE) ○ B#16#18 (UNSYNC, FREEZE) ○ B#16#24 (SYNC, UNFREEZE) ○ B#16#28 (SYNC, FREEZE)
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。 ブロックが実行されるたびに、必ず RET_VAL を評価するようにします。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY=1: SYNC/FREEZE ジョブが未完了です。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

注記
DPV1 スレーブにアクセスする場合、これらのスレーブのエラー情報は DP マスタから命令に送ることができます。 このエラー情報については、STATUS[3]、「STATUS」パラメータを参照してください。

エラーコード* (W#16#...)	説明
0000	ジョブがエラーなしで完了済み。
7000	REQ=0 を使った最初の呼び出し。LADDR、GROUP および MODE で指定されたジョブが有効ではありません。BUSY の値は 0 です。
7001	REQ=1 を使った最初の呼び出し。LADDR、GROUP および MODE で指定されたジョブがトリガされました。BUSY の値は 1 です。
7002	中間呼び出し(REQ は対象外)。 有効になった SYNC/FREEZE ジョブがまだ実行中です。BUSY の値は 1 です。
8090	LADDR で選択されたモジュールは DP マスタではありません。
8093	この命令は LADDR で選択されたモジュール(DP マスタの設定またはバージョン)には許可されていません。
8094	GROUP パラメータが不正です。
8095	MODE パラメータが不正です。
80A4	PROFIBUS の通信が中断されました。
80B0	GROUP で選択されたグループは設定されていません。
80B1	GROUP で選択されたグループは、この CPU に割り当てられていません。
80B2	MODE で指定された SYNC ジョブは、GROUP で選択されたグループには許可されていません。
80B3	MODE で指定された FREEZE ジョブは、GROUP で選択されたグループには許可されていません。
80C2	DP マスタ:でのリソースの一時的な不足: DP マスタは、現在 CPU で最大数のジョブの処理中です。
80C3	1 回にトリガできるのは 1 つの SYNC/UNSYNC ジョブのみであるため、現在この SYNC/UNSYNC ジョブは、有効にできません。 ユーザープログラムを確認してください。
80C4	1 回にトリガできるのは 1 つの FREEZE/UNFREEZE ジョブのみであるため、現在この FREEZE/UNFREEZE ジョブは、有効にできません。 ユーザープログラムを確認してください。
80C5	DP インターフェースでの直接的な短絡
80C6	CPU による I/O 切断によってジョブが中止されました。
80C7	DP マスタのウォームまたはコールドリスタートのため、ジョブが中止されました。
一般エラー情報	関連項目:GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DPNRM_DG: 診断データの DP スレーブからの読み出し

説明

「DPNRM_DG」命令を使用し、DP スレーブの現在の診断データを読み出します。この診断データは、EN 50 170 Volume 2、PROFIBUS で指定された形式です。

次の表は、スレーブ診断データの基本構造を示します。詳細については、DP スレーブのマニュアルを参照してください。

Byte (バイト)	意味
0	ステーションステータス 1
1	ステーションステータス 2
2	ステーションステータス 3
3	マスタステーション番号
4	ベンダー ID (上位バイト)
5	ベンダー ID (下位バイト)
6 ...	追加のスレーブ固有診断情報

読み出されたデータは、転送でエラーが発生しなかった場合には、RECORD で示されたコピー先の領域に入力されます。 読み出しプロセスを開始するには、「DPNRM_DG」命令が呼び出されたときに REQ 入力パラメータに値「1」を割り当てます。

ファンクションの説明

読み出しプロセスは非同期で実行されます。つまり、複数の呼び出しにわたって実行されます。 出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

パラメータ

以下の表に、「DPNRM_DG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: 読み出し要求
LADDR	Input	HW_DPSLAVE	I、Q、M、D、L、または定数	DP スレーブの設定した診断アドレス。 注記: アドレスは 16 進形式で指定する必要があります。たとえば診断アドレス 1022 は、以下を意味します。 LADDR:=W#16#3FE.
RET_VAL	Return	DINT, INT, LREAL, REAL	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。 エラーが発生しなかった場合、実際に転送

				されるデータの長さが RET_VAL に入力されます。
RECORD	Output	VARIANT	I、Q、M、D、L	読み出された診断データの宛先領域。BYTE データタイプのみが許可されます。 読み出されるデータレコード最小長さ、またはコピー先の領域は 6 です。送信するデータレコードの最大長は 240 です。標準スレーブは、最大 244 バイトまでの 240 バイト以上の診断データを提供することができます。この場合、最初の 240 バイトはコピー先の領域に転送され、データにオーバーフロービットが設定されます。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: 読み出しプロセスが未完了です。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

「データタイプ変換の概要」には、他のプログラミング言語でのデータタイプ変換に関する情報が記載されています。

RECORD パラメータ

CPU が読み出された診断データの実際の長さを評価します。

RECORD に指定された長さが

- 提供されたデータのバイト数未満の場合、このデータは破棄され、RET_VAL に対応するエラーコードが入力されます。
- 提供されたデータのバイト数よりも大きいか、等しい場合、このデータは宛先領域に受け入れられ、実際の長さが RET_VAL に正の値で入力されます。

> **注記**
> RECORD の実際のパラメータが、必ずジョブに属するすべての呼び出しに対応するようにしてください。
>
> ジョブは、LADDR 入力パラメータによって、一意で識別されます。

240 バイトを超える診断データを持つ標準スレーブ

標準診断データ数が 241～244 バイトの標準スレーブの場合、次の点に注意してください。

RECORD に指定された長さが

- 240 バイト未満の場合、データは破棄され、対応するエラーコードが RET_VAL に入力されます。
- RECORD に指定された長さが 240 バイト以上である場合、標準診断データの最初の 240 バイトはコピー先の領域に転送され、データにオーバーフロービットが設定されます。

RET_VAL パラメータ

- ファンクションの実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
- データ転送中にエラーがなかった場合、RET_VAL は読み出されたデータの長さを正の値のバイトで含みます。

> **注記**
> DP スレーブで読み出すデータ量は、その診断ステータスによって異なります。

RET_VAL パラメータのエラー情報の評価については、次の表を参照してください。

エラーコード (W#16#...)	説明	制限事項
7000	REQ=0 による最初の呼び出し: 有効なデータ転送がありません。BUSY の値は「0」です。	-
7001	REQ=1 による最初の呼び出し: データ転送がトリガされました。BUSY の値は 1 です。	リモート I/O
7002	中間の呼び出し(REQ は無関係): データ転送が既に有効です。BUSY の値は「1」です。	リモート I/O
8090	LADDR パラメータで指定されたアドレスが無効です。	-
8093	この命令は、LADDR および IOID を使用して選択したモジュールでは許可されません。	-
80A2	• レイヤー 2 での DP プロトコルエラー(たとえば、スレーブ故障またはバスの問題) • ET 200S の場合、DPV0 モードでデータレコードが読み出せません。	リモート I/O
80A3	ユーザーインターフェース/ユーザーによる DP プロトコルエラー	リモート I/O
80A4	PROFIBUS の通信が中断されました。	リモート I/O
80B0	• モジュールタイプで命令が使用できません。 • モジュールがデータレコードを認識しません。 • データレコード番号 241 は許可されていません。 • 「WR_REC」の場合、データレコード 0 および 1 は許可されていません。	-
80B1	RECORD パラメータで指定された長さが不正です。	指定された長さ < データレコード長
80B2	設定されたスロットが 割り当てられていません。	-
80B3	実際のモジュールタイプが指定されたモジュールタイプと一致していません。	-
80C0	使用可能な診断データがありません。	-
80C1	モジュールで、同じデータレコードの前の書き込みジョブのデータがモジュールによって処理されていません。	-
80C2	モジュールは、現在 CPU で可能な最大数のジョブを処理しています。	-
80C3	必要なリソース(メモリなど)が現在占有されています。	-
80C4	内部の一時的なエラー。 ジョブを実行できませんでした。 ジョブを繰り返します。 このエラーが頻発する場合は、据付けを点検して電気的な干渉の発生源を探します。	-
80C5	リモート I/O が使用不可です。	リモート I/O
80C6	優先度クラス中止(再起動またはバックグラウンド)のため、データレコードの転送が停止されました。	リモート I/O
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得	-

* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

DP_TOPOL: DP マスタシステムのトポロジの特定

説明

この命令を使用して、選択した DP マスタシステムのトポロジ決定をトリガします。 この命令を呼び出すと、DP マスタシステムのすべての診断リピータをアドレス指定します。

注記
トポロジは、1 回に 1 つの DP マスタシステムのみで決定することができます。

トポロジ決定は、エラーが発生した場合にエラーの詳細な場所を表示するための前提条件です。 設定を行った後、および DP マスタシステムの物理的な設定の変更を行った後は毎回、「DP_TOPOL」によってトポロジ決定を繰り返す必要があります。

物理的設定の変更とは、

- ラインの長さの変更
- ステーションまたはリピータ機能を持つコンポーネントの追加または撤去
- ステーションアドレスの変更

診断リピータがエラーを報告した場合、1 回の「DP_TOPOL」パスの間「DP_TOPOL」が DPR および DPRI 出力を書き込みます。 選択した DP マスタシステムの複数の診断リピータによってエラーが報告された場合、「DP_TOPOL」が DPR および DPRI と、エラーを報告した最初の診断リピータについての情報を書き込みます。 診断情報全体をプログラミングデバイスまたは「DPNRM_DG」命令を使用して読み出すことができます。 どの診断リピータもエラーを報告しない場合、DPR および DPRI 出力の値は NULL になります。

エラー発生後にトポロジ決定を繰り返す場合、最初に「DP_TOPOL」をリセットする必要があります。 これは、REQ=0 および R=1 を使用して「DP_TOPOL」を呼び出して行います。

ファンクションの説明

"「DP_TOPOL」は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 REQ=1 で DP_TOPOL を呼び出して、バストポロジの決定を開始します。 プロセスをキャンセルする場合は、R=1 で「DP_TOPOL」を呼び出します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

注記
トポロジの決定には数分間かかる場合があります。

ジョブの識別

入力パラメータ DP_ID は一意でジョブを指定します。

「DP_TOPOL」を呼び出して、トポロジが再決定される前にこの命令を再度呼び出すと、この命令の応答方法は、新規の呼び出しが同じジョブを含むか否かで異なります。 DP_ID パラメータが完了していないジョブと一致すると、この呼び出しは後続の呼び出しとしみなされ、値 W#16#7002 が RET_VAL に入力されます。 一方、別のジョブが含まれる場合、CPU はこれを拒否します。

パラメータ

以下の表に、「DP_TOPOL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ=1:トポロジ決定をトリガします
R	Input	BOOL	I、Q、M、D、L、または定数	R=1:トポロジ決定をキャンセルします
DP_ID	Input	HW_IO-SYSTEM	I、Q、M、D、L、または定数	トポロジが決定されるマスタシステムの DP マスタシステム ID
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY=1:トポロジ決定がまだ完了していません。
DPR	Output	BYTE	I、Q、M、D、L	エラーを報告した診断リピータの PROFIBUS アドレス。
DPRI	Output	BYTE	I、Q、M、D、L	エラーを報告したセグメント診断リピータの測定: • ビット 0 = 1: セグメント DP2 の一時障害 • ビット 1 = 1: セグメント DP2 の永続的な障害 • ビット 4 = 1: セグメント DP3 の一時障害 • ビット 5 = 1: セグメント DP3 の永続的な障害

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

次の表で「本当の」エラー情報(エラーコード W#16#8xyz)を参照する場合、2 つの場合を分けて考えます。

- 一時的なエラー(エラーコード W#16#80A2～80A4、80C3、80C5):

  このタイプのエラーは、ユーザーの操作なしで解決する可能性があります。すなわち、再度「DP_TOPOL」を呼び出すことを推奨します(必要に応じて、複数回の呼び出し)。

  一時的なエラーの例: 必要なリソースが現在使用中である場合(W#16#80C3)。
- 永続的なエラー(エラーコード W#16#8082、80B0、80B2):

  このタイプのエラーは、自然に解決しません。「DP_TOPOL」の新規の呼び出しは、エラーを解消してからのみ実行するよう推奨します。 永続的なエラーの例: DP マスタ/CPU がこのサービスをサポートしていません。 (W#16#80B0).

エラーコード* (W#16#...)	説明
0000	ジョブがエラーなしで完了済み。
7000	REQ=0 による最初の呼び出し: トポロジ決定がトリガされていません。 BUSY の値は「0」です。

7001	REQ=1 による最初の呼び出し: トポロジ決定実行の要求が送信されました。 BUSY の値は「1」です。
7002	中間呼び出し(REQ は対象外): トポロジ決定がまだ完了していません。BUSY の値は「1」です。
7010	トポロジ識別をキャンセルしようと試みました。 しかし、DP_ID で指定された実行中のジョブはありません。 BUSY の値は「0」です。
7011	R=1 による最初の呼び出し: トポロジ決定のキャンセルがトリガされました。BUSY の値は「1」です。
7012	中間呼び出し トポロジ決定のキャンセルがまだ完了していません。 BUSY の値は「1」です。
7013	最終の呼び出し: トポロジ決定がキャンセルされました。 BUSY の値は「0」です。
8082	指定された DP_ID で設定された DP マスタはありません。
80A2	トポロジ決定中にエラーが発生しました。詳細情報については、出力パラメータ DPR および DPRI を参照してください。
80A3	トポロジ決定中にエラーが発生しました。 モニタリングタイムが経過しました(タイムアウト)。
80A4	PROFIBUS の通信が中断されました。
80B0	DP マスタ/CPU がこのサービスをサポートしていません。
80B2	トポロジ決定中にエラーが発生しました。 選択した DP マスタシステムで、診断リピータが見つかりませんでした。
80C3	必要なリソースが現在使用中である場合。 考えられる原因: 2 番目のトポロジ決定を開始しました(一度に 1 回のトポロジ決定のみが許可されます)。
80C5	DP マスタシステムは、現在使用できません。
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

ASI

この章には下記に関する情報が記載されています：

- ASI_CTRL (S7-1200, S7-1500)

ASI_CTRL

この章には下記に関する情報が記載されています：

- ASI_CTRL の説明 (S7-1500)
- ASi コマンド (S7-1500)

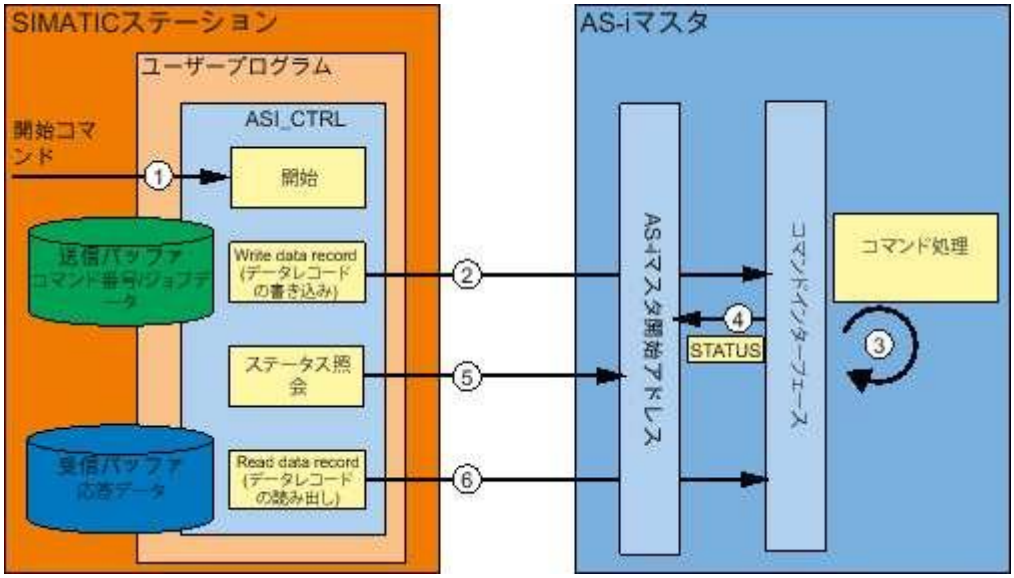
ASI_CTRL の説明

説明

「ASI_CTRL」命令を使用すると、PLC のユーザープログラムから AS-i マスタの動作を制御できます。 この命令は、コマンドプロトコルを自動的に処理します。 また、SIMATIC AS-i マスタでのパラメータ割り当て、および情報データの読み出しも可能です。 コマンドインターフェースの機能および操作については、AS-i マスタのマニュアルで解説されています。

中央に挿入される AS-i マスタ、および PROFIBUS DP を使用したリモートの AS-i マスタがサポートされています。 PROFINET IO (たとえば IE/PB Link PN IO)と組み合わせることも可能です。

以下の概要図に、「ASI_CTRL」命令の機能を示します。



①	REQ パラメータでの処理の開始。
②	プログラムは必要なコマンドを「RDREC」命令を使用して AS-i マスタに送信します。
③	AS-i マスタはそのコマンドを処理します。
④	AS-i マスタの現在のステータスは、バイナリデータの入力領域(論理開始アドレス)に格納されます。
⑤	「ASI_CTRL」命令は 4 つのステータスビットを周期的に照会し、評価します。
⑥	コマンドの処理が完了すると、「RDREC」によってコマンドジョブが完了します。 コマンドによっては、「RDREC」のデータフィールドにコマンドの応答データや追加のステータス情報が含まれる場合があります。

IE/ AS-i リンクと DP/AS-i リンクのコマンド呼び出しの相違点

コントローラが AS-i マスタとコマンドを交換する方法は、IE/AS-i リンクと DP/AS-i リンクでは大きく異なります。

- **IE/AS-i リンク**(PROFINET)では、データレコードインターフェースを使用していました。 データレコードの書き込み」(「WRREC」命令)、または「データレコードの読み出し」(「RDREC」命令)では、異なるコマンドがさまざまなデータレコード番号で呼び出されていました。
- **DP/AS-i リンク**(PROFIBUS)はコマンドインターフェースを使用します。 データレコードの書き込み」(「WRREC」命令)、または「データレコードの読み出し」(「RDREC」命令)では、すべてのコマンドがデータレコード番号 2 で呼び出されます。 コマンドのタイプは、書き込みジョブのデータコ内容によって決まります。

「ASi_3422」命令と比較した場合の変更点。

「ASI_CTRL」命令は、「ASi_3422」命令(S7-300/400 CPU の場合のみ)の修正バージョンであり、機能性と互換性が向上しています。 主な変更点は、以下の通りです。

- 書き込み、および読み出し診断データレコードについては、「WR_REC」、および「RD_REC」命令が、「RDREC」、および「WRREC」命令に変更されました。 機能は同じですが、後者は PROFINET IO 経由でのデータ転送をサポートしています。
- 命令のブロックタイプが、ファンクション(FC)からファンクションブロック(FB)に変更されました。 "「ASI_CTRL」にはインスタンスデータブロックが設けられており、マルチインスタンスに対応しています。
- 「ASI_CTRL」の仮パラメータの名称は、SIMATIC システムブロックに準拠します。 STARTUP 入力パラメータは存在しません。 STATUS パラメータの定義は、「RDREC」、および「WRREC」命令に基づきます。 パラメータ DONE、および新しいパラメータ BUSY のステータス識別子も調整されています。

「ASI_CTRL」命令の操作

「ASI_CTRL」命令は、非同期ファンクションブロックです。つまり、その処理は複数の呼び出しにわたって実行されます。

- REQ = TRUE の場合、ジョブが開始します。
- ジョブステータスは、出力パラメータ BUSY と出力パラメータ STATUS の 2 つの中央のバイトで表示されます。
- BUSY パラメータは、ジョブ処理中にセットされます。 最初の呼び出しで、STATUS に値 00700100_H が割り当てられます。 その後、このジョブのすべての呼び出しで、値 00700200_H が割り当てられます。 ジョブが完了すると、パラメータ DONE または ERROR に結果が出力されます。
  - エラーが発生しなかった場合、DONE がセットされます。 AS-i マスタからの応答データを処理するジョブの場合、このデータは指定された受信バッファ内に提供されます。 この場合、STATUS パラメータは 提供されたデータ量をバイトで表示します。 応答データのないジョブの場合、値 00000000_H は STATUS に格納されます。
  - ジョブ処理中にエラーが発生すると、ERROR がセットされます。 このような場合、受信バッファの内容は無効になります。 発生したエラーの詳細説明を照会するためのエラーコードが STATUS パラメータに入力されます。

コマンド呼び出しの回数

「ASI_CTRL」命令を使用してコマンドを送信する場合、「RDREC」または「WRREC」を使用して同一の AS-i マスタ他のコマンドを同時に送信することはできません。 これは、同一の AS-i マスタに対する命令の複数呼び出しにも当てはまります。

「ASI_CTRL」命令は、実行中に割り込みを入れることはできません。 したがって、呼び出しは相互に割り込むプログラム優先度クラスにはプログラミングできません(たとえば、OB 1 および OB 35 の呼び出し)。

パラメータ

以下の表に、「ASI_CTRL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、定数	REQ = TRUE の場合、ジョブが既に実行中でなければ新しいジョブが開始します。 エッジ評価は発生しません。
LADDR	Input	WORD	I、Q、M、D、L、定数	AS-i マスタのハードウェア識別子。 モジュールのプロパティからアドレスを取得できます。
SD	Input	VARIANT	I、Q、M、D、L	送信バッファ このパラメータは、コマンドが指定されるメモリ領域を参照します(「ASi コマンド」を参照)。
RD	Input	VARIANT	I、Q、M、D、L	受信バッファ このバッファは、応答データを送信するコマンドのみに関連します。 このパラメータは、コマンド応答が格納されたメモリ領域を参照します(「ASi コマンド」を参照)。
DONE	Output	BOOL	I、Q、M、D、L	DONE = TRUE: ジョブがエラーなしで完了しました。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = TRUE: ジョブが進行中です。
ERROR	Output	BOOL	I、Q、M、D、L	ERROR = TRUE: ジョブがエラーによって中止されました。
STATUS	Output	DWORD	I、Q、M、D、L	ジョブステータス/エラーコード 「STATUS パラメータ」の説明を参照してください。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記

パラメータ LADDR、SD および RD

パラメータ LADDR、SD、および RD は、ジョブの進行中はいずれブロックサイクル内でも変更してはなりません。 これらのパラメータは一定に保つ必要があります。

STATUS パラメータ

次の表に、DONE および ERROR の場合に表示される可能性がある STATUS を示します。

DONE	ERROR	STATUS	意味
0	0	00700000$_H$	最初の呼び出しで REQ = FALSE:有効なジョブなし。
0	0	00700100$_H$	最初の呼び出しで REQ = TRUE:ジョブ開始済み。

0	0	00700200$_H$	以降の呼び出し(REQ とは無関係):ジョブ進行中。
1	0	00000000$_H$	ジョブがエラーなしで完了しました。 応答データなし。
1	0	0000xx00$_H$	ジョブがエラーなしで完了しました。 xx バイトの応答データ。
0	1	C0818400$_H$	仮オペランド RD のデータタイプが無効です。
0	1	C0818500$_H$	AS-i マスタとの通信エラー(LADDR パラメータで不正なアドレスが設定されています)。
0	1	C0838100$_H$	不正な AS-i スレーブアドレス。
0	1	C0838200$_H$	AS-i スレーブが無効(LAS に存在しません)。
0	1	C0838300$_H$	AS インターフェースでのエラー(SD パラメータの値が小さすぎる可能性があります)。
0	1	C0838400$_H$	現在の AS-i マスタのステータスではこのコマンドは無効です。
0	1	C0838500$_H$	アドレス「0」の AS-i スレーブは存在しません。
0	1	C0838600$_H$	AS-i スレーブに無効な設定データがあります(I/O または ID コード)。
0	1	C083A100$_H$	アドレス指定された AS-i スレーブが AS インターフェースで見つかりません。
0	1	C083A200$_H$	アドレス「0」の AS-i スレーブは存在しません。
0	1	C083A300$_H$	AS インターフェースに新規アドレスの AS-i スレーブが既に存在しています。
0	1	C083A400$_H$	AS-i スレーブアドレスを削除できません。
0	1	C083A500$_H$	AS-i スレーブアドレスを設定できません。
0	1	C083A600$_H$	AS-i スレーブアドレスを保存できません。
0	1	C083A700$_H$	拡張 ID1 コード読み出し中のエラー。
0	1	C083A800$_H$	ターゲット範囲が不正です(たとえば、B スレーブアドレスが標準スレーブで使用されています)。
0	1	C083B100$_H$	文字列転送中に長さのエラーが発生しました。
0	1	C083B200$_H$	文字列転送中にプロトコルエラーが発生しました。
0	1	C083F800$_H$	不明なジョブ番号またはジョブパラメータです。

0	1	C083F900$_H$	AS-i マスタが EEPROM エラーを検出しました。

ASi コマンド

説明

コマンドインターフェースにより、コントローラと AS-i マスタはパラメータ割り当てや情報データを交換することが可能です。

これらのコマンドは、

- AS-i マスタ仕様で定められた M4 マスタプロファイルのすべての機能を提供します。
- コントローラからの AS-i マスタのすべての設定を可能にします。

> **注記**
>
> **サポートされている AS-i コマンド**
>
> サポートされている AS-i および詳細説明については、関連する AS-i マスタのマニュアルを参照してください。

送信バッファの全般的な構造

次の表に、コマンドおよびジョブの送信バッファの全般的な構造を示します。 コマンド番号の領域は、必ず埋められている必要があります。 ジョブデータのバイト数は、コマンドの説明に記載されています(AS-i マスタのマニュアルを参照してください)。 「q」が送信バッファの開始アドレスです。

バイト	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0
	意味							
q + 0	コマンド番号							
q + 1	ジョブデータ							
q + 2	ジョブデータ							
q + ...	ジョブデータ							

受信バッファの全般的な構造

次の表に、コマンド応答データの受信バッファの全般的な構造を示します。 応答データのバイト数はコマンドによって異なります。 応答データを返さないコマンドもあります。このようなコマンドには、データで埋められない仮の受信バッファを用意します。 「n」が受信バッファの開始アドレスです。

バイト	ビット 7	ビット 6	ビット 5	ビット 4	ビット 3	ビット 2	ビット 1	ビット 0
	意味							
n + 0	コマンド番号(エコー)							
n + 1	応答データ							
n + 2	応答データ							
n + ...	応答データ							

注意

メモリ領域は上書きされる場合があります

「ASI_CTRL」命令の受信バッファが短すぎる場合、隣接するメモリ領域が上書きされる場合があります。「ASI_CTRL」命令の呼び出し時に、RD パラメータの ANY ポインタで指定される長さは関係しません。 受信バッファで必要な長さは、コマンドの説明に記載されています。

コマンド番号 39_H、41_H、42_H、43_H、44_H の場合:

コマンドが返すデータが小さい場合でも、受信バッファには 221 バイトの長さ(バイト 0～220)が必要です。 コマンドによっては、AS-i マスタによって受信バッファの最上位バイトが値ゼロで上書きされる場合があります。

AS-i コマンド

使用可能な AS-i コマンドを以下の表に示します。

名前	パラメータ	戻り値	コード化
Set_permanent_parameter (Set_Permanent_Parameter)	スレーブアドレス、パラメータ		00_H
Get_permanent_parameter (Get_Permanent_Parameter)	[スレーブアドレス]	パラメータ	01_H
Write_parameter (Write_Parameter)	スレーブアドレス、パラメータ	パラメータエコー	02_H
Read_parameter (Read_Parameter)	[スレーブアドレス]	パラメータ値	03_H
Store_actual_parameters (Store_Actual_Parameters)			04_H
Set_configuration_data	スレーブアドレス、設定		25_H
Get_configuration_data	[スレーブアドレス]	設定データの設定	26_H
Store_actual_configuration (Store_Actual_Configuration)			07_H
Get_actual_configuration	[スレーブアドレス]	実際の設定データ	28_H
Configure_LPS	LPS		29_H
Set_offline_mode	モード		$0A_H$
Select_auto-program	モード		$0B_H$
Set_mode	モード		$0C_H$
Change_AS–iSlave_address (Change_AS–iSlave_Address)	アドレス 1、アドレス 2		$0D_H$
Get_AS-iSlave_status	[スレーブアドレス]	AS-i スレーブのエラーレコード	$0F_H$
Read_lists_and_flags		LDS、LAS、LPS、フラグ	30_H
Get_overall_configuration		実際の設定データ、現在のパラメータ、LAS、フラグ	39_H

Set_overall_configuration	全体的な設定		$3A_H$
Write_parameter_list	パラメータリスト		$3C_H$
Read_parameter_echo_list		パラメータエコーリスト	33_H
Write_CTT2_request	スレーブアドレス CTT2 文字列	CTT2 文字列	44_H
Read_version_identifier		バージョン文字列	14_H
Read_AS-i_slave	[スレーブアドレス]	ID コード	17_H
Read_AS-i_slave_extended_ID1	[スレーブアドレス]	拡張 ID1 コード	37_H
Write_AS-iSlave_extended_ID1	拡張 ID1 コード		$3F_H$
Read_AS-iSlave_extended_ID2	[スレーブアドレス]	拡張 ID2 コード	38_H
Read_AS-iSlave_IO	[スレーブアドレス]	I/O 構成	18_H
Read_I/O_error_list		LPF	$3E_H$
Write_AS–i-slave_parameter_string	スレーブアドレス、パラメータ文字列		40_H
Read_AS–iSlave_parameter_string	[スレーブアドレス]	パラメータ文字列	41_H
Read_AS-iSlave_ID_string	[スレーブアドレス]	ID 文字列	42_H
Read_AS-iSlave_diagnostic_string	[スレーブアドレス]	診断文字列	43_H
Read_AS-i_line_error_counter			$4A_H$
Read_and_clear_AS-i_line_error_counter			$4B_H$
Read_AS-iSlave_error_counter	[スレーブアドレス]		$4C_H$
Read_and_clear_AS-iSlave_error_counter	[スレーブアドレス]		$4D_H$
DP/AS-i F-Link 用の追加コマンド:			
AS–i_status/diag_of_F_slaves		すべての AS-i SAFE スレーブのステータス/診断	51_H

注記

AS-i マスタコマンドインターフェースの再初期化

コマンド 77_H は上記の表には記載されていません。 この呼び出しは、AS-i マスタのコマンドインターフェースを再初期化します。 AS-i マスタが指定した現在処理中のいずれのコマンドも終了します。

DP/AS-i リンク Advanced の V2.1.20 以降では、コマンド $0E_H$ も使用可能です。 この呼び出しにより、配線の地絡モニタリング機能の解放およびブロックが可能です。

PROFIenergy

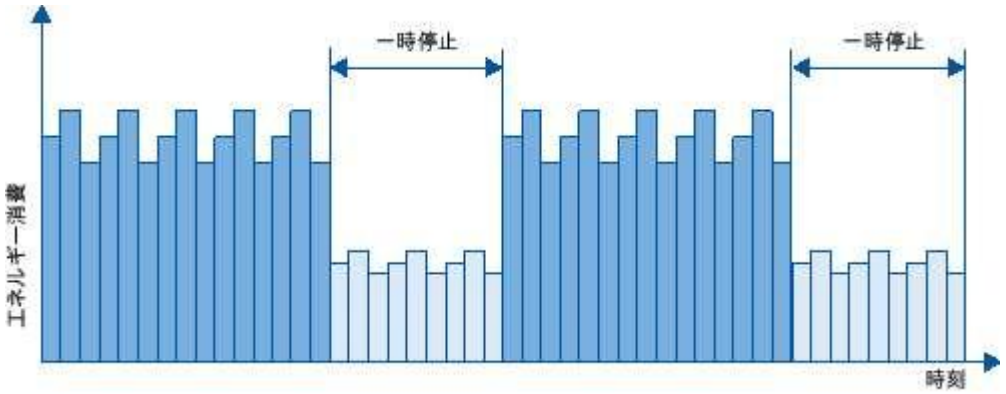
この章には下記に関する情報が記載されています：

- PROFIenergy の説明 (S7-1500)
- IO コントローラ (S7-1500)
- iDevice/iSlave (S7-1500)

PROFIenergy の説明

PROFIenergy

PROFIenergy はメーカーやデバイスに依存しない、PROFINET での電力管理用プロファイルです。PROFIenergy を使用することで、生産の停止中や予期せぬ中断時に電力消費を低減するためのデバイスのシャットダウンを集中的に管理することが可能になります。



PROFINET デバイス/電源モジュールは、PROFINET IO コントローラのユーザープログラムの特別なコマンドによってスイッチが切られます。追加のハードウェアは不要です。 PROFINET デバイスは、PROFIenergy コマンドを直接解釈します。

PROFIenergy コントローラ(PE コントローラ)

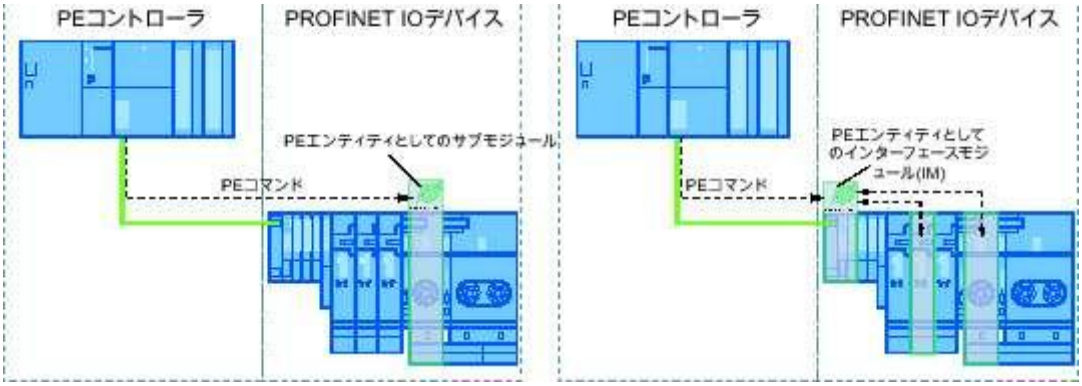
PE コントローラは、下位デバイスのアイドル状態を有効化/無効化する PLC です。 個々の製造コンポーネント、および製造ライン全体のスイッチオフと再始動をユーザープログラムから行います。 コマンド(たとえば「Start_Pause」や「End_Pause」)は、対応する命令(ファンクションブロック)を使用して下位のデバイスに送信されます。 これらのコマンドは、PROFINET 通信プロトコル経由で送信されます。

PROFIenergy エンティティ(PE エンティティ)

PE エンティティは、PE コントローラの PROFIenergy コマンドを受信し、これらのコマンドを適宜実行します(たとえば、測定値を返す、省電力モードを有効化するなど)。 PROFIenergy 対応デバイスでの PE エンティティの実装は、デバイスおよびメーカーに依存します。

たとえば、PE エンティティは以下で実行可能です。

- サブモジュールのプロキシ内: PE コマンドは、アドレス指定されたサブモジュール、および既存の下位サブモジュールに対して有効です。
- モジュールのプロキシ内: PE コマンドは、モジュール内の異なるサブモジュールに対して有効です。



- プロキシ機能を持たないネットワーク接続されたサブモジュール: この場合、PE コマンドはここのサブモジュールに対してのみ有効です。

PROFIenergy 命令

- IO コントローラ用の命令
  - 「PE_START_END」命令は、PROFINET デバイスのアイドル状態を有効化または無効化を行う最も簡単な方法です(PROFIenergy コマンド「Start_Pause」および「End_Pause」)。 これには、命令内で信号立ち上がりエッジ、および信号立ち下りエッジを使用します。
  - 「PE_CMD」を使用すると、「Start_Pause」および「End_Pause」を含むすべての PROFIenergy コマンドを送信することが可能です。 他のコマンドと組み合わせることで、たとえば PROFINET デバイスの現在のステータスや動作を照会することができます。
  - 「PE_DS3_Write_ET200S」は、ET 200S の最大 8 つのスロットの切り替え特性を定義するために使用します。 この命令は PROFIenergy 命令ではありませんが、ET 200S 向けの PROFIenergy の機能を補完します。
- iDevice 用の命令

  「PE_I_DEV」命令を使用すると、PROFIenergy を iDevice にも実装できます。 この命令は、iDevice 上で PROFIenergy コマンドを受信し、これらのコマンドを処理のためにユーザープログラムに転送します。 コマンドの処理後、ユーザープログラムは IO コントローラに確認を送信するために、再度「PE_I_DEV」命令を呼び出します。 これらの応答のために、「PE_I_DEV」命令に応答データを返すための対応する予備のブロックが各コマンドに用意されています。

PROFIenergy コマンド(PE コマンド)

PE コントローラは、PE コマンドを PE エンティティに送信します。 PE コマンドとして、PE エンティティを省電力モードに切り替える制御コマンド、またはステータスや測定値を読み出すコマンドが使用できます。

- 制御用 PI コマンド

  PROFIenergy は、「PE_Start_End」命令、または「PE_CMD」命令のいずれかを使用して実行可能な 2 つの制御コマンドをサポートしています。
  - Start_Pause: 最適な省電力モードを開始(PE Energy-saving mode)
  - End_Pause: 省電力モードを終了(PE_ready_to_operate モードに切り替え)
- ステータスまたは測定値読み取り用 PI コマンド

  次のステータスコマンドを「PE_CMD」命令で使用することで、制御システムから特定の状況情報を読み出すことが可能です。
  - PE_Identify: PE エンティティによってどの PE コマンドがサポートされているかを読み出します。
  - PEM_Status: PE エンティティの現在有効なモードの読み出し(たとえば PE_ready_to_operate)。
  - Query_Modes: 時間及び電力に関する情報を含む、サポートされているすべての省電力モードの概要を出力します。

- Query_Measurement: PE エンティティの測定値を出力します。

使用例

PROFIenergy 命令の使用例は、エントリ「PROFIenergy - SIMATIC S7 での省電力モード」の産業オンラインサポートに記載されています。

IO コントローラ

この章には下記に関する情報が記載されています：

- PE_START_END: 省電力モードの開始と終了 (S7-1500)
- PE_CMD: 省電力モードの開始と終了/ステータス情報の読み取り (S7-1500)
- PE_DS3_Write_ET200S: 電源モジュールの切り替え動作を設定します。 (S7-1500)
- PE_WOL:WakeOnLan を介した省電力モードの開始と停止 (S7-1500)
- PROFIenergy コマンド (S7-1500)

PE_START_END: 省電力モードの開始と終了

説明

「PE_START_END」命令は、指定された PE エンティティ(たとえば ET 200S)の省電力モードの開始および終了に使用します。

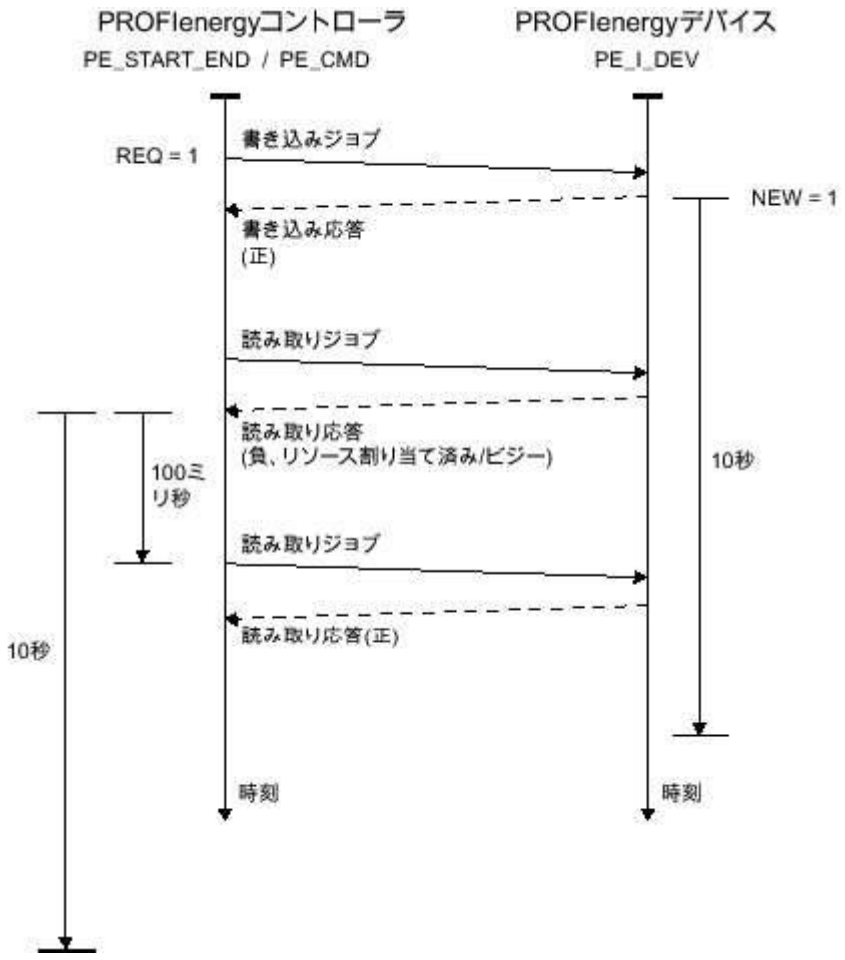
「PE_START_END」命令は省電力データを読み出す必要のないフィールドデバイスのみが、対応する PE デバイスに接続されている場合に、PE コントローラ内で使用することを推奨します。 または、「PE_CMD」を使用して省電力データを読み出すことも可能です。

省電力モードは、PE コントローラのユーザープログラムで設定します。「PE_START_END」命令が完了すると、PE エンティティは現在の省電力モードをレポートし、このデータをパラメータ PE_MODE_ID に出力します。

「PE_START_END」命令の書き込み、および読み出しジョブ

「PE_START_END」命令は「WRREC」を使用し、PROFIenergy コマンドを内部的に書き込みジョブとして PE エンティティに送信します。 "その後、「PE_START_END」は PE エンティティからの確認を待ちます。 確認データレコードは、100 ミリ秒ごとに「RDREC」命令で読み出されます。 PE エンティティから確認を受信するまで、この機能は読み出しジョブを 100 ミリ秒間隔で 10 秒間、呼び出しジョブを繰り返します。 PE エンティティの応答データは、「RDREC」命令でも読み出されます。

以下に、書き込み、および読み出しジョブのフローチャートを示します。



パラメータ

以下の表に、「PE_START_END」命令のパラメータを示します。

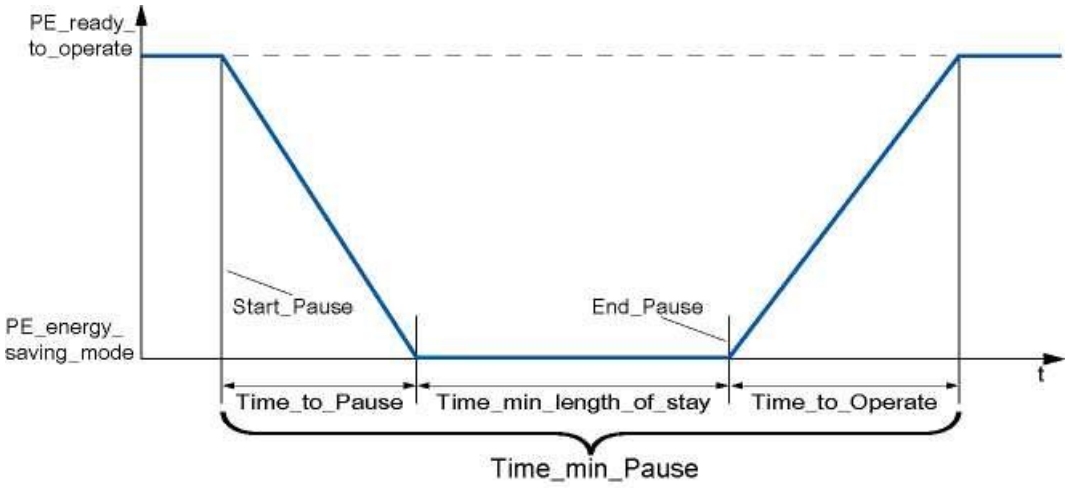
パラメータ	宣言	データタイプ	メモリ領域	説明
START	入力	BOOL	I、Q、M、D、L、または定数	PE コマンド「Start_Pause」をパラメータ ID でアドレス設定して PE エンティティに送信します。
END	入力	BOOL	I、Q、M、D、L、または定数	PE コマンド「End_Pause」をパラメータ ID でアドレス設定して PE エンティティに送信します。
ID	入力	HW_SUBMOD-ULE	I、Q、M、D、L、または定数	PE エンティティのアドレス PROFINET IO デバイス用ヘッドモジュールのハードウェア ID を使用します。 ハードウェア ID については、割り当てられた IO コントローラのシステム定数を参照してください。ヘッドモジュールの名前は、IO デバイスの名前とサフィックス[Head] (例:「IO_Device_1[Head]」)から構成されます。 PE エンティティが I デバイスである場合、代わりに転送領域のハードウェア識別子を使用します。
PAUSE_TIME	入力	TIME	I、Q、M、D、L、P、または定数	計画された一時停止時間。 • 範囲: T#1MS 終了 T#24D20H31M23S647MS • 開始値: T#0MS
VALID	出力	BOOL	I、Q、M、D、L	PE コマンドが正常に送信されました。
BUSY	出力	BOOL	I、Q、M、D、L	PE コマンドはまだ処理中です。
ERROR	出力	BOOL	I、Q、M、D、L	処理中にエラーが発生しました。 エラーメッセージは STATUS パラメータで出力されます。
STATUS	出力	DWORD	I、Q、M、D、L、P	ブロックステータス/エラー番号です(「STATUS パラメータ」を参照)。
PE_MODE_ID	出力	BYTE	I、Q、M、D、L、P	省電力モードの識別番号です(その一時停止時間の省電力レベル)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

PAUSE_TIME パラメータ

PAUSE_TIME パラメータを使用して、PE エンティティの省電力期間の既定の時間を設定します。PE エンティティは、一時停止時間が十分な長さであるか、そしてその一時停止時間を実行できるかチェックします。 最小の一時停止時間(Time_min_Pause)は、デバイスを省電力モードに切り替える時間

(Time_to_Pause)と再度動作モードに切り替える時間(Time_to_Operate)を合計した時間よりも長くなければなりません。



ET 200S は、計画された一時停止時間が ET 200S で保存されている最小の一時停止時間(PM-E_Pause_Min)以上であるかをチェックします。 これは 10 秒に固定されています。 より短い一時停止時間を使用すると、ET 200S の電源モジュール(PM-E)がオンのままになります。

モジュールは一時停止時間終了時に自動的に切り替わらず、「END」コマンドが送信されるまで OFF モードのままになります。 これにより、不要なピークロードの原因となる非同期の ON モードへの切り替えを防止します。

STATUS パラメータ

エラー情報は、STATUS 出力パラメータに出力されます。 ARRAY[1...4] of BYTE と解釈された場合、エラー情報は次の構造になります。

フィールドエレメント	名前	意味
STATUS[1]	Function_Num	エラーの原因 • B#16#**00**:エラーなし • B#16#**DE**: データレコード読み出し中のエラーです。 • B#16#**DF**: データレコード書き出し中のエラーです。 • B#16#**C0**: 命令、または内部的に使用されている「RDREC」および「WRREC」通信命令からのエラーメッセージです。
STATUS[2]	Error_Decode	エラー ID の場所 • **80**: IEC 61158-6 で定義された、またはアプリケーション固有の DPV1 エラーです。 • **FE**: DP/PNIO プロファイル - PROFIenergy 固有のエラーです。
STATUS[3]	Error_Code_1	エラー ID • Error_Decode = **80** の場合: ○ **80**: 入力パラメータ START および END の同時立ち上がりエッジ。 ○ **81**: CMD_PARAM および CMD_PARAM_LEN パラメータで長さの矛盾が発生しています。 ○ **82-8F**: その他のエラーメッセージ(予約済み)

		• Error_Decode = **FE** の場合: ○ **01**:「Service Request ID」が無効です。 ○ **02**: 不正な「Request_Reference」です。 ○ **03**:「Modifier」が無効です。 ○ **04**:「Data Structure Identifier RQ」が無効です。 ○ **05**:「Data Structure Identifier RS」が無効です。 ○ **06**:「PE energy-saving modes」がサポートされていません。 ○ **07**:「Response」が長すぎます(送信可能な最大長を超えています)。 ○ **08**:「Count」が無効です。 ○ **50**: 適切な「energy mode」を使用できません。 ○ **51**: 指定された時間値がサポートされていません。 ○ **52**:「PE_Mode_ID」が無効です。 ○ **53**: デバイスは動作中のため、「PE energy-saving mode」に切り替えできません ○ **54**: ファンクションはここでは使用できません。 不正なデバイスが設定されているか、 設定が不正です。 ○ **55** ~ **FF**:予約済み
STATUS[4]	Error_Code_2	製造元固有のエラー ID 拡張

注記

「RDREC」および「WRREC」命令のエラーメッセージ

「PE_START_END」命令は、通信に「WRREC」および「RDREC」命令を使用します。 これらの命令のエラーメッセージはフィールドエレメント STATUS[1]～STATUS[4]に出力されます。

「WRREC」および「RDREC」命令のエラーコードに関する説明については、対応する STATUS パラメータの説明を参照してください。

PE_CMD: 省電力モードの開始と終了/ステータス情報の読み取り

説明

PE コントローラで使用される「PE_CMD」命令は、PE エンティティの省電力モードの一時停止を開始、または終了します。 追加情報および電力測定は、「PE_CMD」を使用して PE エンティティから読み出すことも可能です。

この命令は、電力測定が読み出される PE デバイスが割り当てられている PE コントローラに使用するのが最適です。 これ以外の場合でも、「PE_START_END」命令は、一時停止の開始および終了に使用できます。

PROFIenergy コマンド(PE コマンド)の転送

「PE_CMD」命令は、PROFIenergy コマンドを PE エンティティに転送します。

この命令は、追加コマンドが将来的に PROFIenergy プロファイルに追加される場合でも使用可能です。 現在の PROFIenergy プロファイルで使用可能なコマンドは、CMD および CMD_MODIFIER パラメータの説明に記載されています(「CMD および CMD_MODIFIER パラメータ」の表を参照)。

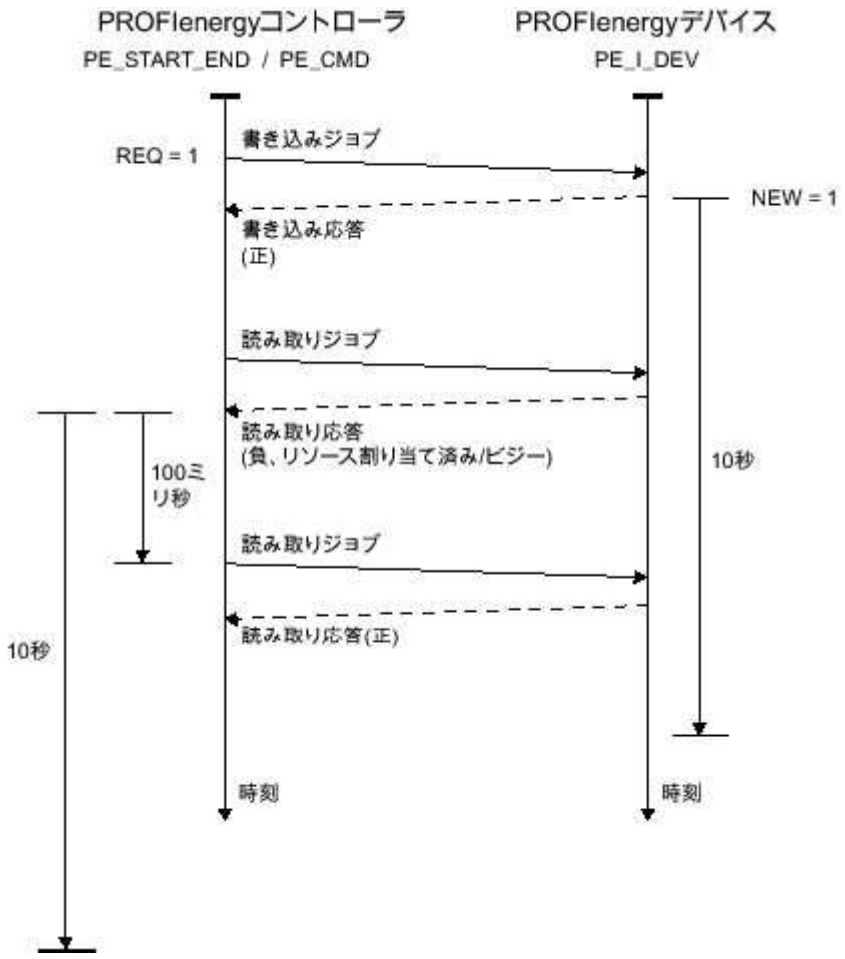
- この命令を使用して PE エンティティに転送される個々の PE コマンドは、定義済みの「Service_Request_ID」に割り当てられます。 Service_Request_ID 01～05、および 16 は、CMD パラメータに割り当てられています。
- PE コマンド 04 (Query_Modes)および 16 (Query_Measurement)の 2 つは、CMD_MODIFIER パラメータでより詳細に定義されます。
- 追加の値は CMD_PARA パラメータで個々の PE コマンドに転送されます(個々の PE コマンドの説明を参照)。 CMD_PARA_LEN パラメータは、CMD_PARA パラメータのデータ長を定義します。

このコマンドは、妥当性テストなしで転送されます。 PE エンティティの応答データは、VARIANT ポインタによってアドレス指定される RESPONSE_DATA データ領域に保存されます(応答フレームの内容に関する情報については、各 PE コマンドの説明を参照してください)。

「PE_CMD」命令の書き込み、および読み出しジョブ

「PE_CMD」命令は「WRREC」を内部的に使用し、PROFIenergy コマンドを書き込みジョブとして PE エンティティに送信します。 "その後、「PE_CMD」は PE エンティティからの確認を待ちます。 確認データレコードは、100 ミリ秒ごとに「RDREC」命令で読み出されます。 PE エンティティから確認を受信するまで、この機能は読み出しジョブを 100 ミリ秒間隔で 10 秒間、呼び出しジョブを繰り返します。 PE エンティティの応答データは、「RDREC」命令でも読み出されます。

以下に、書き込み、および読み出しジョブのフローチャートを示します。



パラメータ

以下の表に、「PE_CMD」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで PE コマンドの転送を開始します。
ID	Input	HW_SUBMOD-ULE	I、Q、M、D、L、または定数	PE エンティティのアドレス PROFINET IO デバイス用ヘッドモジュールのハードウェア ID を使用します。 ハードウェア ID については、割り当てられた IO コントローラのシステム定数を参照してください。ヘッドモジュールの名前は、IO デバイスの名前とサフィックス[Head] (例:「IO_Device_1[Head]」)から構成されます。 PE エンティティが I デバイスである場合、代わりに転送領域のハードウェア識別子を使用します。
CMD	Input	BYTE	I、Q、M、D、L、P、または定数	PROFIenergy プロファイルに準拠した PROFIenergy コマンドの

				Service-Request-ID (「CMD および CMD_MODIFIER パラメータ」を参照)。 PROFIenergy プロファイルの拡張子に続けてサービス要求 ID を追加可能です。
CMD_MODIFIER	Input	BYTE	I、Q、M、D、L、P、または定数	PROFIenergy サブコマンド (CMD=3、または CMD=16 の場合のみ。「CMD および CMD_MODIFIER パラメータ」を参照)。 PROFIenergy プロファイルの拡張子に続けてサブコマンドを追加可能です。
CMD_PARA	Input	VARIANT	I、Q、M、D、L	PE コマンドに関するパラメータ • Get mode: PE_mode_ID • Get measurement values: List of Measurement_Ids 完全な Service Data Request が格納されます。
CMD_PARA_LEN	Input	INT	I、Q、M、D、L、P、または定数	コマンドパラメータの実際の長さ(<=CMD_PARA で定義された長さ)が命令によって検証されます。
RESPONSE_DATA	InOut	VARIANT	I、Q、M、D、L	PROFIenergy 情報 コマンドによっては、ブロックヘッダーを含んだ完全な応答フレームとなる場合があります。 注記: バッファが小さすぎる場合、VARIANT ポインタで指定されるバイト数が入力されます。
VALID	Output	BOOL	I、Q、M、D、L	コマンドが正常に送信されました。
BUSY	Output	BOOL	I、Q、M、D、L	コマンドはまだ処理中です。
ERROR	Output	BOOL	I、Q、M、D、L	処理中にエラーが発生しました。
STATUS	Output	DWORD	I、Q、M、D、L	ブロックステータス/エラー番号です(「STATUS パラメータ」を参照)。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

CMD および CMD_MODIFIER パラメータ

CMD	CMD_MODIFIER	PROFIenergy コマンド	説明
01	0	Start_Pause	省電力モードを開始、または他の省電力モードに切り替えます。

02	0	End_Pause	省電力モードを終了します。
03	1	Query_Modes - 省電力モードのリスト	サポートされている省電力モードを出力します。
	2	Query_Modes - モードの取得	現在有効な省電力モードの属性を出力します。
04	0	PEM_Status	省電力モードのステータスを照会します。
05	0	PE_Identify	サポートされている PE コマンドの番号と詳細を読み出します。
16	1	Query_Measurement - Get_Measurement_List	PE エンティティでサポートされている測定値のリスト。
	2	Query_Measurement - Get_Measurement_Values	PE エンティティの測定値を出力します。

STATUS パラメータ

エラー情報は、STATUS 出力パラメータに出力されます。 ARRAY[1...4] of BYTE と解釈された場合、エラー情報は次の構造になります。

フィールドエレメント	名前	意味
STATUS[1]	Function_Num	エラーの原因 • B#16#**00**:エラーなし • B#16#**DE**: データレコード読み出し中のエラーです。 • B#16#**DF**: データレコード書き出し中のエラーです。 • B#16#**C0**: 内部の「RDREC」および「WRREC」通信命令からのエラーメッセージです。
STATUS[2]	Error_Decode	エラー ID の場所 • **80**: IEC 61158-6 で定義された、またはアプリケーション固有の DPV1 エラーです。 • **FE**: DP/PNIO プロファイル - PROFIenergy 固有のエラーです。
STATUS[3]	Error_Code_1	エラー ID • Error_Decode = **80** の場合: ○ **81**: CMD_PARA および CMD_PARA_LEN パラメータの長さが矛盾しているか、データレコードの最大長(4095 バイト)を超過しています。 ○ **82-8F**: その他のエラーメッセージ(予約済み) • Error_Decode = **FE** の場合: ○ **01**:「Service Request ID」が無効です。 ○ **02**: 不正な「Request_Reference」です。 ○ **03**:「Modifier」が無効です。 ○ **04**:「Data Structure Identifier RQ」が無効です。 ○ **05**:「Data Structure Identifier RS」が無効です。 ○ **06**:「PE energy-saving modes」がサポートされていません。 ○ **07**:「Response」が長すぎます(送信可能な最大長を超えています)。 ○ **08**:「Count」が無効です。

		○ 50: 適切な省電力モード(energy mode)を使用できません。 ○ 51: 指定された時間値がサポートされていません。 ○ 52: 「PE_Mode_ID」が無効です。
STATUS[4]	Error_Code_2	製造元固有のエラー ID 拡張

注記

「RDREC」および「WRREC」命令のエラーメッセージ

「PE_CMD」命令は、通信に「WRREC」および「RDREC」命令を使用します。 これらの命令のエラーメッセージはフィールド工レメント STATUS[1]～STATUS[4]に出力されます。

「WRREC」および「RDREC」命令のエラーコードに関する説明については、対応するステータスパラメータの説明を参照してください。

PE_DS3_Write_ET200S: 電源モジュールの切り替え動作を設定します。

説明

「PE_DS3_Write_ET200S」命令は、電源モジュールの切り替え動作に関する基本設定を ET 200S に送信します。 ET 200S の 8 スロットまでの切り替え動作(たとえば電源モジュールなど)を「PE_DS3_Write_ET200S」命令で定義することができます。

注記
この命令は PROFIenergy プロファイルの一部ではありませんが、SIMATIC 固有の機能を補完します。

パラメータ

次の表に、「PE_DS3_Write_ET200S」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
ENABLE	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジがデータレコードの転送をトリガします。 データレコードは、電圧が OFF または ON に切り替わった後、再度転送する必要があります。
ID	Input	HW_SUB-MODULE	I、Q、M、D、L、または定数	ET 200S のアドレス アドレスはハードウェアコンフィグレーションから取得可能です。
SLOT_NO_X	Input	INT	I、Q、M、D、L、P、または定数	切り替え可能な電源モジュールのスロット番号 X。
FUNC_X	Input	INT	I、Q、M、D、L、P、または定数	このスロット内のモジュールのファンクションです。 FUNC_X パラメータを使用して、PM-E (ET 200S の電源モジュール)の切り替え動作を定義します: • FALSE: ○「PAUSE_START」の場合: -PM-E への影響なし -PM-E はオンのまま ○「PAUSE_STOP」の場合: -PM_E を再度オンに切り替え • TRUE: ○「PAUSE_START」の場合: -PM_E をオフに切り替え ○「PAUSE_STOP」の場合: -PM-E を再度オンに切り替え
BUSY	Output	BOOL	I、Q、M、D、L	転送がまだ完了していません。

DONE	Output	BOOL	I、Q、M、D、L	転送がエラーなしで完了しました。
ERROR	Output	BOOL	I、Q、M、D、L	転送完了(エラーあり)。
STATUS	Output	DWORD	I、Q、M、D、L、P	エラー番号(「PE_Start_End」命令の STATUS パラメータを参照)

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

PE_WOL:WakeOnLan を介した省電力モードの開始と停止

この章には下記に関する情報が記載されています：

PE_WOL の説明

説明

「PE_WOL 」命令は、PROFIenergy コマンド「Start_Pause」および「End_Pause」を PROFINET I/O システムの複数の PROFIenergy 有効なデバイスに送信します。

複数の PE デバイスは命令を使用して調整されます。ただし PE デバイスは、UDP 接続で「Wake on LAN」機能をサポートしています。

「PE_WOL」命令は、統合されたイーサネットインターフェースを備えた CPU 上でのみ実行することができます。この CPU は、約 400 KB のサイズのブロックをロードできるはずです。 Ethernet CP 経由で接続されている PROFINET I/O システムでは、ブロックを使用することはできません。

命令「PE_WOL 」非同期で実行されます。

定義: Wake on LAN

Wake on LAN 機能を使うと、特殊なイーサネットパケットを受信する時に、ほぼシャットダウンに近い状態からデータ処理装置を回復させることができます。

このプロシージャの場合、データ処理装置はこのようなパケットを受信できるネットワークコントローラを装備している必要があります。

このパケット(Magic Packet™)の書式は特殊です。 ネットワークアダプタの MAC アドレスが 15 倍格納されています。

デバイスの選択

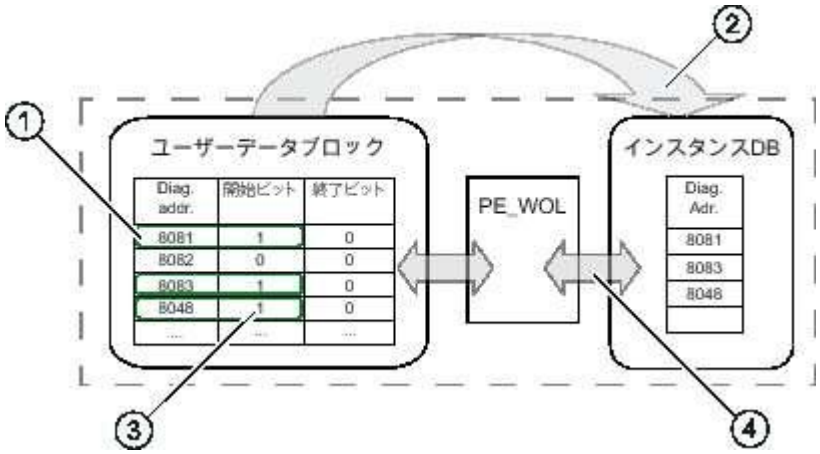
デバイスは、パラメータ PENERGY (タイプ: "PE_PLUS")のユーザーデータブロックを使用して選択します。ユーザー DB は、複数のデバイスを処理するためのデータベースです。

「PE_WOL」の初期化前に、ユーザー DB に少なくとも以下の情報を格納する必要があります。

- PROFINET I/O システムの ID
- 「Wake On LAN」に使用される接続のデータ
- 「Wake On LAN」に使用されるポート番号
- 各デバイス
  - 休止時間(PauseTime)
  - PE_SLEEP_MODE (EnableSleep)へのデバイスの切り替え

「PE_WOL」命令を COM_RST パラメータを使用して初期化します。 ユーザー DB に格納されたジョブは、初期化後順次処理されます。

以下の図に、PE コマンド「Start_Pause」が複数のデバイスにどのように送信されるかを示します。



(1) 手順 1: シャットダウンされるデバイスのビット「CmdStartPause」がユーザーにより「1」にセットされます。

(2) 手順 2: シャットダウンされるデバイスの診断アドレス(CmdStartPause = 「1」)はキューにリンクされます。

(3) 手順 3: ジョブがリンクされると、ビット「CmdStartPause」は自動的にリセットされます。

(4) 手順 4: ジョブがリンクされると、直ちに命令「PE_WOL」がジョブを処理します。

PROFIenergy コマンド「CmdStartPause」または「CmdEndPause」は、START および END パラメータにより PROFINET IO システムで認識されたすべてのデバイスに送信することができます。

ジョブ処理のステータスと処理中に考えられるエラーが STATUS パラメータにより出力されます。

ユーザー DB を使用した命令の操作

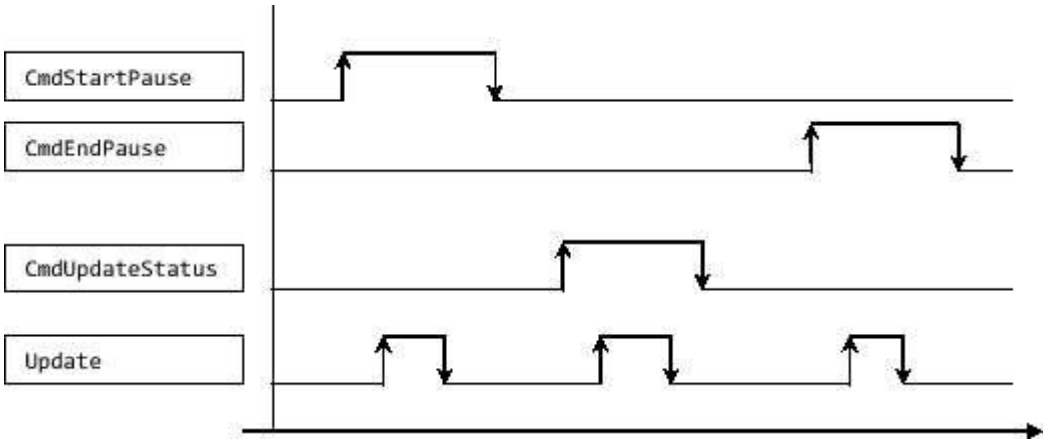
命令「PE_WOL」の操作は、ユーザー DB を経由した場合のみ実行できます。 この場合、基本的なプロシージャが適用されます。

1. デバイスで実行されるコマンドの選択:
   - START_PAUSE (ユーザー DB 内の「CmdStartPause」)
   - ENDE_PAUSE (ユーザー DB 内の「CmdEndPause」)
   - UPDATE_STATUS (ユーザー DB 内の「CmdUpdateStatus」)
2. 更新のビットの設定(ユーザー DB のヘッダーでの「Update」)

   2 つの更新の間で「更新」に偽(FALSE)を指定して少なくとも 1 つの CPU サイクルを渡す必要があります。それ以外の場合、エッジ検出は保証できません。

PE コマンドの優先順位付け

以下の図に、考えられるコマンドの時間的順序を示します。



前のコマンド呼び出しが正常に終了したか、エラーが発生して終了したかに関係なく、順次処理されます。

2 つのコマンド「CmdEndPause」および「CmdUpdateStatus」が同時にセットされる場合、1 つのコマンドのみが実行されます。 ブロック内で優先順位付けがあります。

- コマンド「CmdStartPause」は優先度が一番高く、選択されると常に実行されます。
- コマンド「CmdEndPause」は 2 番目に高い優先度です。
- コマンド「CmdUpdateStatus」は最も低い優先度です。

3 つのコマンドすべてが同時に設定されると、実行されないコマンドは事前に選択されたままになります。 この場合、次の立ち上がりエッジは次のコマンドを実行します。

パラメータ

次の表に、「PE_WOL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
COM_RST	Input	BOOL	I、Q、M、D、L	ブロックをリセットし、再初期化を実行します。 ここで真(TRUE)が設定されると、初期化が開始されますが、完全に終了しません。 立ち下がりエッジのみが初期化を続行し、初期化後、通常の動作モードに切り替わります。
START	Input	BOOL	I、Q、M、D、L	信号立ち上がりエッジは、この機能をサポートする検出されたデバイスすべてで「CmdStartPause」PROFIenergy コマンドを実行します。
END	Input	BOOL	I、Q、M、D、L	信号立ち上がりエッジは、この機能をサポートする検出されたデバイスすべてで「CmdEndPause」PROFIenergy コマンドを実行します。
PENERGY	InOut	PE_PLUS	D	複数のデバイスを処理するデータベースを格納した DB へのポインタ。
STATUS	Output	DWORD	I、Q、M、D、L	命令の現在のステータスのステータス/エラー番号(「STATUS パラメータ」を参照)。

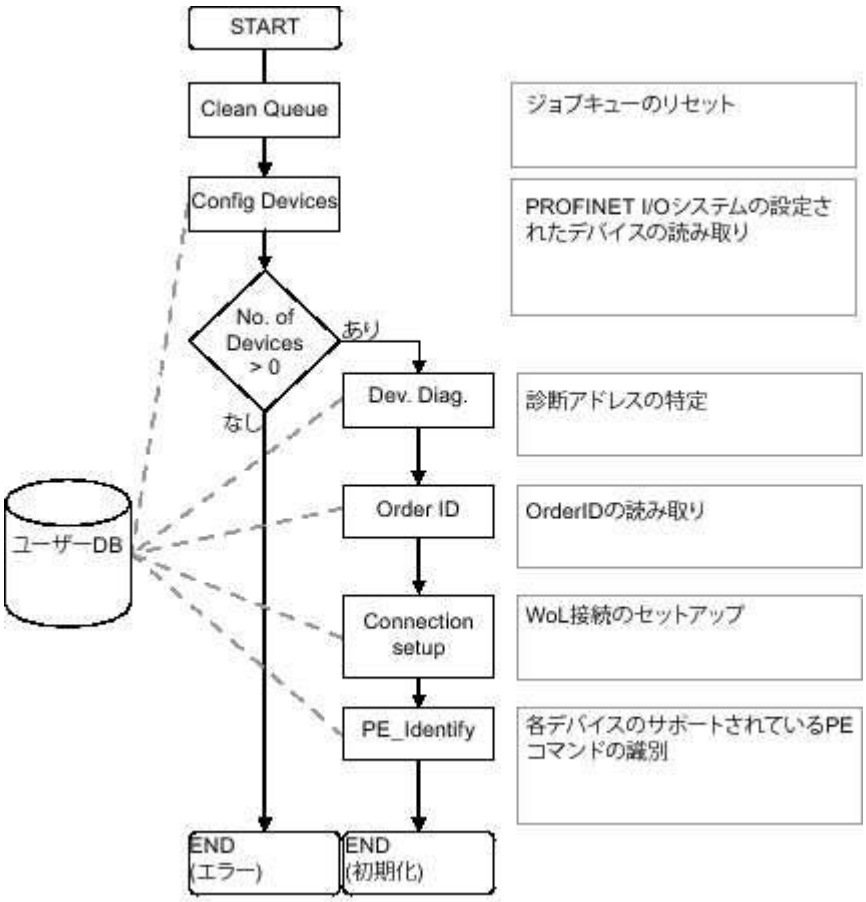
有効なデータタイプの追加情報については、「**有効なデータタイプの概要**」を参照してください。

COM_RST パラメータ

初期化ルーチンのシーケンス

「PE_WOL」命令の初期化を COM_RST パラメータにより開始します。

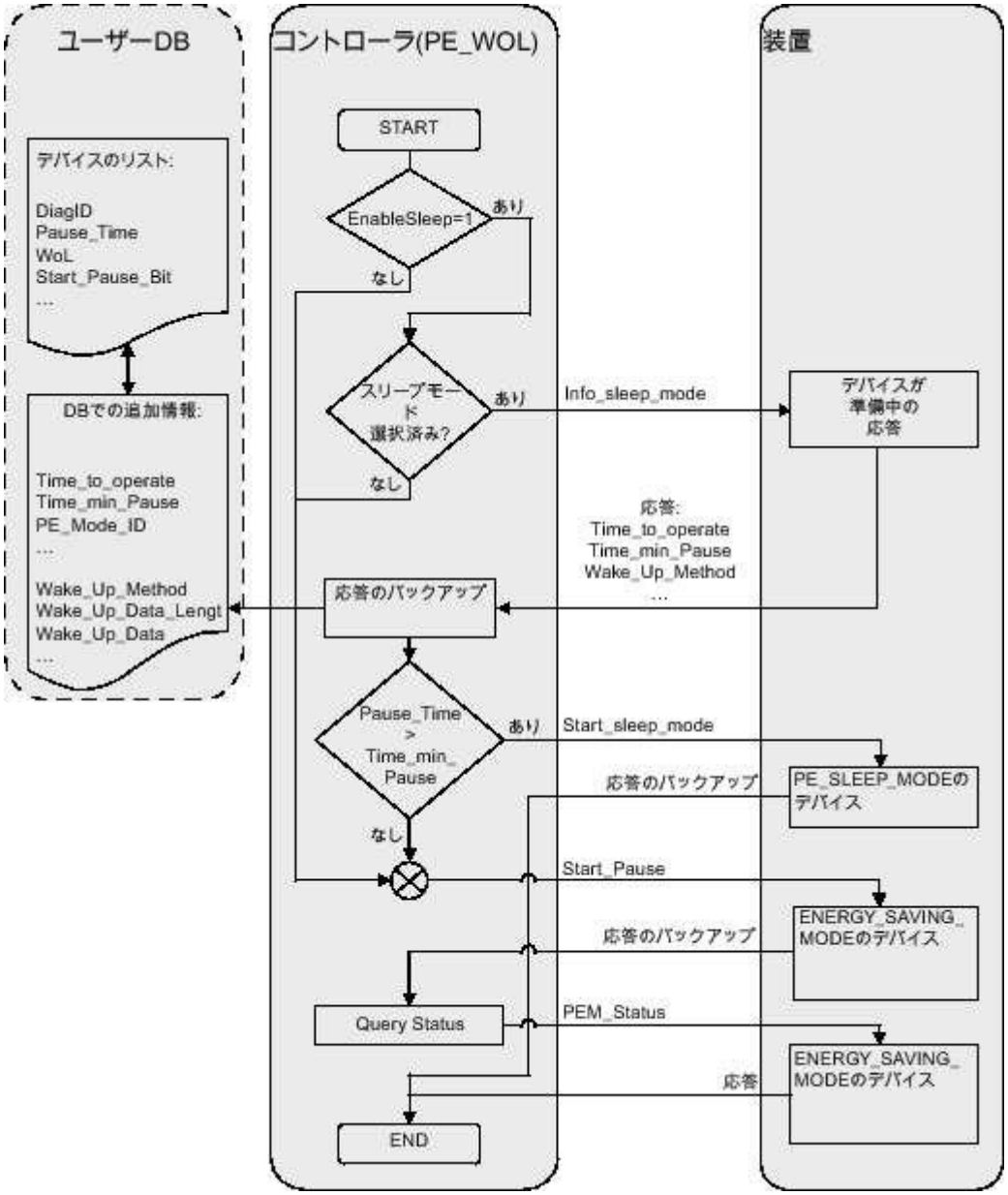
下のフローダイアグラムは、初期化ルーチンを示します。



START パラメータ

CmdStartPause コマンドのシーケンス

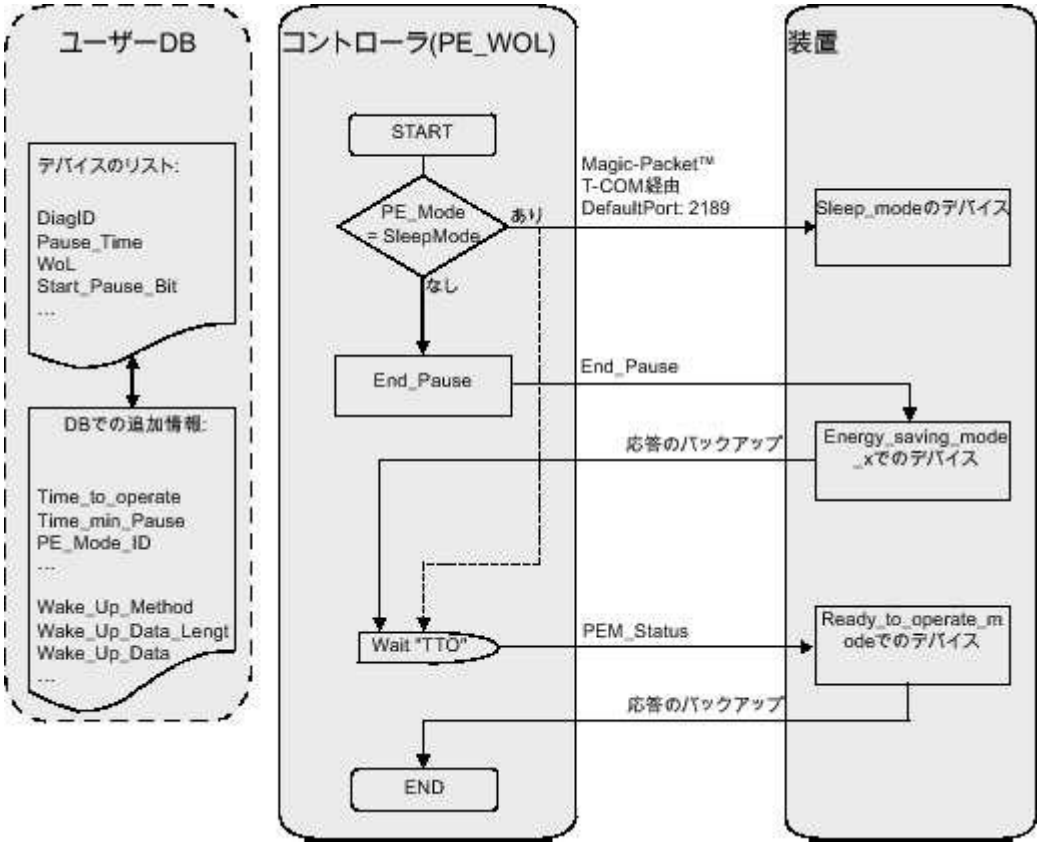
以下の図に、CmdStartPause コマンドの実行時に、内部で使用されるファンクションとデバイスでの操作を示します。



END パラメータ

CmdEndPause コマンドのシーケンス

以下の図に、CmdEndPause コマンドの実行時に、内部で使用されるファンクションとデバイスでの操作を示します。



PENERGY パラメータ

PENERGY パラメータのデータブロック

「PE_WOL」PROFIenergy 命令のユーザー DB は、複数のデバイスを処理するためのデータベースを表します。

データブロックは通常 2 つの部分に分解されます。 これらは、次です。

- ヘッダー 110 バイト
- 最大 256 個のデバイスのデバイスセクションで、それぞれ 100 バイト(Device) これには、以下のものがあります。
  - デバイス固有のデータ(Device)
  - PROFIenergy 特定のデータ(PE)
  - ジョブ処理のためのデータ(Task)
  - ユーザーデータ(UserData)

このデータブロックは、最適化されたアクセスで動作します。

接続パラメータ「Connection」

「PE_WOL」命令は「ユーザー通信を開く」の領域で接続リソースを予約します。 この領域は UDP 接続として使用されます。 このため、データブロックに以下のパラメータを定義する必要があります。

- 接続 ID (「Connection.id」パラメータ)

  接続 ID は範囲 1~4095 の整数です。ファームウェアによって割り当てられた通信リソース、たとえばバッファの送受信などを識別するのに使われます。
  接続 ID は CPU 全体で一意である必要があります。
- [Wake on LAN]ファンクション(「Header.PortNo」パラメータ)に使用されるポート番号。

  [Wake On LAN]パケットの送信元である UDP ポートの番号。 これらのポート番号は通信リソースの一部であり、接続 ID によってファームウェアで識別されて提供されます。ここで使用されるポート 2189 の既定の設定は、IANA では現在割り当てされません。 ポート番号は接続設定に転送され、パラメータ「Connection.local_tsap_id[1]」に適用されます。 ポート番号とリモートポートの長さは、パラメータ「Connection.rem_tsap_id[1]」および「Connection.rem_tsap_id_LEN」を使用して定義し、手動で割り当てる必要があります。
- インターフェース ID(「Connection.local_device_id」パラメータ)

  インターフェース ID も接続記述子の一部です。 この ID は接続に使用するための CPU インターフェースを識別します。 考えられる有効な値がいくつかあります。

  しかし、使用されている CPU やインターフェースに適合させる必要があります。
  - B#16#01(サブスロット IF1 のイーサネットインターフェース使用の S7-1500 CPU、ET 200S CPU または WinAC RTX の場合)
  - B#16#02(CPU 315(F)-2PN/DP または CPU 317(F)-2PN/DP の場合)
  - B#16#03(CPU 319(F)-2PN/DP の場合)
  - B#16#05(CPU 41x(F)-3PN/DP の場合)
  - B#16#06(サブスロット IF2 のイーサネットインターフェース使用の WinAC RTX の場合)
  - B#16#0B(サブスロット IF3 のイーサネットインターフェース使用の WinAC RTX の場合)
  - B#16#0F(サブスロット IF4 のイーサネットインターフェース使用の WinAC RTX の場合)

データブロックの構造体

データブロックの構造体は以下の通りです。

名前	データタイプ	OFF-SET	コメント
Header	PE_HEADER	-	ヘッダー情報
Update (1)	BOOL	-	データ領域での変更を示す信号 • True = ユーザーによる変更を示します。 • False = 変更の用途を示します。
Initialized	BOOL	-	初期化が完了したことを示す信号 • True = 初期化が終了済み。 • False = ブロックが初期化されていないことを示します。
LinkUp	BOOL	-	イーサネットインターフェースの設定が正常に終了したことを示します。 • True = インターフェースが使用準備完了。 • False = インターフェースがまだ設定されていません。
LinkDown	BOOL	-	設定されていないインターフェースを示します。 • True = インターフェースがまだ設定されていません。 • False = インターフェースが現在設定中であるか、現在設定済みです。
PROFINET_ID (1)	INT	-	PROFINET I/O システムの ID
Reserved	ARRAY [1..37] OF BYTE	-	予約済み
LastDeviceID	INT	-	この PROFINET I/O システムの最大の Device-ID が収納されています。
PortNo (1)	WORD	-	[Wake on LAN]ファンクションに使用されるポート番号(既定 = 2189).
Connection	TCON_Param	-	[Wake on LAN]接続の接続設定を格納します。
BLOCK_LENGTH	UInt	-	構造体の長さ(常に B#16#40).
ID (1)	CONN_OUC	-	接続 ID
CONNEC-TION_TYPE (1)	USINT	-	接続タイプ(UDP = B#16#13)
ACTIVE_EST (1)	BOOL	-	アクティブ接続の確立(常に UDP ではパッシブ)
LOCAL_DE-VICE_ID (1)	USINT	-	インターフェース ID を格納します(CPU 依存)
LO-CAL_TSAP_ID_LEN (1)	USINT	-	独自/ローカル UDP ポートの長さをバイト単位で格納します。
REM_SUB-NET_ID_LEN (1)	USINT	-	未使用(常に B#16#00)
REM_STADDR_LEN (1)	USINT	-	リモート IP アドレスまたは B#16#00 の長さを格納します。

REM_TSAP_ID_LEN (1)	USINT	-	リモート UDP ポートの長さをバイト単位で格納します。 長さ指定は手動で入力する必要があります。
NEXT_STADDR_LEN (1)	USINT	-	既定のルータアドレスの長さ(関連なし)を格納します。
LOCAL_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	独自/ローカルポート番号を格納します。 値は初期化中にパラメータ PortNo から適用されます。
REM_SUBNET_ID (1)	ARRAY[1..6] OF USINT	-	未使用(常に B#16#00)
REM_STADDR (1)	ARRAY[1..6] OF USINT	-	リモート IP アドレスを格納します。
REM_TSAP_ID (1)	ARRAY[1..16] OF BYTE	-	リモート UDP ポート番号を格納します。 ポート番号は手動で入力する必要があります。
NEXT_STADDR (1)	ARRAY[1..6] OF BYTE	-	対象外
SPARE (1)	WORD	-	
Device	ARRAY[1..256] OF PE_DEVICE	-	デバイスの配列
Device	PE_DEV	-	デバイスごとのデータを格納します。
DeviceID	HW_DEVICE	-	デバイスのハードウェア識別子。 ハードウェアコンフィグレーションによる割り当て
PE_EntityID	HW_IO	-	PROFIenergy エンティティのハードウェア識別子。 ハードウェアコンフィグレーションによる割り当て
MACAdr	ARRAY[1..6] OF BYTE	-	デバイスの MAC アドレスを格納します。
IPAdr	ARRAY[1..4] OF BYTE	-	デバイスの IP アドレスを格納します。
OrderID_MxLen	Byte	-	OrderID の最大長が収納されています。
OrderId_ActLen	Byte		OrderID の実際の長さが収納されています。
OrderID_Data	ARRAY[1..20] OF CHAR		デバイスの OrderID を格納します。
PE	PE_PE	-	PROFIenergy 特有のデータ
ModeID	BYTE	-	PE_MODE_IDProfienergy 仕様による
Result	BYTE	-	PROFIenergy 仕様に従った PE ErrorCode。
PauseTime (1)	TIME	-	ミリ秒単位のアイドル時間を格納します。
TimeToPause	TIME	-	デバイスが一時停止モードになるのにかかる時間を格納します。
TimeToOperate	TIME	-	デバイスが動作モードになるのにかかる時間を格納します。
MinSleepTime	TIME	-	デバイスの最小時間を PE_SLEEP_MODE に格納します。

SleepToOperate	TIME	-	デバイスが PE_SLEEP_MODE から動作モードに進むのにかかる時間を格納します。
StatusOperate	BOOL	-	デバイスの動作モードを示します。
StatusPause	BOOL	-	デバイスが一時停止モードであることを示します。
StatusSleep	BOOL	-	デバイスの PE_SLEEP_MODE を示します。
StatusTransitOK	BOOL	-	エネルギー状態から別の状態への移行を示します。
StatusInTransit	BOOL	-	現在の状態移行を示します。
StatusTransitNOK	BOOL	-	状態の変更が正常に終了しなかったことを示します。
StatusError	BOOL	-	デバイスでのエラーを示します。
StatusRetryEx	BOOL	-	コマンドの実行が正常に終了しなかったことを示します。 このコマンドを実行する試みはもう行いません。
CmdStartPause (1)	BOOL	-	このデバイスの START_PAUSE コマンドをキューに入れます。
CmdEndPause (1)	BOOL	-	このデバイスの END_PAUSE コマンドをキューに入れます。
CmdUpdateStatus (1)	BOOL	-	このデバイスの PEM_STATUS コマンドをキューに入れます。
EnableSleep (1)	BOOL	-	このデバイスには PE_SLEEP_MODE が可能です。 • True = アイドル時間が十分である場合、デバイスは PE_SLEEP_MODE に移行します • False = デバイスは PE_SLEEP_MODE には移行できません。
Services	WORD	-	すべてのサポートされている PROFIenergy サービスを示します。
UserData (2)	ARRAY[1..24] OF BYTE	-	ユーザー定義のデータ
Task	PE_TASK	-	ジョブ処理
Cmd	BYTE	-	ジョブ処理の内部ビット
CmdJ	BYTE	-	ジョブ処理の内部ビット
TimeStart	BOOL	-	遅延時間を開始します。
TimeStarted	BOOL	-	遅延時間が開始しました。
TimeDone	BOOL	-	遅延時間が経過したことを示します。
Done	BOOL	-	このジョブが完了したことを示します。
DelayedCmd	BOOL	-	遅延したコマンドがまだ保留状態であることを示します。
IsV1_0	BOOL	-	このデバイスは仕様を準拠していることを示します。 V1.0 デバイス
IsWakeOnLAN	BOOL	-	このデバイスは[Wake On LAN]により起動されることを示します。
RetryCount	BYTE	-	PE_COMMANDS の繰り返しカウンタ

Duration	TIME	-	ミリ秒単位の遅延の値を格納します。
StartTime	TIME	-	遅延時間の開始時刻を格納します。
MachineState	INT	-	ジョブの内部ステータスを格納します。

(1) ユーザーが入力。

(2) ユーザーが使用できる。

STATUS パラメータ

STATUS パラメータ

STATUS パラメータでの出力値は 3 つの領域に分割されます。

- ビット 31～24: MESSAGE
- ビット 23～16: LOCATION
- ビット 15～0: INFORMATION

次の表に、3 つの領域の各エラーコードの意味を示します。

MESSAGE に可能な値

エラーコード (W#16#...)	説明
00	エラーは発生していません。
50	初期化された命令。
51	PROFINET I/O システムの設定を決定。
52	この命令は、PROFINET I/O システム内で設定された任意のデバイスを決定することができませんでした。
53	設定されたデバイスの論理アドレスを決定。
54	デバイスのインターフェース情報の読み取り。
55	設定されたデバイスの I&M データ(データレコード 0 のみ)の決定。
56	UDP 経由の「Wake on LAN」MagicPaket™送信のための PROFINET インターフェースの設定。
57	接続済みデバイスの PROFIenergy 機能の決定。
62	無効な PROFINET I/O システム ID を検出しました。 原因となる番号が[INFORMATION]フィールドに表示されます。
70	命令が初期化され、ジョブを処理します。 INFORMATION フィールドの値は、現在有効なジョブ数を含みます。
80	この命令は、ジョブの処理中に初期化を実行できません。 インスタンス DB またはユーザー DB のいずれかが再ロードされた場合、通常この状況が発生します。
FF	不明なエラーが発生しました。

LOCATION の考えられる値

エラーコード (W#16#...)	説明
00	この命令は初期化されていないか、アイドル状態であるかいずれかです。

70	この命令は、ジョブを待機しています。
71	この命令はジョブリストにジョブを添付しています。
72	この命令はジョブ送信を準備しています。
73	この命令はジョブをデバイスに送信中です。
74	この命令はデバイスからの応答を待機しています。
75	この命令はデバイスの応答を評価しています。
76	この命令はジョブリストからジョブを削除しています。
FF	不明なエラーが発生しました。

INFORMATION の考えられる値

エラーコード (W#16#...)	**説明**
0000	追加の情報はなく、ジョブは有効ではありません。
0001 -00FF	1 – 255 ジョブは現在処理されています。
7000	COM_RST による再初期化が開始されましたが、まだ完了していません。
8001	パラメータ 1 でのエラー
8002	パラメータ 2 でのエラー
8003	パラメータ 3 でのエラー
8004	パラメータ 4 でのエラー
8005	パラメータ 5 でのエラー。このエラーは、相互接続されていないか、またはユーザー DB への相互接続が無効な場合に報告されます。 考えられる原因: • ユーザー DB が小さすぎます。 • ユーザー DB は書き込み保護されています。 • RAM にユーザー DB がありません。 • ユーザー DB が使用している CPU に無効です。
8085 ~ 80CE	接続設定時のエラー。 内部的に使用される TCON 命令のエラーメッセージが出力 されます。 エラーメッセージの説明は、STATUS パラメータのテーブルで確認できます。
8100	許可されている最大 256 個のジョブを超えるジョブを挿入しようとしました。 これは一時的なエラーで、多少のジョブを完了すると解決されます。 挿入されたジョブは受け入れられないため、もう一度挿入する必要があります。
8200	無効またはサポートされていない PROFIenergy コマンド(PE_COMMAND)を送信しようとしました。
8400	接続 ID が許容範囲内にありません。 初期化が中止されました。 [Wake on LAN]の接続設定の ID をチェックします。 パラメータ PENERGY > Header > Connection > ID のデータブロックを参照してください。
84xx	通信エラーが発生しました。 エラーを生成したデバイスの数が「xx」に出力されます。

85xx	エラーがデバイス xx から返されました。 エラーを生成したデバイスの数が「xx」に出力されます。
8600	要求されたウェークアップ方法は現在サポートされていません。
FFFF	不明なエラーが発生しました。

PROFIenergy コマンド

この章には下記に関する情報が記載されています：

メッセージフレームの構造

PROFIenergy プロファイルに準拠した応答フレームの構造

以下の表は、PROFIenergy プロファイルに準拠した応答フレームの基本構造を指名しています。 応答フレームは、全般セクション(Header)と固有のセクション(Service Data Response)で構成されています。 応答フレームの個々のセクションの内容は、関連する PROFIenergy コマンドの説明に記載されています。

ブロック定義	属性	値	データタイプ	説明
BlockHeader	BlockType	801 hex	WORD	
	BlockLength		WORD	フィールド BlockType および BlockLength を考慮しないバイト数。
	BlockVersionHigh	1 hex	BYTE	
	BlockVersionLow	0 hex	BYTE	
Response Header	Service_Request_ID	1hex ~ FF hex	BYTE	実行される PI コマンドの ID。 PE エンティティによって処理される PE コマンドの ID は、応答フレームに返されます。 • 01: Start_Pause • 02: End_Pause • 03: Query_Modes • 04: PEM_Status • 05: PE_Identify • 06 ~ 09:予約済み • 16: Query_Measurement • 11 ~ CF:予約済み • D0 ~ FF:製造元固有
	Request_Reference	1hex ~ FF hex	BYTE	照会/応答のペアを特定する固有の番号(応答時にサーバーによって返されます)。
Service Header Response	Status	1hex ~ FF hex	BYTE	PI コマンド実行の有無に関する情報。 • 00: 予約済み • 01: 完了 • 02: 完了(エラーあり) • 03: 不完全なデータ • 04 ~ CF:予約済み • D0 ~ FF: Service_Request_ID によって異なる

	Data_Structure_Identifier_RS	1_{hex} ~ FF hex	BYTE	• 00: 予約済み • 01 ~ FF: データ構造体は Service_Request_ID によって異なる • 0xFF - エラー
Service Data Response				Service-Request-ID によって異なる • サービス応答 ID については、「PE_CMD」命令の CMD および CMD_MODIFIER パラメータを参照してください。 • 応答フレームの個々の内容は、PE コマンドの説明に記載されています(たとえば「Start_Pause」コマンドを参照)。

PI コマンド「Start_Pause」

説明

PE コマンド「Start_Pause」を使用して、省電力モードを開始します。 コマンド Start_Pause は、以下の操作にも使用できます。

- PE を「準備完了」状態(PE_ready_to_operate)から省電力モード(PE_energy_saving_mode)に切り替える。
- PE エンティティは自動的に省電力モードを切り替えます。

  省電力モードを切り替えると、電力消費は増加、または減少します。

PI コマンド「Start_Pause」の呼び出し

コマンド「Start_Pause」は、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	1	PE コマンド「Start_Pause」の呼び出し。
CMD_MODIFIER	0	コマンド「Start_Pause」のコマンド呼び出しに関するその他の事項はありません。
CMD_PARA_LEN	4	CMD_PARA パラメータの長さ、4 バイト。
CMD_PARA	VARIANT	「Pause_Time」(TIME)の値に対する VARIANT ポインタ。

応答フレーム(Service Data Response)

PE エンティティの以下の応答フレームデータは、パラメータ RESPONSE_DATA に参照されるデータブロックに書き込まれます(「PE_CMD」命令を参照)。

属性	値	データタイプ	説明
PE_Mode_ID	1 ~ 255	BYTE	省電力モードの識別番号
予約済み	0	BYTE	-

PI コマンド「End_Pause」

説明

PE コマンド「End_Pause」を使用して、PE エンティティの省電力モードを終了します。

PE コマンド「End_Pause」の呼び出し

コマンド「End_Pause」は、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	2	PE コマンド「End_Pause」の呼び出し。
CMD_MODIFIER	0	コマンド「End_Pause」のコマンド呼び出しに関するその他の事項はありません。
CMD_PARA_LEN	0	CMD_PARA パラメータの長さ、0 バイト。
CMD_PARA	対象外	-

応答フレーム(Service Data Response)

PE エンティティ応答フレームの以下のデータは、RESPONSE_DATA パラメータに参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
Time_to_operate	-	DWORD	「PE_ready_to_operate」モードへの切り替えの推定時間。

PI コマンド「Query_modes」 - 「List_Energy_Saving_Mod

説明

PE コマンド「Query_modes」およびサブコマンド(修飾子)「List_Energy_Saving_Modes」を使用して、PE エンティティがサポートしているすべての省電力モード(PE_Mode_ID)を出力します。

照会結果は RESPONSE_DATA パラメータが参照するデータブロックに応答フレームとして書き込まれます。

PE コマンド「Query_modes」 - 「List_Energy_Saving_Modes」の呼び出し

コマンド「List_Energy_Saving_Modes」は、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	3	PE コマンド「Query_modes」の呼び出し。
CMD_MODIFIER	1	コマンド呼び出しの仕様: サブコマンド「List_Energy_Saving_Modes」を選択して、サポートされている省電力モードの数とタイプを出力します。
CMD_PARA_LEN	0	CMD_PARA パラメータの長さ、0 バイト。
CMD_PARA	対象外	-

応答フレーム(Service Data Response)

PE エンティティ応答フレームの以下のデータは、RESPONSE_DATA パラメータに参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
Number_of_PE_Mode_IDs	1	BYTE	省電力モードの数。
PE_Mode_IDs	-	Array [...] of BYTE	サポートされている省電力モードの ID の配列。 個々の ID の意味は、PE エンティティによって異なります。

PI コマンド「Query_modes」 - 「Get_Mode」

説明

PE コマンド「Query_modes」およびサブコマンド(修飾子)「Get_Mode」を使用し、現在有効な省電力モードの属性を出力します。

PE コマンド「Query_modes」 - 「Get_Mode」の呼び出し

「PE_CMD」によるこのコマンドの呼び出しには、以下のパラメータを使用します。

パラメータ	値	説明
CMD	3	PE コマンド「Query_modes」の呼び出し
CMD_MODIFI-ER	2	コマンド呼び出しの仕様: サブコマンド「Get_Mode」を選択して、現在有効なモードのステータスを出力します。
CMD_PARA_LEN	1	CMD_PARA パラメータの長さ、1 バイト。
CMD_PARA	VARIANT	VARIANT pointer to the value for PE_MODE_ID.

応答フレーム(Service Data Response)

PE エンティティ応答フレームの以下のデータは、RESPONSE_DATA パラメータに参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
PE_Mode_ID	• 0 "「PE_power_off」モード • 1...254 PE エンティティの省電力モード(メーカー固有) • 255 "「PE_ready_to_operate」モード	BYTE	現在有効な省電力モードの ID。
PE_Mode_Attrib-utes	ビット 0: • = 0: 静的なエネルギー消費および時間値のみを使用できます。 • = 1: 動的なエネルギー消費および時間値を使用できます。 ビット 1 ~ 7: • 予約済み	BYTE	

Time_min_Pause [1]	日付なしの時間差	DWORD	PI モードの最小一時停止時間。最小一時停止時間は、以下の属性値の合計です。 • Time_to_Pause • Time_to_operate • Time_min_length_of_stay 「PAUSE_TIME パラメータ」(「PE_START_END: 省電力モードの開始と終了」命令)の説明を参照してください。
Time_to_Pause [1]	日付なしの時間差	DWORD	スイッチオフ時間: 省電力モードの呼び出しから省電力モード開始までの時間(PE_ready_to_operate から PE_energy_saving_mode への移行時間)。 スイッチオフ時間は PE によって異なります。
Time_to_operate [1]	日付なしの時間差	DWORD	スイッチオン時間: 省電力モード(PE_energy_saving_mode)から動作準備完了モード(PE_ready_to_operate)までの移行時間。 PE エンティティは、出力操作で経過時間を動的に計算します。
Time_min_length_ of_stay [1]		DWORD	PE エンティティの省電力モードの最小有効期間。
Time_max_length_ of_stay [1]		DWORD	PE エンティティの省電力モードの最大有効期間。
Mode_Power_ Consumption [2]		REAL	省電力モードが有効な PE エンティティの電力消費。 単位: kW
Energy_ Consumption_ to_pause [2]		REAL	動作準備完了モード(PE_ready_to_operate)から省電力モード(PE_energy_saving_mode)への移行中の PE エンティティの電力消費。 単位: kWh
Energy_ Consumption_ to_operate [2]		REAL	省電力モード(PE_energy_saving_mode)から動作準備完了モード(PE_ready_to_operate)への移行中の PE エンティティの電力消費。 単位: kWh
[1] 期間が無限の場合、「0xFFFFFFFF」が出力されます。 期間がゼロの場合「0」が出力されます。 [2] PE エンティティによって電力消費のデータが定義されていない場合、値として「0.0」が出力されます。			

PI コマンド「PEM_Status」

説明

PE コマンド「"PEM_Status」を使用して、現在有効な PE エンティティ省電力モードのステータスを照会します。

PE コマンド「PEM_Status」の呼び出し

コマンド「PEM_Status」は、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	4	PE コマンド「PEM_Status」の呼び出し。
CMD_MODIFIER	0	コマンド「PEM_Status」のコマンド呼び出しに関するその他の事項はありません。
CMD_PARA_LEN	0	CMD_PARA パラメータの長さ、0 バイト。
CMD_PARA	対象外	-

応答フレーム(Service Data Response)

PE エンティティ応答フレームの以下のデータは、RESPONSE_DATA パラメータに参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
PE_Mode_ID_Source	• 0 "「PE_power_off」モード • 1～254 PE エンティティの省電力モード(メーカー固有) • 255 "「PE_ready_to_operate」モード	BYTE	PE コマンド送信前の PE エンティティのモード。
PE_Mode_ID_Destination	• 0 "「PE_power_off」モード • 1～254 PE エンティティの省電力モード(メーカー固有) • 255 "「PE_ready_to_operate」モード	BYTE	PE コマンド実行後の PE エンティティのモード。
Time_to_operate	日付なしの時間差	DWORD	スイッチオン時間: 省電力モード(PE_energy_saving_mode)から

			動作準備完了モード (PE_ready_to_operate)までの移行時間。 PE エンティティは、出力中に経過時間を動的に計算します。
Remain-ing_time_to_destination	日付なしの時間差	DWORD	他のモードへの切り替えまでの残り時間。
Mode_Power_Consumption		REAL	省電力モードが有効な PE エンティティの電力消費。 単位: kW
Energy_Consumption_to_Destination		REAL	現在の PI 以降の電力消費 単位: kWh
Energy_Consumption_to_operate		REAL	省電力モード(PE_energy-saving mode)から動作準備完了モード (PE_ready_to_operate)への移行中の PE エンティティの電力消費。 単位: kWh

PI コマンド「PE_Identify」

説明

PE コマンド「PE_Identify」を使用して、PE エンティティがサポートする PE コマンドの数および詳細を読み出します。 サポートされているコマンド数は、PE エンティティによって異なります。PE_Identify 自体が PE コマンドであるため、3 つ以上のサポートされている PE コマンドが信号立ち上がりエッジで出力されます。Start_Pause、End_Pause および PE_Identify.

PE コマンド「PE_Identify」の呼び出し

コマンド「PE_Identify」は、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	5	コマンド「PE_Identify」の呼び出し
CMD_MODIFIER	0	コマンド「PE_Identify」のコマンド呼び出しに関するその他の事項はありません。
CMD_PARA_LEN	0	CMD_PARA パラメータの長さ、0 バイト。
CMD_PARA	対象外	-

応答フレーム(Service Data Response)

PE エンティティ応答フレームの以下のデータは、RESPONSE_DATA パラメータに参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
Count [1]	6	BYTE	サポートされている PROFIenergy コマンドの数
Start_Pause	1	BYTE	最初のサポートされている PE コマンド (Service_Request_ID)
End_Pause	2	BYTE	...
Query_Modes	3	BYTE	...
PEM_Status	4	BYTE	...
PE_Identify	5	BYTE	...
Query_Measurement	16	BYTE	最後のサポートされている PE コマンド (Service_Request_ID)
[1] サポートされているコマンド数は、メーカーおよび使用している PE エンティティによって異なります。 上記の値は、6 つの PE コマンドすべてがサポートされている応答フレームでの例です。			

PI コマンド「Query_Measurement」 - 「Get_Measurement_list」

説明

PE コマンド「Query_Measurement」およびサブコマンド(修飾子)「Get_measurement_list」を使用して、PE エンティティによってサポートされている特定の測定値を照会します。 サポートされている測定値は、RESPONSE_DATA パラメータによって参照されるデータブロック内のリストとして出力されます。

PE コマンド「Query_Measurement」 - 「Get_Measurement_list」の呼び出し

このコマンドは、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	16	コマンド「Query_Measurement」の呼び出し
CMD_MODIFIER	1	コマンド呼び出しの仕様: サブコマンド「Get_Measurement_List」を選択して、サポートされている測定値のリストを出力します。
CMD_PARA_LEN	0	CMD_PARA パラメータの長さ、0 バイト。
CMD_PARA	対象外	-

応答フレーム(Service Data Response)

PE エンティティ応答フレームの以下のデータは、RESPONSE_DATA パラメータに参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
Count	-	BYTE	測定 ID の数
reserved	-	BYTE	
...			
Measurement_ID	-	WORD	最初のサポートされている Measurement_ID。Measurement_ID はメーカーに依存します。 詳細は、各 PE エンティティのマニュアルを参照してください。
Accuracy_Domain	-	BYTE	「精度ドメイン」の表を参照してください。
Accuracy_Class	-	BYTE	「精度クラス」の表を参照してください。
Range	-	REAL	測定値のスケール終了値を指定します(精度ドメイン 1 のみ)。 属性範囲は、属性 Measurement_ID で定義されたものと同じ単位を使用します(使用できる単位は各 Measurement_ID につき 1 つのみ)。
...			
Measurement_ID	-	WORD	最後のサポートされている Measurement_ID
Accuracy_Domain	-	BYTE	「精度ドメイン」の表を参照してください。
Accuracy_Class	-	BYTE	「精度クラス」の表を参照してください。

Range	-	REAL	測定値のスケール終了値を指定します(精度ドメイン 1 のみ)。 属性範囲は、属性 Measurement_ID で定義されたものと同じ単位を使用します(使用できる単位は各 Measurement_ID につき 1 つのみ)。

精度ドメイン

精度ドメイン	説明
0	予約済み
1	精度の誤差は、スケール終了値のパーセンテージとして出力されます。 想定される誤差のパーセンテージは、精度クラスに分割されます(精度ドメイン 1 および 2 の精度クラスの表を参照)。
2	精度の誤差は、現在の測定値のパーセンテージとして出力されます。 想定される誤差のパーセンテージは、精度クラスに分割されます(精度ドメイン 1 および 2 の精度クラスの表を参照)。
3	測定精度は、IEC 61557-12 規格に基づいています。 外部センサのないパフォーマンス測定およびモニタリングデバイス(PMD)のファンクションパフォーマンスクラス、および外部センサ付き PMD のシステムパフォーマンスクラスは、精度ドメイン 3 の精度クラス」の表に記載された通りコーディングされています。
4	精度のエントリは、EN 50470-3 規格、第 8 章に基づいています(精度ドメイン 4 の精度クラス)。

精度クラス

精度ドメイン 1 および 2 の精度クラス

精度クラス	0	1	2	3	4	5	6	7	8
意味	予約済み	0.01%	0.02%	0.05%	0.1%	0.2%	0.5%	1%	1.5%
精度クラス	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**>15**	
意味	2%	2.5%	3%	5%	10%	20%	>20%	未定義	

精度ドメイン 3 の精度クラス

精度クラス	0	1	2	3	4	5	6	7	8
意味	予約済み	0,02	0,05	0,1	0,2	0,5	1	1,5	2
精度クラス	**9**	**10**	**11**	**12**	**13**	**14**	**>13**		
意味	2,5	3	5	10	20	20%	未定義		

精度ドメイン 4 の精度クラス

精度クラス	0	1	2	3	4	5	6	>7
意味	予約済み	0,5	1,0	1,5	2,0	2,5	3,0	未定義

PI コマンド「Query_Measurement」-「Get_Measurement_values」

説明

PE コマンド「Query_Measurement」およびサブコマンド(修飾子)「Get_measurement_values」を使用して、PE エンティティによってサポートされている測定値のリストを出力します。 測定値は、RESPONSE_DATA パラメータによって参照されるデータブロック内のリストとして出力されます。

PE コマンド「Query_Measurement」 - 「Get_Measurement_values」の呼び出し

このコマンドは、以下のパラメータを持つ「PE_CMD」命令で呼び出されます。

パラメータ	値	説明
CMD	16	コマンド「Query_Measurement」の呼び出し
CMD_MODIFIER	2	コマンド呼び出しの仕様: コマンド「Get_Measurement_Values」を選択して、サポートされている測定値のリストを出力します。
CMD_PARA_LEN	0	測定値の数によって異なります。 パラメータの長さは、属性 count および転送される測定値用の属性の長さの合計によって決まります。
CMD_PARA	VARIANT	照会される測定値のリストを伴ったデータ構造体に対する VARIANT ポインタ(「CMD_PARA パラメータ」を参照)。

パラメータ CMD_PARA

CMD_PARA パラメータ VARIANT でポインタによって指定された構造体は、以下の設定がされている必要があります。

属性	値	データタイプ	説明
Count	-	BYTE	測定値の数(Measurement-IDs)
reserved	0	BYTE	未使用
Measurement_ID	-	WORD	必要な最初の測定値
...			
Measurement_ID	-	WORD	必要な最後の測定値

応答フレーム(Service Data Response)

PE エンティティの以下の応答フレームデータは、パラメータ RESPONSE_DATA に参照されるデータブロックに書き込まれます(「PE_CMD」を参照)。

属性	値	データタイプ	説明
Count [1]	-	BYTE	測定値の数(Measurement-IDs)
reserved	0	BYTE	未使用

Length_of_Structure	2 ~ 65535	WORD	構造体の長さ(バイト単位)
Measurement_Data_Structure_ID	1 = 唯一の値	BYTE	以降の構造体を定義。
Measurement_ID	0 ~ 65535	WORD	サポートされている測定値の ID。
Status_of_Measurement_Value	1 ~ 3	BYTE	測定値のステータス: • 1: 使用可能 • 2: サポートされていません • 3: 無効
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	データタイプ TimeOfDay にタイムスタンプを追加。
...			
Length_of_Structure	-	WORD	構造体の長さ(バイト)
Measurement_Data_Structure_ID	-	BYTE	以降の構造体を定義。
Measurement_ID	-	WORD	サポートされている測定値の ID。
Status_of_Measurement_Value	-	BYTE	測定値のステータス: • 1: 使用可能 • 2: サポートされていません • 3: 無効
Transmission_Data_Type	-	REAL	
End_of_demand	-	TOD	データタイプ TimeOfDay にタイムスタンプを追加。
[1] 必要な測定値のデータ長が、プロトコル層の PDU (プロトコルデータユニット)のサイズを超える場合、データは完全には転送されず、サポートされている測定値のみが出力されます。			

iDevice/iSlave

この章には下記に関する情報が記載されています：

- PE_I_DEV: I デバイスでの PROFIenergy コマンドの制御 (S7-1500)
- 「PE_I_DEV」命令の補助ブロック (S7-1500)

PE_I_DEV: I デバイスでの PROFIenergy コマンドの制御

説明

「PE_I_DEV」命令を使用して、インテリジェント IO デバイス(iDevice)内で PROFIenergy プロファイルを処理します。 ET 200S などの標準 PROFIenergy 互換 IO デバイス内でファームウェアによって実行されるファンクションは、「PE_I_DEV」命令および対応する補助ブロックによって iDevice 内で実行されます。

- 「PE_I_DEV」命令は、iDevice のユーザープログラムによって周期的に呼び出され、すべての PROFIenergy コマンドを受信します。
- PROFIenergy 応答は、補助ブロックを設定することで生成されます。 一時停止中の応答は、完全なプログラミングが可能です。 応答データは 10 秒以内で返される必要があります。10 秒を超えると IO コントローラ内のこの命令の STATUS パラメータで「State conflict 0x80B5」が発生します。

この命令を使用するにあたって、PROFIenergy 規格に関する特別な知識は必要ありません。

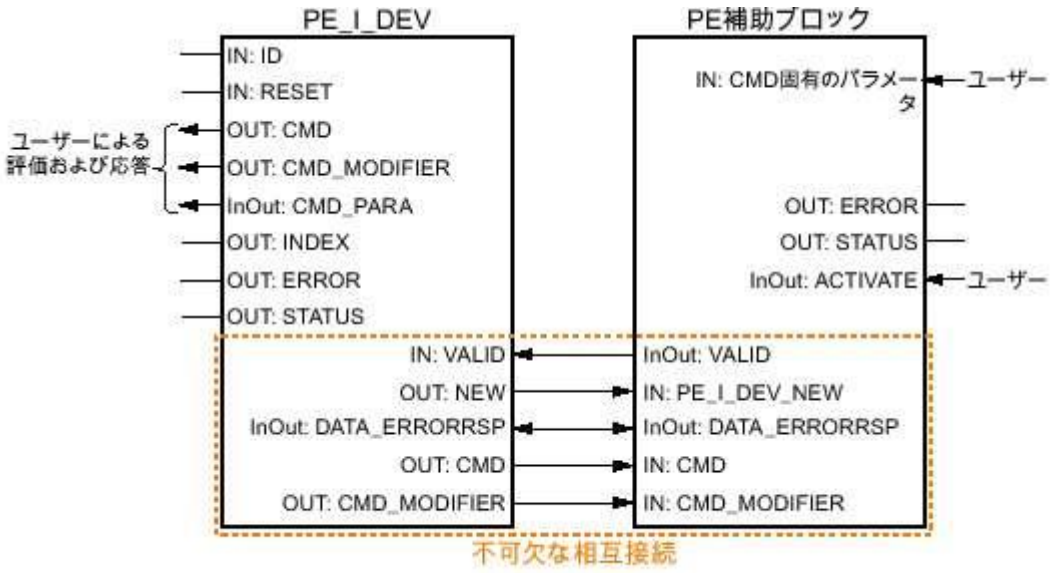
PROFIenergy 補助ブロック(PE 補助ブロック)

PE 補助ブロックを使用して、応答フレームを生成します。 関連するブロックの入力パラメータに、応答データを(プレーンテキストで)入力します。

- 各 PROFIenergy コマンドに対して、肯定応答用の対応する補助ブロックが用意されています。
  - PE コマンド「Start_Pause」: PE_Start_RSP
  - PE コマンド「End_Pause」: PE_End_RSP
  - PE コマンド「Query_modes」 - 「List_Energy_Saving_Modes」: PE_List_Modes_RSP
  - PE コマンド「Query_modes」 - 「Get_Mode」: PE_Get_Mode_RSP
  - PE コマンド「PEM_Status」: PE_PEM_Status_RSP
  - PE コマンド「PE_Identify」: PE_Identify_RSP
  - PE コマンド「Query_Measurement」-「Get_Measurement_list」: PE_Measurement_List_RSP
  - PE コマンド「Query_Measurement」-「Get_Measurement_values」: PE_Measurement_Value_RSP
- PROFIenergy コマンドに関係なく、否定応答用の共有補助ブロックが用意されています(PE_Error_RSP を参照)。

補助ブロックの相互接続

「PE_I_DEV」命令および補助ブロックは連携します。 単純に相互接続するパラメータもあります。次の図に、どのパラメータを相互接続する必要があるかを示します。



パラメータ

以下の表に、「PE_I_DEV」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RESET	Input	BOOL	I、Q、M、D、L、または定数	命令をリセットします。
ID	Input	HW_SUB-MODULE	I、Q、M、D、L、または定数	転送領域のアドレス。PROFIenergy のデータと共に IO コントローラに提供されます。 ハードウェア識別子は、システム定数で確認できます。
VALID	Input	BOOL	I、Q、M、D、L、または定数	PROFIenergy コントローラへの応答データの準備が完了しており、送信可能です。
CMD_PARA	Output	VARIANT	I、Q、M、D、L	以下のパラメータ用: • Get mode: PE_mode_ID • Get measurement values: Measurement_ID のリスト(読み出されるタグの ID のリスト。任意の時点で 1 つまたは複数のタグの読み出しが可能)。 最大長: 234 バイト
DATA_ERRORRSP	InOut	VARIANT	I、Q、M、D、L	PROFIenergy コントローラへの確認データを含むデータ領域に対するポインタ。 これは、補助ブロックでも使用するポインタと対応するする必要があります。

INDEX	Output	INT	I、Q、M、D、L	PROFIenergy レコードのデータレコード番号(0x80A0 にセット)
CMD	Output	INT	I、Q、M、D、L	PROFIenergy プロファイルに準拠した PROFIenergy コマンドの Service-Request-ID (「CMD および CMD_MODIFIER パラメータ」を参照)。 PROFIenergy プロファイルの拡張子に続けて PE コマンド(Service-Request-IDs)を追加可能です。
CMD_ MODIFIER	Output	INT	I、Q、M、D、L	PROFIenergy サブコマンド: • CMD=3 または CMD=16 の場合のみ。「CMD および CMD_MODIFIER パラメータ」を参照してください。 • 他のすべてのコマンドの場合: "0". PROFIenergy プロファイルの拡張子に続けてサブコマンドを追加可能です。
NEW	Output	BOOL	I、Q、M、D、L	PROFIenergy コントローラから新しいデータが使用できます。
ERROR	Output	BOOL	I、Q、M、D、L	コマンド完了(エラーあり)。
STATUS	Output	DWORD	I、Q、M、D、L	エラー情報(「STATUS パラメータ」を参照)。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

CMD および CMD_MODIFIER パラメータ

CMD	CMD_ MODIFIER	PROFIenergy コマンド	説明
01	0	Start_Pause	省電力モードを開始、または他の省電力モードに切り替えます。
02	0	End_Pause	省電力モードを終了します。
03	1	Query_Modes - List energy saving Modes	サポートされている省電力モードを出力します。
	2	Query_Modes - Get Mode	現在有効な省電力モードの属性を出力します。
04	0	PEM_Status	省電力モードのステータスを照会します。
05	0	PE_Identify	サポートされている PE コマンドの番号と詳細を読み出します。
16	1	Query_Measurement - Get_Measurement_List	PE エンティティでサポートされている測定値のリスト。
	2	Query_Measurement - Get_Measurement_Values	PE エンティティの測定値の出力。

STATUS パラメータ

エラー情報は、STATUS 出力パラメータに出力されます。 ARRAY[1...4] of BYTE と解釈された場合、エラー情報は次の構造になります。

フィールドエレメント	名前	意味
STATUS[1]	Function_Num	エラーの原因 • B#16#**00**:エラーなし • B#16#**DE**: データレコード読み出し中のエラーです。 • B#16#**DF**: データレコード書き出し中のエラーです。 • B#16#**C0**: 「PE_I_DEV」命令、または内部的に使用されている「PRVREC」および「RCVREC」通信命令からのエラーメッセージです。
STATUS[2]	Error_Decode	エラー ID の場所 • **80**: IEC 61158-6 で定義された、またはアプリケーション固有の DPV1 エラーです。
STATUS[3]	Error_Code_1	エラー ID(Error_Decode = **80** の場合): • **B1**: 書き込みの長さのエラー(書き込みの長さにエラーがあるか、データタイプ VARIANT による長さの情報が不十分です)。
STATUS[4]	Error_Code_2	PROFINET エラーの場合: IO コントローラエラーメッセージの出力。PROFINET エラーが発生しなかった場合、STATUS[4]の値は「0」となります。

注記

「PRVREC」および「RCVREC」命令のエラーメッセージ

「PE_I_DEV」命令は、通信に「PRVREC」および「RCVREC」命令を使用します。 これらの命令のエラーメッセージはフィールドエレメント STATUS[1]～STATUS[4]に出力されます。

「PRVREC」および「RCVREC」命令のエラーコードの意味については、対応する STATUS パラメータの説明を参照してください。

「PE_I_DEV」命令の補助ブロック

この章には下記に関する情報が記載されています：

PE_Error_RSP: コマンドへの否定応答の生成

説明

要求されたコマンドが全般的、一時的を問わずサポートされていない場合、補助ブロック「PE_Error_RSP」(Response with failure)は否定応答を生成します。 応答の生成は、要求されたコマンドに依存しません。

パラメータ

次の表に、「PE_Error_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。 パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
ERROR_CODE	Input	BYTE	I、Q、M、D、L、または定数	エラー番号
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。
VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。

STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

PE_Start_RSP: 一時停止の開始時にコマンドへの応答を生成

説明

補助ブロック「PE_Start_RSP」(一時停止開始)は、PE コマンド Start_Pause への応答を生成します。 この命令は、デバイスが切り替えた省電力モードを返します(PE_MODE_ID パラメータ)。

一時停止の長さによって応答が異なる場合は、これを省電力モードにフィードバックとして返すことが可能です(PE_Mode_ID = 1 の場合は短時間の一時停止、PE_Mode_ID = 2 の場合はより長い一時停止、など)。

パラメータ

次の表に、「PE_Start_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
PE_MODE_ID	Input	BYTE	I、Q、M、D、L、または定数	プロセスが仮定する PE モード
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。
VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメ

				ータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

PE_End_RSP: 一時停止の終了時にコマンドへの応答を生成

説明

補助ブロック「PE_End_RSP」は、PE コマンド End_Pause への応答を生成します。 返される応答は、現在のモードから「Ready_To_Operate」モードへの切り替えに必要な時間です。

パラメータ

次の表に、「PE_End_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
Time_to_Operate	Input	DWORD	I、Q、M、D、L、または定数	現在のモードから「Ready_To_Operate」への切り替えに必要な時間。
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。
VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領

				域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

PE_List_Modes_RSP: 照会された省電力モードを応答として生成

説明

補助ブロック「PE_List_Modes_RSP」は、PE コマンド List_Energy_Saving_Modes への応答を生成します。 生成される応答には、サポートされている省電力モードの数と ID が含まれています。

パラメータ

次の表に、「PE_List_Modes_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
CMD_MODIFIER	Input	INT	I、Q、M、D、L、または定数	PROFIenergy サブコマンド(CMD=3、または CMD=16 の場合にのみ評価)。このパラメータは、「PE_I_DEV」命令の CMD_MODIFIER 出力パラメータと相互接続されている必要があります。
Number_of_PE_Mode_IDs	Input	BYTE	I、Q、M、D、L、または定数	サポートされている省電力モードの数。 許可されている値: 1～254
PE_MODE_ID	Input	VARIANT	I、Q、M、D、L	サポートされている省電力モードの ID(PE_Mode_ID)が格納されている領域をポイントします。 有効な範囲: 1～254
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。
VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。

				この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

PE_Get_Mode_RSP: 照会された省電力データを応答として生成

説明

補助ブロック「PE_Get_Mode_RSP」は、コマンド Get_Mode への応答を生成します.応答フレームには、省電力モードの時間、パフォーマンスまたは省電力データが含まれます。

パラメータ

次の表に、「PE_Get_Mode_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
CMD_MODIFI-ER	Input	INT	I、Q、M、D、L、または定数	PROFIenergy サブコマンド(CMD=3、または CMD=16 の場合にのみ評価)。このパラメータは、「PE_I_DEV」命令の CMD_MODIFIER 出力パラメータと相互接続されている必要があります。
PE_Mode_ID	Input	BYTE	I、Q、M、D、L、または定数	現在有効な省電力モードの ID。
Time_min_Pause	Input	DWORD	I、Q、M、D、L、または定数	この PE 省電力モードの最小一時停止時間。 これは、以下の 3 つのパラメータの合計です。 • Time_to_Pause • Time_to_operate • Time_min_length_of_stay
Time_to_Pause	Input	DWORD	I、Q、M、D、L、または定数	START パラメータ(「PE_I_DEV」を参照)のエッジから要求された PE 省電力モードに達するまでの時間。
Time_to_Oper-ate	Input	DWORD	I、Q、M、D、L、または定数	「PE_ready_to_operate」までの最大スイッチオン時間。 「Time_to_oper-ate」パラメータは、関連する計算に直接使用することが可能です。 値は静的な最大値であるか、または PE 工

				ンティティによって動的に計算されます。
Time_min_Lenght_ of_stay	Input	DWORD	I、Q、M、D、L、または定数	この PE モードの PE エンティティの最小滞留時間。
Time_max_Lenght_of_stay	Input	DWORD	I、Q、M、D、L、または定数	この PE モードの PE エンティティの最大滞留時間。
Mode_Power_ Consumption	Input	DWORD	I、Q、M、D、L、または定数	現在の PE モードでの電力消費(単位 kW)。
Energy_Con-sum_ to_Pause	Input	DWORD	I、Q、M、D、L、または定数	「PE_ready_to_operate」から現在の PE モードまでの電力消費(単位 kWh)。
Energy_Con-sum_ to_operate	Input	DWORD	I、Q、M、D、L、または定数	現在の PE モードから「PE_ready_to_operate」までの電力消費(単位 kWh)。
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。
VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

PE_PEM_Status_RSP: PEM ステータスを応答として生成 

説明

補助ブロック「PE_PEM_Status_RSP」は、コマンド PEM_Status への応答を生成します。

パラメータ

次の表に、「PE_PEM_Status_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
PE_MODE_ID_Source	Input	BYTE	I、Q、M、D、L、または定数	PEM_STATUS の Source および Destination。 値: • 0x00: PE_POWER_OFF • 0x01 – 0xFE:メーカー固有 • 0xFF: PE_READY_TO_OPERATE
PE_MODE_ID_Destination	Input	BYTE	I、Q、M、D、L、または定数	
Time_to_Operate[1]	Input	DWORD	I、Q、M、D、L、または定数	「PE_ready_to_operate」までの最大スイッチオン時間。 "「Time_to_operate」パラメータは、関連する計算に直接使用することが可能です。 値は静的な最大値であるか、または PE エンティティによって動的に計算されます。
Remaining_time_to_destination[1]	Input	DWORD	I、Q、M、D、L、または定数	オプション: 要求された PE モードまでの残り時間。 動的な値、または静的な最大値
Mode_Power_Consumption[2]	Input	DWORD	I、Q、M、D、L、または定数	現在の PE モードでの電力消費(単位 kW)。
Energy_Consumption_to_Destination[2]	Input	DWORD	I、Q、M、D、L、または定数	要求された PE モードまでの電力消費(単位 kW)。
Energy_Consumption_to_operate[2]	Input	DWORD	I、Q、M、D、L、または定数	現在の PE モードから「PE_ready_to_operate」までの電力消費(単位 kWh)。

ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。
VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」: VARIANT 設定のエラー、不正な範囲など。
[1] 時間を指定しない場合、最大値「0xFFFFFFFF」を指定します。 時間が「ゼロ」の場合、「0x00」を使用します。 [2] 電力消費量が定義しない場合、「0.0」を指定します。				

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

PE_Identify_RSP: サポートされている PROFIenergy コマンドを応答として生成

説明

補助ブロック「PE_Identify_RSP」は、コマンド PE_Identify への応答を生成します。 コマンドへの応答に、どの PROFIenergy コマンドがサポートされているかを指定します。 PE_IDENTIFY 自体が PE コマンドであり、応答内に含める必要があることにご注意ください。

パラメータ

次の表に、「PE_Identify_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
Start_Pause	Input	BOOL	I、Q、M、D、L、または定数	関連する各 PROFIenergy コマンドに対してパラメータは 1 つずつ設けられています: • 0: PE コマンドがサポートされていません • 1: PE コマンドがサポートされています
End_Pause	Input	BOOL	I、Q、M、D、L、または定数	
Query_Modes	Input	BOOL	I、Q、M、D、L、または定数	
PEM_Status	Input	BOOL	I、Q、M、D、L、または定数	
PE_Identify	Input	BOOL	I、Q、M、D、L、または定数	
Query_Measure-ment	Input	BOOL	I、Q、M、D、L、または定数	
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。

VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

PE_Measurement_List_RSP: サポートされている測定値のリストを応答として生成

説明

補助ブロック「PE_Measurement_List_RSP」は、コマンド Get_measurement_list への応答を生成します。 どの測定値(Measurement-IDs)がサポートされているかを応答内で指定します。

パラメータ

次の表に、「PE_Measurement_List_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
CMD_MODIFIER	Input	INT	I、Q、M、D、L、または定数	PROFIenergy サブコマンド(CMD=3、または CMD=16 の場合にのみ評価)。このパラメータは、「PE_I_DEV」命令の CMD_MODIFIER 出力パラメータと相互接続されている必要があります。
Count	Input	BYTE	I、Q、M、D、L、または定数	サポートされている測定値の数(測定 ID)
Measurement_List	Input	VARIANT	D	サポートされている Measurement_ID 配列へのポインタ。 PROFIenergy プロファイルに準拠した配列の構造に関する情報については、PI コマンド「Query_Measurement」-「Get_Measurement_list」を参照してください。
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。

VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

PE_Measurement_Value_RSP: 照会された測定値を応答として生成

説明

補助ブロック「PE_Measurement_Value_RSP」は、コマンド Get_measurement_values への応答を生成します。 要求された測定値を応答で返します。

パラメータ

次の表に、「PE_Measurement_Value_RSP」補助ブロックのパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PE_I_DEV_NEW	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、「PE_I_DEV」命令の NEW 出力パラメータと相互接続されている必要があります。パラメータ値に「1」がセットされている場合のみ、補助ブロックが処理されます。
CMD	Input	INT	I、Q、M、D、L、または定数	PROFIenergy コマンドの Service-Request-ID このパラメータは、「PE_I_DEV」命令の CMD 出力パラメータと相互接続されている必要があります。
CMD_MODIFIER	Input	INT	I、Q、M、D、L、または定数	PROFIenergy サブコマンド(CMD=3、または CMD=16 の場合にのみ評価)。このパラメータは、「PE_I_DEV」命令の CMD_MODIFIER 出力パラメータと相互接続されている必要があります。
Count	Input	BYTE	I、Q、M、D、L、または定数	測定値の数(Measurement_Values)
Measurement_Values	Input	VARIANT	D	測定値(Measurement_IDs)の配列へのポインタ。 PROFIenergy プロファイルに準拠した配列の構造に関する情報については、PI コマンド「Query_Measurement」 - 「Get_Measurement_values」を参照してください。
ACTIVATE	InOut	BOOL	I、Q、M、D、L	この命令は、入力パラメータを入力 ACTIVATE の立ち上がりエッジで DATA_ERROR_RSP データ領域にコピーします。その後、パラメータはこの命令によってリセットされます。 このパラメータは、「PE_I_DEV」命令のパラメータ NEW で立ち上がりエッジが検出された後、10 秒以内にセットされる必要があります。

VALID	InOut	BOOL	I、Q、M、D、L	このパラメータは、「PE_I_DEV」命令の VALID 入力と相互接続されている必要があります。 この補助ブロックは、PROFIenergy コントローラの応答データが使用可能になり、転送ができるようになるとすぐにパラメータをセットします。
DATA_ERRORRSP	InOut	VARIANT	D	応答データが格納されているデータ領域上のポインタです。 このパラメータは、「PE_I_DEV」命令の DATA_ERRORRSP のポインタと同一です。アドレス指定されたデータ領域は、完全な PROFIenergy フレームを含んでいます。 最小長:244 バイト
ERROR	Output	BOOL	I、Q、M、D、L	• "0": エラーは発生していません。 • "1": エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	• "0": エラーは発生していません。 • 「0x80B1」:VARIANT 設定のエラー、不正な範囲など。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

モジュールパラメータ割り当て

この章には下記に関する情報が記載されています：

- データレコードの書き込みと読み出し (S7-1500)
- RD_DPAR: モジュールデータレコードの読み出し (S7-1500)
- RD_DPARA: モジュールデータレコードの非同期読み出し (S7-1500)
- RD_DPARM: 構成されたシステムデータからデータレコード読み出し (S7-1500)
- WR_DPARM:データレコードの転送 (S7-1500)

データレコードの書き込みと読み出し

原理

使用しているプログラムがデータレコードを転送することができる書き込み専用システムデータ領域を持つモジュールがあります。 この領域には、0 から最大 240 の番号のデータレコードが含まれます。すべてのモジュールがすべてのデータレコードを含むとは限りません(次の表を参照)。

また、使用しているプログラムがデータレコードを読み出すことができる読み出し専用システムデータ領域を持つモジュールもあります。 この領域には、0 から最大 240 の番号のデータレコードが含まれます。すべてのモジュールがすべてのデータレコードを含むとは限りません(次の表を参照)。

注記
両方のシステムデータ領域を持つモジュールもあります。 これらは物理的に別の領域で、これらに共通しているのは論理構造のみです。

書き込み専用システムデータ領域

次の表に、書き込み専用システムデータ領域の構造を示します。 この表は、個々のデータレコードの許容サイズと、どの命令を使用して書き込みを行うかも示します。

データレコード番号	内容	サイズ	書き込みに使用できる命令
0	パラメータ	-	WR_DPARM
1	パラメータ	-	WR_DPARM
2 ~ 127	ユーザーデータ	各≤ 240 バイト	WR_DPARM WR_REC
128 ~ 240	パラメータ	各≤ 240 バイト	WR_DPARM WR_REC

読み出し専用システムデータ領域

次の表に、読み出し専用システムデータ領域の構造を示します。 この表は、個々のデータレコードの許容サイズと、どの命令を使用して読み出しを行うかも示します。

データレコード番号	内容	サイズ	読み出しに使用できる命令
0	モジュール固有の診断データ(システム全体に標準として設定)	4 バイト	RD_REC
1	チャンネル固有の診断データ (データレコード 0 を含む)	4 ~ 220 バイト	RD_REC

2 ~ 127	ユーザーデータ	各≤ 240 バイト	RD_REC
128 ~ 240	診断データ	各≤ 240 バイト	RD_REC

システムリソース

短い間隔で次々と複数の非同期データレコード転送を開始した場合、オペレーティングシステムのシステムリソースの割り当てによって、確実にすべてのジョブが実行され、相互に干渉しません。

システムリソースの限界に達した場合、これは RET_VAL で示されます。 この一時的なエラーの状況は、ジョブを繰り返すだけで解消することができます。

「同時に」有効な単一命令タイプのジョブの最大数は、CPU によって異なります。

RD_DPAR: モジュールデータレコードの読み出し

説明

この命令を使用して、設定したシステムデータからアドレス指定されたコンポーネントの番号 INDEX を持つデータレコードを読み出します。 これは、基本ラックのモジュール、または分散型コンポーネント(PROFIBUS DP または PROFINET IO)などです。

出力パラメータ VALID の値 TRUE は、データレコードがターゲット範囲 RECORD に問題なく転送されたことを示します。 この場合、LEN 出力パラメータには、読み出したデータレコードの実際の長さがバイトで含まれます。

データレコードの転送中にエラーが発生した場合、これは出力パラメータ ERROR で示されます。 この場合、出力パラメータ STATUS にはエラー情報が含まれます。

ファンクションの説明

「RD_DPAR」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 呼び出しデータレコードの転送は、REQ = 1 で「RD_DPAR」を呼び出して開始します。

出力パラメータ BUSY および出力パラメータ STATUS のバイト 2 と 3 は、ジョブのステータスを示します。 STATUS のバイト 2 と 3 は、非同期で動作する命令の出力パラメータ RET_VAL と一致します。

関連項目 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

データレコードの転送は、出力パラメータ BUSY が値 FALSE を持つときに完了します。

パラメータ

次の表に、「RD_DPAR」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: 読み出し要求
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	CPU またはインターフェースの識別番号。 番号は自動的に割り当てられ、CPU のプロパティまたはハードウェアコンフィグレーションのインターフェースに保存されます。
INDEX	Input	INT	I、Q、M、D、L、または定数	データレコード番号
RECORD	InOut	VARIANT	I、Q、M、D、L	読み出されるデータレコードのターゲット範囲です。
VALID	Output	BOOL	I、Q、M、D、L	新規データレコードが受信され、これが有効
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: ジョブがまだ完了していません。
ERROR	Output	BOOL	I、Q、M、D、L	ERROR = 1: 読み出しプロセス中にエラーが発生しました。

STATUS	Output	DWORD	I、Q、M、D、L	• 呼び出し ID (バイト 2 および 3)またはエラーコード • バイト 1: B#16#00、エラーがない場合。 それ以外の場合は、DPV1-PDU からのファンクション ID: 読み出しのデータレコードのエラーの場合には B#16#DE、データレコード書き込みのエラーの場合には B#16#DF。 DPV1 プロトコルエレメントが使用されていない場合: B#16#C0。 • バイト 4: 製造元固有のエラー ID 拡張
LEN	Output	INT	I、Q、M、D、L	読み出されたデータレコード情報の長さ

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明	制約
0000	エラーは発生していません。	-
7000	REQ=0 による最初の呼び出し:有効なデータ転送がありません。BUSY の値は「0」です。	-
7001	REQ=1 による最初の呼び出し:データ転送がトリガされました。BUSY の値は 1 です。	リモート I/O
7002	中間呼び出し(REQ は対象外):データ転送が既に有効です。BUSY の値は「1」です。	リモート I/O
8090	LADDR パラメータで指定されたアドレスが無効です。	-
8092	配列のビット列または整数以外のデータタイプが RECORD パラメータで指定されています。	-
8093	この命令は、LADDR で選択されたモジュールに対しては無効です。	-
80A1	データレコードをモジュールに送信中の否定確認(モジュールに異常が発生したか、送信中に接続が解除されました)。	-
80A2	レイヤー 2 での DP プロトコルエラー、DP スレーブでハードウェア/インターフェースにエラーが発生した可能性があります。	リモート I/O
80A3	ユーザーインターフェース/ユーザーでの DP プロトコルエラー。	リモート I/O
80A4	通信バスでの通信問題	CPU と外部 DP インターフェースモジュール間でエラーが発生しました。
80B0	モジュールタイプで命令が不可能、またはモジュールがデータレコードを認識しません。	-
80B1	送信するデータレコードの長さが不正です。	-

80B2	設定されたスロットが 割り当てられていません。	-
80B3	実際のモジュールタイプが指定されたモジュールタイプに対応していません。	-
80C1	モジュールで、同じデータレコードの前の書き込みジョブのデータがモジュールによって処理されていません。	-
80C2	モジュールは、現在 CPU で可能な最大数のジョブを処理しています。	-
80C3	必要なリソース(メモリなど)が現在占有されています。	
80C4	内部の一時的なエラー。 ジョブを実行できませんでした。 ジョブを繰り返します。 このエラーが頻発する場合は、据付けを点検して電気的な干渉の発生源を探します。	-
80C5	リモート I/O が使用不可であるか、非アクティブ。	リモート I/O
80C6	優先度クラス中止(再起動またはバックグラウンド)のため、データレコードの転送がキャンセルされました。	リモート I/O
80D0	このモジュールに対するエントリがありません。	-
80D1	データレコード番号がこのモジュールに対して設定されていません(データレコード番号 > 241 は拒否されます。)。	-
80D2	モジュールには、そのタイプ識別子のパラメータを割り当てられません。	-
80D5	データレコードが静的です。	-
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得	-
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

RD_DPARA: モジュールデータレコードの非同期読み出し 

説明

この命令を使用して、設定したシステムデータから選択されたモジュールの番号 RECNUM を持つデータレコードを読み出します。 読み出されたデータレコードは、パラメータ RECORD で定義されたターゲット範囲に格納されます。

ファンクションの説明

「RD_DPARA」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。REQ = 1 で命令を呼び出して、読み出しプロセスを開始します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目 **非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味**.

パラメータ

次の表に、「RD_DPARA」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: 読み出し要求
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	CPU またはインターフェースの識別番号。 番号は自動的に割り当てられ、CPU のプロパティまたはハードウェアコンフィグレーションのインターフェースに保存されます。
RECNUM	Input	BYTE	I、Q、M、D、L、または定数	データレコード番号(許可される値: 0～240)
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。 転送中にエラーが発生しなかった場合、以下のケースが区別されます。 • ターゲット範囲が読み出されたデータレコードよりも大きい場合、RET_VAL には実際に読み出したデータレコードの長さがバイトで含まれます。 • 読み出されたデータレコードの長さがターゲット範囲の長さと等しい場合、RET_VAL に「0」が含まれます。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: ジョブがまだ完了していません。

RECORD	Output	VARIANT	I、Q、M、D、L	読み出されるデータレコードのターゲット範囲です。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明	制約
0000	エラーは発生していません。	-
7000	REQ=0 による最初の呼び出し:有効なデータ転送がありません。BUSY の値は 0 です。	-
7001	REQ=1 による最初の呼び出し:データ転送がトリガされました。BUSY の値は 1 です。	リモート I/O
7002	中間呼び出し(REQ は対象外):データ転送が既に有効です。BUSY の値は「1」です。	リモート I/O
8090	指定された論理ベースアドレスが無効です。	-
8092	配列のビット列または整数以外のデータタイプが RECORD パラメータで指定されています。	-
8093	この命令は、LADDR で選択されたモジュールに対しては無効です。	-
80A1	データレコードをモジュールに送信中の否定確認(モジュールに異常が発生したか、送信中に接続が解除されました)。	-
80A2	レイヤー 2 での DP プロトコルエラー、DP スレーブでハードウェア/インターフェースにエラーが発生した可能性があります。	リモート I/O
80A3	ユーザーインターフェース/ユーザーでの DP プロトコルエラー。	リモート I/O
80A4	PBUS+の通信が途切れました	-
80B0	モジュールタイプで命令が不可能、またはモジュールがデータレコードを認識しません。	-
80B1	送信するデータレコードの長さが不正です。 RD_DPARM の場合: RECORD によって指定されたターゲット範囲が短すぎます。	-
80B2	設定されたスロットが 割り当てられていません。	-
80B3	実際のモジュールタイプが指定されたモジュールタイプに対応していません。	-
80C1	モジュールで、同じデータレコードの前の書き込みジョブのデータがモジュールによって処理されていません。	-
80C2	モジュールは、現在 CPU で可能な最大数のジョブを処理しています。	-
80C3	必要なリソース(メモリなど)が現在占有されています。	
80C4	内部の一時的なエラー。 ジョブを実行できませんでした。	-

	ジョブを繰り返します。このエラーが頻発する場合は、据付けを点検して電気的な干渉の発生源を探します。	
80C5	リモート I/O が使用不可であるか、非アクティブ。	リモート I/O
80C6	優先度クラス中止(再起動またはバックグラウンド)のため、データレコードの転送がキャンセルされました。	リモート I/O
80D0	このモジュールに対するエントリがありません。	-
80D1	データレコード番号がこのモジュールに対して設定されていません(データレコード番号 > 241 は拒否されます。)。	-
80D2	モジュールには、そのタイプ識別子のパラメータを割り当てられません。	-
80D5	データレコードが静的です。	-
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得	-
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

RD_DPARM: 構成されたシステムデータからデータレコード読み出し

説明

この命令を使用して、設定したシステムデータからアドレス指定されたモジュールの番号 RECNUM を持つデータレコードを読み出します。 読み出されたデータレコードは、RECORD パラメータで指定されたターゲット範囲に格納されます。

パラメータ

次の表に、「RD_DPARM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IOID	Input	BYTE	I、Q、M、D、L、または定数	アドレス領域識別子: • B#16#54 = 周辺機器入力(PI) • B#16#55 = 周辺機器出力(PQ) モジュールがハイブリッドモジュールである場合、下位アドレスの領域識別子を指定する必要があります。 アドレスが同一の場合は、B#16#54 を指定します。
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	CPU またはインターフェースのハードウェア ID。 番号は自動的に割り当てられ、ハードウェアコンフィグレーションの CPU またはインターフェースのプロパティに格納されます。
RECNUM	Input	BYTE	I、Q、M、D、L、または定数	データレコード番号(許可される値: 0～240)
RET_VAL	Return	INT	I、Q、M、D、L	読み出されたデータがターゲット範囲の領域に収まり、転送中にエラーが発生しなかった場合、読み出されたデータレコードのバイトでの長さ。 命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
RECORD	Output	VARIANT	I、Q、M、D、L	読み出されるデータレコードのターゲット範囲です。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明	制約
0000	エラーは発生していません。	-
7000	REQ=0 による最初の呼び出し:有効なデータ転送がありません。BUSY の値は「0」です。	-
7001	REQ=1 による最初の呼び出し:データ転送がトリガされました。BUSY の値は 1 です。	リモート I/O
7002	中間呼び出し(REQ は対象外): データ転送が既に有効です。BUSY の値は「1」です。	リモート I/O
8090	指定された論理ベースアドレスが無効です。	-
8092	BYTE 以外のタイプが RECORD パラメータで VARIANT 参照で指定されています。	-
8093	この命令は、LADDR で選択されたモジュールに対しては無効です。	-
80B1	RECORD によって指定されたターゲット範囲が短すぎます。	-
80C3	必要なリソース(メモリなど)が現在占有されています。	-
80D0	このモジュールに対するエントリがありません。	-
80D1	データレコード番号がこのモジュールに対して設定されていません(データレコード番号 > 241 は拒否されます。)。	-
80D2	モジュールには、そのタイプ識別子のパラメータを割り当てられません。	-
80D5	データレコードが静的です。	-
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得	-
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

WR_DPARM:データレコードの転送

説明

「WR_DPARM」命令を使用し、番号 RECNUM のデータレコードを構成データからアドレス指定されたモジュールに転送します。 この命令では、データレコードが静的であるか、動的であるかは関係ありません。

パラメータ

次の表に、「WR_DPARM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	REQ = 1: 書き込み要求
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	CPU またはインターフェースのハードウェア ID 番号は自動的に割り当てられ、CPU のプロパティまたはハードウェアコンフィグレーションのインターフェースに保存されます。
RECNUM	Input	BYTE	I、Q、M、D、L、または定数	データレコード番号
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: 書き込みプロセスが未完了です。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明	制約
0000	エラーは発生していません。	-
7000	REQ=0 による最初の呼び出し:有効なデータ転送がありません。BUSY の値は 0 です。	-
7001	REQ=1 による最初の呼び出し:データ転送がトリガされました。BUSY の値は 1 です。	リモート I/O
7002	中間呼び出し(REQ は対象外):データ転送が既に有効です。BUSY の値は「1」です。	リモート I/O
8090	LADDR パラメータで指定されたアドレスが無効です。	-
8093	この命令は、LADDR で選択されたモジュールに対しては無効です。	-

80A1	データレコードをモジュールに送信中の否定確認(モジュールに異常が発生したか、送信中に接続が解除されました)。	-
80A2	レイヤー 2 での DP プロトコルエラー、DP スレーブでハードウェア/インターフェースにエラーが発生した可能性があります。	リモート I/O
80A3	ユーザーインターフェース/ユーザーでの DP プロトコルエラー。	リモート I/O
80A4	PBUS+の通信が途切れました	-
80B0	モジュールタイプで命令が不可能、またはモジュールがデータレコードを認識しません。	-
80B1	送信するデータレコードの長さが不正です。	-
80B2	設定されたスロットが 割り当てられていません。	-
80B3	実際のモジュールタイプが指定されたモジュールタイプに対応していません。	-
80C1	モジュールで、同じデータレコードの前の書き込みジョブのデータがモジュールによって処理されていません。	-
80C2	モジュールは、現在 CPU で可能な最大数のジョブを処理しています。	-
80C3	必要なリソース(メモリなど)が現在占有されています。	
80C4	内部の一時的なエラー。 ジョブを実行できませんでした。 ジョブを繰り返します。このエラーが頻発する場合は、据付けを点検して電気的な干渉の発生源を探します。	-
80C5	リモート I/O が使用不可であるか、非アクティブ。	リモート I/O
80C6	優先度クラス中止(再起動またはバックグラウンド)のため、データレコードの転送がキャンセルされました。	リモート I/O
80D0	このモジュールに対するエントリがありません。	-
80D1	データレコード番号がこのモジュールに対して設定されていません(データレコード番号 > 241 は拒否されます。)。	-
80D2	モジュールには、そのタイプ識別子のパラメータを割り当てられません。	-
80D5	データレコードが静的です。	-
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得	-
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

割り込み

この章には下記に関する情報が記載されています：

- ATTACH: 割り込みイベントに OB を追加 (S7-1200, S7-1500)
- DETACH: 割り込みイベントから OB を削除 (S7-1200, S7-1500)
- 周期割り込み (S7-1200, S7-1500)
- 時刻割り込み (S7-1200, S7-1500)
- 遅延割り込み (S7-1200, S7-1500)
- 同期エラー (S7-1500)
- 非同期エラーイベント (S7-1200, S7-1500)

ATTACH: 割り込みイベントに OB を追加

説明

「ATTACH」命令を使用し、オーガニゼーションブロック(OB)をハードウェア割り込みイベントに割り当てます。

- OB_NR パラメータにオーガニゼーションブロックのシンボリック名または数値名を入力します。次に、これが EVENT パラメータで指定されたイベントに割り当てられます。
- EVENT パラメータでハードウェア割り込みイベントを選択します。 既に生成されたハードウェア割り込みイベントは、「システム定数」の PLC タグに記載されています。

「ATTACH」命令がエラーなしで実行された後 EVENT パラメータのイベントが発生した場合は、OB_NR パラメータのオーガニゼーションブロックが呼び出され、そのプログラムが実行されます。

ADD パラメータを使用し、オーガニゼーションブロックの他のイベントへの前の割り当てをキャンセルするか、または保持するか指定します。 ADD パラメータの値が「0」の場合、既存の割り当ては現在の割り当てで置き換えられます。

ハードウェア割り込みイベント

イベントがランタイム時に応答できるほど長時間保留されていない場合、ハードウェア割り込みを使用できます。 各ハードウェア割り込みは、ハードウェア割り込み OB に割り当てることができます。これらの OB は、特定のイベントに対する応答を含んでいます。

ハードウェア割り込みは異なるイベントで作成できます。 これの例を次に示します。

- デジタル入力の立ち上がりまたは立ち下がりエッジの検出。
- アナログ入力の定義された下限値および上限値の違反。
- 高速カウンタの外部リセット、オーバーフロー/アンダーフロー、方向反転など。

ファンクションの原理

各ハードウェア割り込みはハードウェア割り込み OB に割り当てることができ、ハードウェア割り込み OB はハードウェア割り込みイベントの発生時に順番に処理するためにキューに置かれます。 OB とイベントの割り当ては、コンフィグレーション中またはランタイム時に行うことができます。

- コンフィグレーション中にイベントを OB に割り当てるには、[ハードウェア割り込み]のハードウェアコンフィグレーションでイベントに割り当てるハードウェア割り込み OB を選択します。
- ATTACH 命令を使用してランタイム中に割り当てを行います。 イベントとハードウェア割り込み OB の割り当ては、EVENT および OB_NR パラメータを使用して行います。

パラメータ

以下の表に、「ATTACH」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_ATT	I、Q、M、D、L、または定数	オーガニゼーションブロック(最大 32767 までの番号がサポートされます。)
EVENT	Input	EVENT_ATT	D、L、または定数	OB に割り当てるハードウェア割り込みイベント。

				ハードウェア割り込みイベントは、入力または高速カウンタのハードウェアデバイス構成で最初に有効にする必要があります。
ADD	Input	BOOL	I、Q、M、D、L、または定数	以前の割り当てに及ぼす影響 • ADD=0 (既定): このイベントは、以前に行われたこの OB のすべてのイベントの割り当てと置き換えられます。 • ADD=1: このイベントがこの OB の前のイベント割り当てに追加されます。
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#....)	**説明**
0	エラーは発生していません。
8090	OB が存在しません。
8091	OB のタイプが正しくありません。
8093	イベントが存在しません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

DETACH: 割り込みイベントから OB を削除

説明

この命令を使用して、ランタイム中にオーガニゼーションブロックの 1 つ以上のハードウェア割り込みイベントへの既存の割り当てをキャンセルします。

OB_NR パラメータにオーガニゼーションブロックのシンボリック名または数値名を入力します。 この場合、EVENT パラメータで指定されたこのオーガニゼーションブロックのイベントへの割り当ては、キャンセルされます。

- 個々のハードウェア割り込みイベントを EVENT パラメータで選択した場合、このハードウェア割り込みイベントへの OB の割り当てはキャンセルされます。 その他の現存の割り当てはすべて、アクティブなままとなります。 オペランドプレースホルダのドロップダウンリストを使用して、個々のハードウェア割り込みイベントを選択できます。
- ハードウェア割り込みイベントを選択しなかった場合、この OB_NR オーガニゼーションブロックに現在割り当てられているすべてのイベントは分離されます。

パラメータ

以下の表に、「DETACH」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_ATT	I、Q、M、D、L、または定数	オーガニゼーションブロック(最大 32767 までの番号がサポートされます。)
EVENT	Input	EVENT_ATT	D、L、または定数	ハードウェア割り込みイベント
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL parameter

エラーコード * (W#16#....)	説明
0	エラーは発生していません。
1	割り当てが存在しません(警告)
8090	OB が存在しません。
8091	OB のタイプが正しくありません。
8093	イベントが存在しません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

周期割り込み

この章には下記に関する情報が記載されています：

- SET_CINT: 周期割り込みパラメータの設定 (S7-1200, S7-1500)
- QRY_CINT: 周期割り込みパラメータの問い合わせ (S7-1200, S7-1500)

SET_CINT: 周期割り込みパラメータの設定

説明

この命令を使って、周期割り込み OB のパラメータを設定します。周期割り込み OB の開始時刻は、OB およびフェーズオフセットのそれぞれの時間間隔から生成されます。

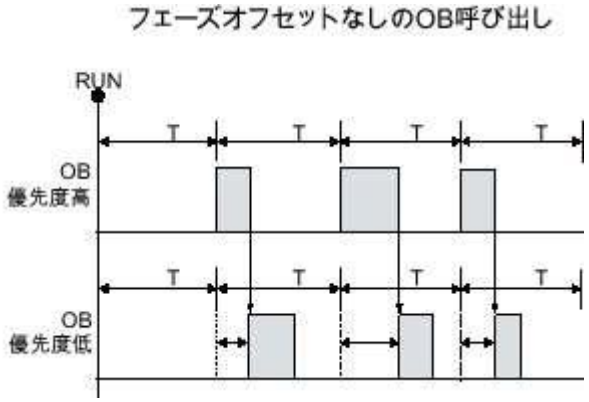
- OB の時間間隔は、OB が定期的に呼び出される間隔です。 たとえば、時間間隔が 100 µs の場合、OB はプログラムの実行中に 100 µs 毎に呼び出されます。
- フェーズオフセットは、周期割り込み OB の呼び出しがオフセットされる時間間隔です。 フェーズオフセットを使って、優先度の低いオーガニゼーションブロックの処理を正確なタイムベースで行うことができます。

OB が存在しない、または使用する時間間隔がサポートされていない場合、RET_VAL パラメータで対応するエラーアラームが出力されます。

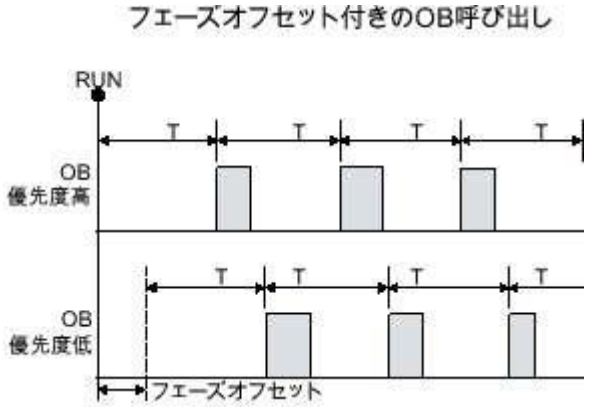
CYCLE パラメータの時間間隔が「0」の場合、OB が呼び出されないことを意味します。

ファンクションの説明

優先度が低い OB および優先度の高い OB が同じ時間間隔で呼び出される場合、優先度が低い OB は、優先度が高い OB が実行された後のみに呼び出されます。 優先度が高い OB の実行時間の長さに応じて、優先度が低い OB の呼び出し時刻をオフセットできます。



優先度が低い OB のフェーズオフセットが設定され、このフェーズオフセットが優先度が高い OB の現在の実行時間よりも長い場合、ブロックは固定されたタイムベースで呼び出されます。



パラメータ

次の表に、「SET_CINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_CY-CLIC	I、Q、M、D、L または定数	OB 番号(<32768)
CYCLE	Input	UDINT	I、Q、M、D、L または定数	時間間隔(マイクロ秒)
PHASE	Input	UDINT	I、Q、M、D、L または定数	フェーズオフセット
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#....)	説明
0	エラーは発生していません。
8090	OB が存在しない、または間違ったタイプ
8091	不正な時間間隔
8092	不正なフェーズオフセット
80B2	OB にイベントが割り当てられていない
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

QRY_CINT: 周期割り込みパラメータの問い合わせ

説明

この命令を使用して、周期割り込み OB の現在のパラメータを照会します。サイクリック割り込み OB は、OB_NR パラメータを使用して特定されます。

照会された周期割り込みパラメータの値は、「QRY_CINT」命令が実行された時点でのものです。

パラメータ

次の表に、「QRY_CINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_CYCLIC (INT)	I、Q、M、D、L、または定数	OB 番号(<32768)または OB の名前(たとえば OB_MyOB)による記号アドレス
CYCLE	Output	UDINT	I、Q、M、D、L	時間間隔(マイクロ秒)
PHASE	Output	UDINT	I、Q、M、D、L	フェーズオフセット
STATUS	Output	WORD	I、Q、M、D、L	周期割り込みのステータス: • ビット 0～ビット 4: STATUS パラメータを参照 • その他のビット: 常時「0」
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

ビット	値	意味
0	0	未使用(常に「0」).
1	0	周期割り込みが有効。
	1	周期割り込みが遅延された。
2	0	周期割り込みが有効になっていないか、期限切れになった。
	1	周期割り込みが有効。
3	0	未使用(常に「0」).
4	0	指定された番号の OB が存在しない。
	1	指定された番号の OB が存在する。
その他のビット		未使用(常に「0」).

RET_VAL パラメータ

エラーが発生した場合、RET_VAL パラメータで関連するエラーコードが表示され、STATUS パラメータは「0」にセットされます。

エラーコード* (W#16#....)	説明
0	エラーなし。
8090	OB が存在しない、または間違ったタイプ
80B2	OB に結果が割り当てられていない
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

時刻割り込み

この章には下記に関する情報が記載されています：

- SET_TINT: 時刻割り込みの設定 (S7-1500)
- SET_TINTL: 時刻割り込みの設定 (S7-1200, S7-1500)
- CAN_TINT: 時刻割り込みのキャンセル (S7-1200, S7-1500)
- ACT_TINT: 時刻割り込み有効化 (S7-1200, S7-1500)
- QRY_TINT: 時刻割り込みのステータス問い合わせ (S7-1200, S7-1500)

SET_TINT: 時刻割り込みの設定

説明

「SET_TINT」命令を使用して、ハードウェアコンフィグレーションで設定を行わずに、ユーザープログラムから時刻割り込みオーガニゼーションブロックの開始値および時間を設定します。

- パラメータ OB_NR で、開始日付および時刻を設定したい時刻割り込み OB の番号を入力します。
- パラメータ SDT および PERIOD を使用して、時刻割り込み OB をいつ、どれくらいの頻度で呼び出すかを指定します。
  - 1 回限りの呼び出し: パラメータ SDT で日付と時刻を入力し、パラメータ PERIOD で値「0」を使用します。
  - 繰り返し呼び出し: パラメータ SDT で最初の呼び出しの日付と時刻を入力します。 パラメータ PERIOD を使用して、OB の後続の呼び出しが行われる時間間隔を定義します。

開始日付および開始時刻を設定するときは、以下の事項を遵守してください。

- パラメータ SDT での日付と時刻の指定は、システム時刻を参照します。
- 開始時刻での秒およびミリ秒の指定は無視され、「0」に設定されます。
- 毎月の時刻割り込み OB を指定する場合は、開始日付 1、2、... 28 日のみが可能です。 この制限により、(たとえば、1 か月の日数が 30 日の月や 2 月に)毎月の呼び出しをスキップするのを防ぎます。

  月の 29 日、30 日および 31 日の代替として、パラメータ PERIOD で「月末」(W#16#2001)設定を使用できます。

「SET_TINT」で時刻割り込みを設定した後、まだ命令「ACT_TINT」でこの時刻割り込みを有効にする必要があります。

注記

時刻割り込み OB に関する追加情報

時刻割り込み OB の使用に関するその他の特殊機能については、各 CPU のオーガニゼーションブロックの説明を参照してください。

S7-1200 の場合: **時刻割り込みのオーガニゼーションブロック**

S7-1500 の場合: **時刻割り込み OB**

パラメータ SDT および PERIOD での設定は、時刻割り込み OB のプロパティでの時刻割り込みの設定と一致しています。

パラメータ

次の表に、「SET_TINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_TOD	I、Q、M、D、L、または定数	時刻割り込み OB の番号 • 時刻割り込み OB には、10～17 の番号が使用可能です。 • 123 で始まる OB 番号を代替として割り当てることもできます。

				OB 番号はプログラムブロックフォルダおよびシステム定数に表示されます。
SDT	Input	DT	D、L、または定数	開始日付および開始時刻
PERIOD	Input	WORD	I、Q、M、D、L、または定数	開始点 SDT から以下までの実行間隔 • W#16#0000 = 単一の実行 • W#16#0201 = 毎分 1 回 • W#16#0401 = 毎時間 1 回 • W#16#1001 = 毎日 1 回 • W#16#1201 = 毎週 1 回 • W#16#1401 = 毎月 1 回 • W#16#1801 = 毎年 1 回 • W#16#2001 = 月末
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、RET_VAL の実パラメータにエラーコードが含まれます。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	パラメータ OB_NR でエラーが発生しました(時刻割り込み OB がアドレス指定されていません)。
8091	パラメータ SDT でエラーが発生しました(日付と時刻の指定が無効です)。
8092	パラメータ PERIOD での入力が不正です。
80A1	設定した開始時刻が過去です。 このエラーコードは、PERIOD = W#16#0000 時にのみ発生します。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

SET_TINTL: 時刻割り込みの設定

説明

「SET_TINTL」命令を使用して、ハードウェアコンフィグレーションで設定を行わずに、ユーザープログラムから時刻割り込みオーガニゼーションブロックの開始値および時間を設定します。

- パラメータ OB_NR で、開始日付および時刻を設定したい時刻割り込み OB の番号を入力します。
- パラメータ SDT および PERIOD を使用して、時刻割り込み OB をいつ、どれくらいの頻度で呼び出すかを指定します。
  - 1 回限りの呼び出し: パラメータ SDT で日付と時刻を入力し、パラメータ PERIOD で値「0」を使用します。
  - 繰り返し呼び出し: パラメータ SDT で最初の呼び出しの日付と時刻を入力します。 パラメータ PERIOD を使用して、OB の後続の呼び出しが行われる時間間隔を定義します。
- パラメータ LOCAL を使用して、パラメータ SDT で指定された時刻がローカルタイムまたはシステム時刻のいずれを参照するかを選択します。
- ACTIVATE パラメータを使用し、オーガニゼーションブロックの設定が直接(ACTIVATE = true)または時刻割り込みオーガニゼーションブロックの「ACT_TINT」が呼び出された後のみ(ACTIVATE = false)に適用されるかを指定します。

開始日付および開始時刻を設定するときは、以下の事項を遵守してください。

- 開始時刻での秒およびミリ秒の指定は無視され、「0」に設定されます。
- 毎月の時刻割り込み OB を指定する場合は、開始日付 1、2、... 28 日のみが可能です。 この制限により、(たとえば、1 か月の日数が 30 日の月や 2 月に)毎月の呼び出しをスキップするのを防ぎます。

  月の 29 日、30 日および 31 日の代替として、パラメータ PERIOD で「月末」(W#16#2001)設定を使用できます。

ローカルタイムの使用中は、以下の事項を遵守してください。

- サマータイムから標準時間への切り替え:サマータイムから標準時間への切り替え中の 2 時間以内に、開始時刻で時刻割り込みオーガニゼーションブロックを呼び出す場合、時間切り替えの 1 時間以内にも遅延割り込みを使用してください。
- 標準時間からサマータイムへの切り替え: サマータイム切り替え日について時刻としてスキップされた 1 時間を指定した場合、単一実行(PERIOD = W#16#0000)時にエラーコード 16#8091 が出力されます。

注記

時刻割り込み OB に関する追加情報

時刻割り込み OB の使用に関するその他の特殊機能については、各 CPU のオーガニゼーションブロックの説明を参照してください。

S7-1200 の場合: **時刻割り込みのオーガニゼーションブロック**

S7-1500 の場合: **時刻割り込み OB**

パラメータ SDT および PERIOD での設定は、時刻割り込み OB のプロパティでの時刻割り込みの設定と一致しています。

パラメータ

次の表に、「SET_TINTL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_TOD	I、Q、M、D、L、または定数	時刻割り込み OB の番号 • 時刻割り込み OB には、10～17 の番号が使用可能です。 • 123 で始まる OB 番号を代替として割り当てることもできます。 OB 番号はプログラムブロックフォルダおよびシステム定数に表示されます。
SDT	Input	DTL	D、L、または定数	開始日付および開始時刻
LOCAL	Input	BOOL	I、Q、M、D、L、または定数	• true:ローカルタイムを使用 • false:システム時刻を使用
PERIOD	Input	WORD	I、Q、M、D、L、または定数	開始点 SDT から以下までの実行間隔 • W#16#0000 = 単一の実行 • W#16#0201 = 毎分 1 回 • W#16#0401 = 毎時間 1 回 • W#16#1001 = 毎日 1 回 • W#16#1201 = 毎週 1 回 • W#16#1401 = 毎月 1 回 • W#16#1801 = 毎年 1 回 • W#16#2001 = 月末
ACTIVATE	Input	BOOL	I、Q、M、D、L、または定数	• true: 時刻割り込みを設定し、有効にする • false: 時刻割り込みを設定し、「ACT_TINT」の呼び出し時にのみ有効にする
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、RET_VAL の実パラメータにエラーコードが含まれます。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	パラメータ OB_NR でエラーが発生しました(時刻割り込み OB がアドレス指定されていません)。
8091	パラメータ SDT でエラーが発生しました(日付と時刻の指定が無効です)。
8092	パラメータ PERIOD での入力が不正です。

80A1	設定した開始時刻が過去です。 このエラーコードは、PERIOD = W#16#0000 時にのみ発生します。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

CAN_TINT: 時刻割り込みのキャンセル

説明

「CAN_TINT」命令を使用して、指定した時刻割り込みオーガニゼーションブロックの開始値および開始時刻を削除します。 これによって、時刻割り込みを無効にすると、オーガニゼーションブロックはそれ以上呼び出されません。

この時刻割り込みを再使用する場合は、開始時刻をリセットしなければなりません(命令「SET_TINTL」または「SET_TINT」)。

この後、この時刻割り込みを再び有効にする必要があります。

- ACTIVE=false パラメータで命令「SET_TINT」または「SET_TINTL」を使用して時刻割り込みをセットする場合は、「ACT_TINT」を呼び出します。
- ACTIVE=true パラメータで、命令「SET_TINTL」を使用して、時刻割り込みを直接に再び有効にすることもできます。

パラメータ

次の表に、「CAN_TINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_TOD	I、Q、M、D、L、または定数	その開始日付および時刻を削除する、時刻割り込み OB の番号。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、RET_VAL の実際のパラメータにエラーコードが含まれます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	OB_NR パラメータのエラー。
80A0	関連する時刻割り込み OB の開始日付/時刻が定義されていません。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

ACT_TINT: 時刻割り込み有効化

説明

命令「ACT_TINT」を使用して、ユーザープログラムから、時刻割り込みオーガニゼーションブロックを有効にすることができます。 この命令の実行の前提条件は、時刻割り込み OB で、開始日付および時刻がセット済みであることです。

パラメータ

次の表に、「ACT_TINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_TOD	I、Q、M、D、L、または定数	時刻割り込み OB の数 • 時刻割り込み OB では、番号 10 ～ 17 を使用できます。 • 123 で始まる OB 番号も、代替番号として使用できます。 この OB 番号は、プログラムブロックフォルダおよびシステム定数に表示されます。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、RET_VAL の実際のパラメータにエラーコードが含まれます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	OB_NR パラメータのエラー(時刻割り込み OB がアドレス指定されていません)。
80A0	関連する時刻割り込み OB に開始日付および時刻が設定されていません。
80A1	有効になった時刻が過去です。 このエラーは、時刻割り込みを 1 回だけ実行する場合のみ、発生します。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

QRY_TINT: 時刻割り込みのステータス問い合わせ

説明

この命令を使用して、STATUS 出力パラメータで、時刻割り込みオーガニゼーションブロックのステータスを表示できます。

パラメータ

次の表に、「QRY_TINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_TOD	I、Q、M、D、L、または定数	そのステータスが照会される時刻割り込み OB の番号 この OB 番号は、プログラムブロックフォルダおよびシステム定数に表示されます。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、RET_VAL の実際のパラメータにエラーコードが含まれます。
STATUS	Output	WORD	I、Q、M、D、L	時刻割り込み OB のステータス(下記を参照)

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。 例外: ビット 4 のステータスメッセージ「0」(この番号の OB が作成されていません)。
8090	OB_NR パラメータのエラー。考えられる原因: • パラメータ OB_NR の値が、CPU によってサポートされている OB 番号ではありません(<1 または> 32767)。 • この OB 番号は、時刻割り込み OB のアドレスを指定していません。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

STATUS パラメータ

エラーが発生した場合(RET_VAL パラメータを参照)、STATUS パラメータで「0」が出力されます。

ビット	値	意味
0	0	RUN 中
	1	スタートアップ中。
1	0	時刻割り込みが有効になっています。
	1	時刻割り込みが無効です。
2	0	時刻割り込みが有効になっていないか、または経過しました。
	1	時刻割り込みが有効になっています。
4	0	OB_NR パラメータで OB 番号が指定された OB が存在しません。
	1	OB_NR パラメータで OB 番号が指定された OB が存在します。
6	0	この時刻割り込みはシステム時刻に基づいています
	1	この時刻割り込みはローカル時刻に基づいています
その他		常時「0」

遅延割り込み

この章には下記に関する情報が記載されています：

- 時間遅延割り込みの使用 (S7-1200, S7-1500)
- SRT_DINT: 遅延割り込み起動 (S7-1200, S7-1500)
- CAN_DINT: 遅延割り込みキャンセル (S7-1200, S7-1500)
- QRY_DINT: 遅延割り込みステータスの問い合わせ (S7-1200, S7-1500)

時間遅延割り込みの使用

定義

「SRT_DINT」命令を呼び出すと、指定された遅延時間の経過後、つまり割り当てられた遅延割り込み OB が呼び出された後に、オペレーティングシステムが割り込みを生成します。

呼び出しの前提条件

オペレーティングシステムが遅延割り込みを呼び出せるようにするには、次の条件を満たしている必要があります。

- 遅延割り込み OB は、「SRT_DINT」命令で開始する必要があります。
- 設定中に遅延割り込み OB を選択解除しないでください。
- 遅延割り込み OB は、CPU に存在していることが必要です。

命令「SRT_DINT」、「CAN_DINT」および「QRY_DINT」の目的"

この命令を使って、次を行います。

- 時間遅延割り込みの起動(「SRT_DINT」)
- 時間遅延割り込みのキャンセル(「CAN_DINT」)
- 時間遅延割り込みのクエリ(「QRY_DINT」)

遅延割り込みに対する影響

次の表に、さまざまな状況とその遅延割り込みへの影響を示します。

条件	結果	対応
遅延割り込みが開始されました(「SRT_DINT」の呼び出しによる)	遅延割り込みが既に開始されています。	遅延時間が上書きされます。遅延割り込みが再度開始されます。
	呼び出し時に遅延割り込み OB が存在しません。	オペレーティングシステムが優先度クラスエラーを生成します(OB 85 の呼び出し)。 OB 85 が存在しない場合、CPU が STOP に移行します。
	起動 OB で割り込みが開始され、CPU が RUN に移行する前に遅延時間が経過します。	遅延割り込み OB の呼び出しが、CPU が RUN モードになるまで遅延されます。
遅延時間が経過しました。	前に開始された遅延割り込み OB がまだ実行中です。	オペレーティングシステムが時刻エラーを生成します(OB 80 の呼び出し)。 OB 80 が存在しない場合、CPU が STOP に移行します。

ウォームリスタートおよびコールドリスタートへの応答

ウォームリスタートまたはコールドリスタート中に、ユーザープログラムで命令によって行われたすべての遅延割り込み設定が消去されます。

起動 OB での開始

起動 OB で遅延割り込みを開始できます。遅延 OB を呼び出すには、2 つの条件を満たす必要があります。

- 遅延時間が経過していること。
- CPU が RUN モードであること。

遅延時間が経過し、CPU がまだ RUN モードでない場合、遅延割り込み OB 呼び出しは CPU が RUN モードになるまで遅延されます。 次に、遅延割り込み OB が呼び出されてから、OB メイン[OB 1]の最初の命令が実行されます。

SRT_DINT: 遅延割り込み起動

説明

「SRT_DINT」命令を使用して、パラメータ DTIME で指定した遅延時間経過後に遅延割り込み OB を呼び出す遅延割り込みを開始します。 遅延時間は、EN イネーブル入力で立ち下がりエッジが生成されたときに開始されます。 遅延時間のカウントダウン中は、イネーブル入力 EN の信号状態は「0」である必要があります。 遅延時間のカウントダウンが中断されると、OB_NR パラメータで設定された OB は実行されません。

精度

「SRT_DINT」命令と遅延割り込み OB の開始の最大時間間隔は、呼び出しが割り込みイベントで遅延されなければ、設定された遅延時間よりも 1 ミリ秒長くなります。

パラメータ

次の表に、「SRT_DINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_DELAY (INT)	I、Q、M、D、L、または定数	遅延時間後に実行される OB の番号
DTIME	Input	TIME	I、Q、M、D、L、または定数	遅延時間(1～60000 ミリ秒) 遅延割り込み OB のカウンタの使用などによって、時間をさらに長くすることができます。
SIGN	Input	WORD	I、Q、M、D、L、または定数	遅延割り込み OB が呼び出されたときに、OB の開始イベント情報に表示される識別子。
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8090	不正なパラメータ OB_NR
8091	不正なパラメータ DTIME
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。

CAN_DINT: 遅延割り込みキャンセル

説明

この命令を使用して、開始された遅延割り込みをキャンセルします。よって、設定された遅延時間後に実行される遅延割り込み OB の呼び出しもキャンセルされます。 OB_NR パラメータで、呼び出しをキャンセルするオーガニゼーションブロックの番号を指定します。

パラメータ

次の表に、「CAN_DINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_DELAY (INT)	I、Q、M、D、L、または定数	呼び出しをキャンセルする OB の番号
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
8090	不正なパラメータ OB_NR
80A0	遅延割り込みが開始されていません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

QRY_DINT: 遅延割り込みステータスの問い合わせ

説明

「QRY_DINT」命令を使用して、遅延割り込みのステータスを照会します。

パラメータ

次の表に、「QRY_DINT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
OB_NR	Input	OB_DELAY (INT)	I、Q、M、D、L、または定数	ステータスが照会されている OB の番号。
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、RET_VAL の実際のパラメータにエラーコードが含まれます。 STATUS パラメータに値「0」が表示されます。
STATUS	Output	WORD	I、Q、M、D、L	遅延割り込みのステータス。次の表を参照してください。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ STATUS

ビット	値	意味
0	0	オペレーティングシステムが RUN
	1	オペレーティングシステムがスタートアップ
1	0	オペレーティングシステムによって遅延割り込みが有効になりました。
	1	遅延割り込みが無効になりました。
2	0	遅延割り込みが有効になっていないか、または経過しました。
	1	遅延割り込みが有効になりました。
3	-	-
4	0	指定された番号の遅延割り込み OB が存在しません。
	1	指定された番号の遅延割り込み OB が存在します。
その他のビット		常時「0」

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	OB_NR パラメータの不正な情報
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

同期エラー

この章には下記に関する情報が記載されています：

- 同期エラーイベントのマスク (S7-1500)
- MSK_FLT: 同期エラーイベントのマスク (S7-1500)
- DMSK_FLT: 非同期エラーイベントのマスク解除 (S7-1500)
- READ_ERR: イベントステータスレジスタ読み出し (S7-1500)

同期エラーイベントのマスク

概要

同期エラーは、プログラミングおよびアクセスエラーです。 これらのエラーは、不正なオペランド領域または番号、または不正なアドレスのあるプログラミングの結果、発生します。 これらの同期エラーの**マスク**とは、次を意味します。

- マスクされた同期エラーは、エラー OB 呼び出しをトリガせず、プログラムされた代替の応答につながりません。
- CPU は、マスクされた発生したエラーをエラーステータスレジスタに「記録」します。

マスクは、「MSK_FLT」命令を呼び出して実行されます。

エラーの**アンマスク**は、前に設定したマスクをキャンセルし、現在の優先度クラスのイベントステータスレジスタで対応するビットをクリアすることを意味します。 マスクは以下の場合にキャンセルされます。

- 「DMSK_FLT」命令を呼び出した場合。
- 現在の優先度クラスが完了した場合。

マスク解除された後にエラーイベントが発生すると、オペレーティングシステムは関連エラー OB を開始します。

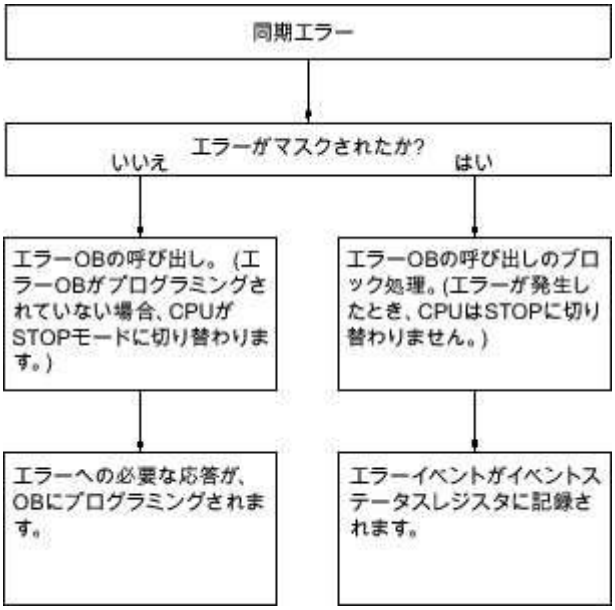
「READ_ERR」命令を使用して、発生したマスクされたエラーを読み出すことができます。

注記
S7-1500 では、エラーイベントのマスク処理またはマスク解除処理に関わりなく、エラーイベントは診断バッファに入力され、CPU のグループエラー LED が点灯します。

エラーの全般的な処理

ユーザープログラムで、プログラミングおよび I/O 領域アクセスのエラーが発生した場合、さまざまな異なる操作が可能です。

- 対応するエラーが発生したときに、オペレーティングシステムで呼び出されるエラー OB をプログラムすることができます。
- 各優先度クラスで、個々にエラー OB 呼び出しを無効にすることができます。 この場合、特定の優先度クラスでこのタイプのエラーが発生したとき、CPU は STOP に移行しません。 CPU は、そのエラーをエラーレジストリに入力します。 ただし、このエントリからは、エラーがいつ、どれほどの頻度で発生したかはわかりません。

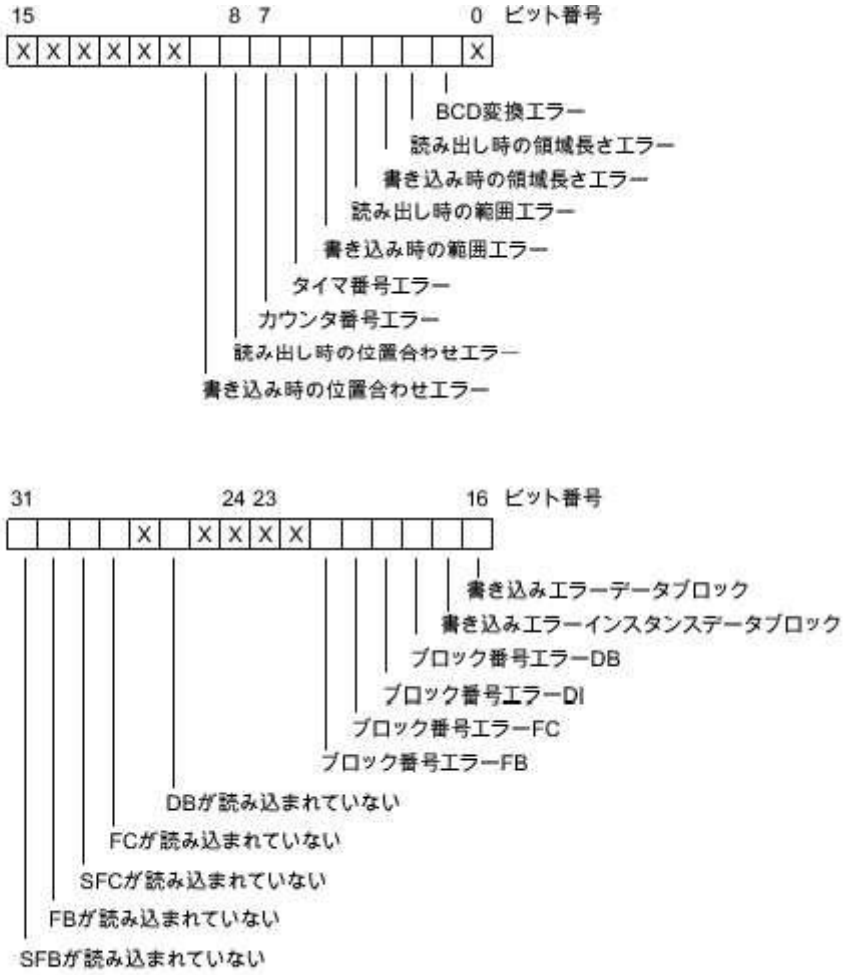


同期エラーは、**エラーフィルタ**と呼ばれる特定のビットパターンに割り当てられます。 このエラーフィルタは、「MSK_FLT」、「DMSK_FLT」、および「READ_ERR」命令の入力および出力パラメータにあります。

同期エラーは、2 つのエラーマスクでマスク処理できる**プログラミングおよびアクセスエラー**に分けられます。 次の図に、エラーフィルタを示します。

プログラミングエラーフィルタ

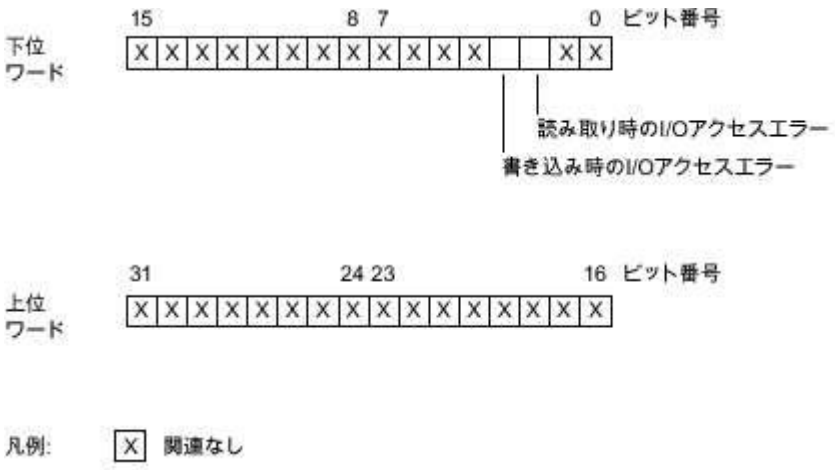
次の図に、プログラミングエラー用のエラーフィルタのビットパターンを示します。 プログラミングエラー用のエラーフィルタは、「PRGFLT_...」パラメータにあります(以下の「プログラミングエラー、下位ワード」または「プログラミングエラー、上位ワード」を参照)。



命令「MSK_FLT」、「DMSK_FLT」、「READ_ERR」の入力および出力パラメータの無関係のビット「x」は「0」にセットされます。

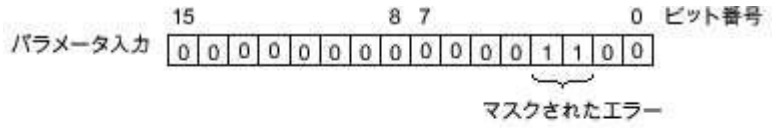
アクセスエラーマスク

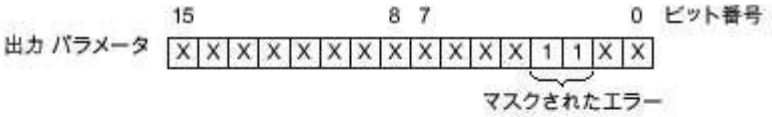
次の図に、アクセスエラー用のエラーマスクのビットパターンを示します。 アクセスエラー用のエラーマスクは、パラメータ ACCFLT_..に存在します。

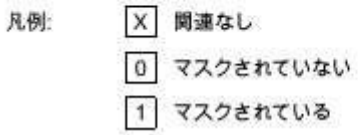


例: 次の図に、すべてのマスク処理されたエラーを備えたアクセスエラー用のエラーマスクの下位ワードを示します。

- 「MSK_FLT」の入力パラメータとして
- 「MSK_FLT」の出力パラメータとして







プログラミングエラーの下位ワード

次の表は、プログラミングエラーのエラーフィルタの下位ワードに割り当てられるエラーの一覧を示します。 また、エラーの考えられる原因も示します。

エラー	イベント ID (W#16#...)	考えられるエラーの原因
BCD 変換エラー	2521	変換する値が BCD 数ではありません(たとえば、5E8)。
読み出し時の領域の長さエラー	2522	アドレス指定したオペランドが、全体として、可能なオペランド範囲内にありません。 例: メモリ領域が 256 バイトの長さしかないにもかかわらず、MW 320 を読み出す必要があります。
書き込み時の領域の長さエラー	2523	アドレス指定したオペランドが、全体として、可能なオペランド範囲内にありません。 例: メモリ領域が 256 バイトの長さしかないにもかかわらず、MW 320 に値を書き込む必要があります。
読み出し時の範囲エラー	2524	間接的なオーバーラップする範囲のアドレス指定時に、オペランドで、不正な範囲識別子が指定されました。 例: • 正:LAR1 P#E 12.0 L W[AR1, P#0.0] • 不正:LAR1 P#12.0 L W[AR1, P#0.0] この操作時に、範囲エラーが報告されます。

書き込み時の範囲エラー	2525	間接的なオーバーラップする範囲のアドレス指定時に、オペランドで、不正な範囲識別子が指定されました。 例: • 正:LAR1 P#E 12.0 T W[AR1, P#0.0] • 不正:LAR1 P#12.0 T W[AR1, P#0.0] この操作時に、範囲エラーが報告されます。
タイマ番号エラー	2526	存在しないタイマがアクセスされます。 例: • SP T [MW 0]で MW 0 = 129 の場合、使用可能なタイマは 128 個のみですが、タイマ 129 が開始されています。
カウンタ番号エラー	2527	存在しないカウンタがアクセスされます。 • 例: ZV Z [MW 0]で MW 0 = 600 の場合、使用可能なカウンタは 512 個のみですが、カウンタ 600 がアクセスされています。
読み出し時の位置合わせエラー	2528	ビットアドレス≠ 0 で、バイト、ワード、またはダブルワードオペランドがアドレス指定されています。 例: • 正:LAR1 P#M12.0 L B[AR1, P#0.0] • 不正:LAR1 P#M12.4 L B[AR1, P#0.0]
書き込み時の位置合わせエラー	2529	ビットアドレス≠ 0 で、バイト、ワード、またはダブルワードオペランドがアドレス指定されています。 例: • 正:LAR1 P#M12.0 T B[AR1, P#0.0] • 不正:LAR1 P#M12.4 T B[AR1, P#0.0]

プログラミングエラーの上位ワード

次の表は、プログラミングエラーのエラーフィルタの上位ワードに割り当てられるエラーの一覧を示します。 また、エラーの考えられる原因も示します。

エラー	イベント ID (W#16#...)	考えられるエラーの原因
書き込みエラーデータブロック	2530	書き込み先のデータブロックが、書き込み保護されています
書き込みエラーインスタンスデータブロック	2531	書き込み先のインスタンスデータブロックが、書き込み保護されています。
ブロック番号エラー DB	2532	データブロックの番号が、最大許容番号を超えています。

ブロック番号エラー DI	2533	インスタンスデータブロックの番号が、最大許容番号を超えています。
ブロック番号エラー FC	2534	呼び出されたファンクション(FC)の番号が、最大許容番号を超えています。
ブロック番号エラー FB	2535	呼び出されたファンクションブロック(FB)の番号が、最大許容番号を超えています。
DB が読み込まれていない	253A	データブロックが読み込まれません。
命令が読み込まれていない	253C ~ 253F	呼び出す命令が読み込まれません。

アクセスエラー

次の表に、アクセスエラー用のエラーマスクに割り当てられているエラーを示します。 また、エラーの考えられる原因も示します。

エラー	イベント ID (W#16#...)	考えられるエラーの原因
I/O アクセスエラー 読み出し時	2942	• I/O 領域のアドレスに、信号モジュールが割り当てられていません。 • 指定されたモジュールモニタ時間内に、この I/O エリアへのアクセスが確認されませんでした(タイムアウト)。
I/O アクセスエラー 書き込み時	2943	

MSK_FLT: 同期エラーイベントのマスク

説明

この命令を使用して、CPU の同期エラーへの応答を制御します。 これは、それぞれの同期エラーをマスキングすることで行います(エラーフィルタについては、同期エラーイベントのマスクを参照)。「MSK_FLT」を呼び出すと、現在の優先度クラスで同期エラーをマスクします。

入力パラメータで同期エラーフィルタの個々のビットを「1」にセットすると、以前に設定された他のビットは値「1」を保持します。 出力パラメータを使用し、読み出し可能な新規エラーフィルタを取得します。 マスクした同期エラーは OB を呼び出さず、単にエラーレジスタに入力されます。 エラーレジスタは、「READ_ERR」命令で読み出すことができます。

パラメータ

次の表に、「MSK_FLT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PRGFLT_SET_MASK	Input	DWORD	I、Q、M、D、L、または定数	マスクするプログラミングエラー
ACCFLT_SET_MASK	Input	DWORD	I、Q、M、D、L、または定数	マスクするアクセスエラー
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
PRGFLT_MASKED	Output	DWORD	I、Q、M、D、L	マスクされたプログラミングエラー
ACCFLT_MASKED	Output	DWORD	I、Q、M、D、L	マスクされたアクセスエラー

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	まだどのエラーもマスクされていません。
0001	1 つ以上のエラーがマスクされています。 しかし、他のエラーはこれからマスクされます。
-	一般エラー情報 関連項目 GetErrorID でローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

DMSK_FLT: 非同期エラーイベントのマスク解除

説明

この命令を使用して、「MSK_FLT」でマスクされたエラーをアンマスクします。 これを行うには、入力パラメータでエラーフィルタの対応するビットを「1」にセットする必要があります。「DMSK_FLT」呼び出しを使って、現在の優先度クラスの対応する同期エラーをアンマスクします。同時に、エラーレジスタで照会されたエントリが消去されます。 出力パラメータを使用し、新規のエラーフィルタを読み出すことができます。

パラメータ

次の表に、「DMSK_FLT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PRGFLT_RESET_MASK	Input	DWORD	I、Q、M、D、L、または定数	アンマスクするプログラミングエラー
ACCFLT_RESET_MASK	Input	DWORD	I、Q、M、D、L、または定数	アンマスクするアクセスエラー
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
PRGFLT_MASKED	Output	DWORD	I、Q、M、D、L	マスクされた状態のプログラミングエラー
ACCFLT_MASKED	Output	DWORD	I、Q、M、D、L	マスクされた状態のアクセスエラー

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	すべての指定されたエラーがアンマスクされました。
0001	1 つ以上のエラーがマスクされていません。 しかし、他のエラーはこれからアンマスクされます。
-	一般エラー情報 関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

READ_ERR: イベントステータスレジスタ読み出し

説明

この命令を使用して、エラーレジスタを読み出します。 エラーレジスタの構造は、プログラミングおよびアクセスエラーフィルタの構造に対応し、「MSK_FLT」および「DMSK_FLT」で入力パラメータとしてプログラムすることができます。

入力パラメータに、エラーレジスタから読み出す同期エラーを入力します。 「READ_ERR」を呼び出すと、必要なエントリをエラーレジスタから読み出し、同時にこれらのエントリを消去します。

エラーレジスタには、現在の優先度クラスでどのマスクされた同期エラーが 1 回以上発生したかを示す情報が含まれます。 ビットが設定されると、対応するマスクされた同期エラーが 1 回以上発生したことを意味します。

パラメータ

次の表に、「READ_ERR」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PRGFLT_QUERY	Input	DWORD	I、Q、M、D、L、または定数	プログラミングエラーの照会
ACCFLT_QUERY	Input	DWORD	I、Q、M、D、L、または定数	アクセスエラーの照会
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
PRGFLT_CLR	Output	DWORD	I、Q、M、D、L	発生したプログラミングエラー
ACCFLT_CLR	Output	DWORD	I、Q、M、D、L	発生したアクセスエラー

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	すべての照会されたエラーがマスクされています。
0001	1 つ以上の照会されたエラーがマスクされていません。
-	一般エラー情報 関連項目 GetErrorID でローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

非同期エラーイベント

この章には下記に関する情報が記載されています：

- DIS_IRT: 割り込みイベント無効化 (S7-1500)
- EN_IRT: 割り込みイベント有効化 (S7-1500)
- DIS_AIRT: 高優先度割り込みと非同期エラーイベント実行の遅延 (S7-1200, S7-1500)
- EN_AIRT: 高優先度割り込みと非同期エラーイベント実行の有効化 (S7-1200, S7-1500)

DIS_IRT: 割り込みイベント無効化

説明

「DIS_IRT」命令を使用して、新規割り込み、および非同期エラーイベントの処理を無効化します。 無効にするとは、割り込みイベントが発生した場合に、CPU のオペレーティングシステムが次のように応答することを意味します。

- 割り込み OB および非同期エラー OB の**いずれも呼び出しません**。
- また、割り込み OB または非同期エラー OB がプログラミングされていない場合、通常の応答を**トリガしません**。

割り込みおよび非同期エラーイベントを無効にすると、これはすべての優先度クラスに対して有効のままになります。 無効は、「EN_IRT」命令またはウォームまたはコールドリスタートのみでキャンセルすることができます。

オペレーティングシステムが割り込みおよび非同期エラーイベントが発生した場合に診断バッファに書き込むかは、MODE に選択した入力パラメータ設定によって異なります。

注記
「DIS_IRT」命令をプログラミングする場合、発生するすべての割り込みが破棄されることを覚えておいてください。

パラメータ

次の表に、「DIS_IRT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	BYTE	I、Q、M、D、L、または定数	どの割り込みおよび非同期エラーが無効になるかを指定します。
OB_NR	Input	INT	I、Q、M、D、L、または定数	DB 番号
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ MODE

MODE (B#16#...)	意味
00	すべての新規に発生する割り込みおよび非同期エラーイベントが無効になります。(同期エラーは無効になりません。) OB_NR パラメータに値「0」を割り当てます。 診断バッファへの入力は継続します。
01	指定した割り込みクラスに属するすべての新規に発生するイベントは無効になります。次のように指定して、割り込みクラスを識別します。

	• 時刻割り込み: 10 • 遅延割り込み: 20 • 周期割り込み: 30 • プロセス割り込み: 40 • DPV1 に関する割り込み: 50 • マルチコンピューティング割り込み: 60 • 冗長化エラー割り込み: 70 • 非同期エラー割り込み: 80 診断バッファへの入力は継続します。
02	指定された割り込みのすべての新たな発生は無効になります。 DB 番号を使って割り込みを指定します。 診断バッファへの入力は継続します。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	OB_NR 入力パラメータに無効な値が含まれています。
8091	MODE 入力パラメータに無効な値が含まれています。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

EN_IRT: 割り込みイベント有効化

説明

この命令を使用して、新規割り込みと、前に「DIS_IRT」命令で無効にした非同期エラーイベントの処理を有効にします。 これは、割り込みイベントが発生した場合、CPU のオペレーティングシステムが、次のいずれかの方法で応答することを意味します。

- 割り込み OB または非同期エラー OB を呼び出します。

  または

- 割り込み OB または非同期エラー OB がプログラミングされていない場合は、標準の応答をトリガします。

パラメータ

次の表に、「EN_IRT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	BYTE	I、Q、M、D、L、または定数	どの割り込みおよび非同期エラーイベントを有効にするかを指定します。
OB_NR	Input	INT	I、Q、M、D、L、または定数	DB 番号
RET_VAL	Return	INT	I、Q、M、D、L	命令の実行中にエラーが発生した場合、戻り値にエラーコードが含まれます。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ MODE

MODE	意味
0	すべての新規に発生する割り込みおよび非同期エラーイベントが有効になります。
1	指定した割り込みクラスに属するすべての新規に発生するイベントが有効になります。 次のように指定して、割り込みクラスを識別します。 • 時刻割り込み: 10 • 遅延割り込み: 20 • 周期割り込み: 30 • プロセス割り込み: 40 • DPV1 に関する割り込み: 50 • マルチコンピューティング割り込み: 60 • 冗長化エラー割り込み: 70 • 非同期エラー割り込み: 80

2	指定した割り込みのすべての新規に発生するイベントが有効になります。 DB 番号を使って割り込みを指定します。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	OB_NR 入力パラメータに無効な値が含まれています。
8091	MODE 入力パラメータに無効な値が含まれています。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

DIS_AIRT: 高優先度割り込みと非同期エラーイベント実行の遅延

説明

「DIS_AIRT」を使用して、現在のオーガニゼーションブロックよりも優先度が高い割り込み OB の処理を遅延します。

オーガニゼーションブロックで「DIS_AIRT」を複数回呼び出すことができます。 「DIS_AIRT」呼び出しは、オペレーティングシステムでカウントされます。 「DIS_AIRT」が実行されるたびに、処理がさらに遅延されます。 遅延をキャンセルするには、「EN_AIRT」命令を実行する必要があります。 すべての遅延をキャンセルするには、「EN_AIRT」の実行数が「DIS_AIRT」呼び出しの数と等しくなることが必要です。

「DIS_AIRT」命令の RET_VAL パラメータで遅延数を照会できます。 RET_VAL パラメータの値が「0」の場合、遅延はありません。

パラメータ

以下の表に、「DIS_AIRT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Return	INT	I、Q、M、D、L	遅延回数

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

戻り値	説明
n	「n」は、処理遅延の数を示します。つまり、命令の完了後の DIS_AIRT 呼び出しの数を示します(割り込み処理は、n = 0 の場合にのみ再び有効になります。EN_AIRT を参照)。

EN_AIRT: 高優先度割り込みと非同期エラーイベント実行の有効化

説明

「EN_AIRT」を使用して、「DIS_AIRT」命令で遅延された割り込みが発生した場合にオーガニゼーションブロックの処理を有効にします。

「EN_AIRT」を実行した場合、「DIS_AIRT」が呼び出されたときにオペレーティングシステムによって登録された処理遅延がキャンセルされます。 すべての遅延をキャンセルするには、「EN_AIRT」の実行数が「DIS_AIRT」呼び出しの数と等しくなることが必要です。 たとえば、「DIS_AIRT」を 5 回呼び出し、よって処理を 5 回遅延した場合、5 回すべての遅延をキャンセルするには「EN_AIRT」を 5 回呼び出す必要があります。

「EN_AIRT」の実行後にまだ有効になっていない割り込み遅延の数を「EN_AIRT」命令の RET_VAL パラメータで照会できます。 RET_VAL パラメータの値「0」は、「DIS_AIRT」で有効になったすべての遅延がキャンセルされたことを意味します。

パラメータ

以下の表に、「EN_AIRT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Return	INT	I、Q、M、D、L	構成された遅延の回数

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

戻り値/ エラーコード* (W#16#...)	説明
n	「n」は、命令が完了した後にまだ有効になっていない処理遅延の数を示します(割り込み処理は、n = 0 の場合のみ有効になります)。
8080	割り込み処理が既に有効になっていたにもかかわらず、ファンクションが再び呼び出されました。
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

アラーム

この章には下記に関する情報が記載されています：

- Program_Alarm: 関連値を含むプログラムアラームを生成 (S7-1500)
- Get_AlarmState: アラームステータスの出力 (S7-1500)
- Gen_UsrMsg:ユーザー診断アラームの生成 (S7-1500)

Program_Alarm: 関連値を含むプログラムアラームを生成

説明

「関連値を含むプログラムアラームを生成」命令は信号をモニタし、プログラムアラームのパラメータ SIG で信号の切り替えを生成します(定義については、 **アラーム設定の概要**も参照)。信号が 0 から 1 に切り替わると着信プログラムアラームが生成され、信号が 1 から 0 に切り替わると発信プログラムメッセージが生成されます。プログラムアラームはプログラム実行と同期的にトリガされます。

プログラムアラームにはパラメータ SD_i で最大 10 の関連値を追加できます(0 ≤ i ≤10)。 SIG パラメータでの信号切り替わり時に関連値が検出され、プログラムアラームに割り当てられます。 関連値の設定に関する詳細情報は、以下のセクションを参照してください。 **アラームへの関連する値の挿入**, **関連する値の例**.

各着信および発信アラームには、タイムスタンプが割り当てられます。

- 信号の切り替えが発生した際の PLC の現在のシステム時刻が既定値として使用されます(TIME-STAMP パラメータの既定値)。
- 他のタイムスタンプを指定する場合は、TIMESTAMP パラメータで作成可能です。

  時間値は必ずシステム時刻(つまり UTC)で指定する必要があります。これは、システム時刻がプラント全体の時刻同期に使用されるためです。
- アラームのタイムスタンプをローカルタイムにするには、ローカルタイムをシステム時刻に変換する変換モジュールをシステムに接続する必要があります。 これ以外に、タイムスタンプをアラームディスプレイ内に正しく表示させる方法はありません。

CPU の現在のシステム時刻を使用するには、TIMESTAMP パラメータを既定値(LDT#1970-01-01-00:00:00.0)にセットします。

「関連値を含むプログラムアラームを生成」命令の呼び出し

この命令は、ファンクションブロック(FB)内でのみ呼び出し可能です。 ブロックは、同期的に処理されます。 つまり、ブロックが終了すると、アラームが直ちにトリガされます。 処理中にエラーがあった場合、エラーコードが出力されます。

FB 内にこの命令が挿入されると、データタイプ「Program_Alarm」のマルチインスタンスがブロックインターフェースの「Static」セクション内に作成されます。 表示されるダイアログで、このマルチインスタンスの任意の名前を選択することができます。 これは、プログラムアラームの名前でもあります。

最後に、要件に一致する命令のパラメータを追加します(「パラメータ」の表を参照)。

プログラムアラームの設定

「Static」セクション、または FB のネットワーク内でプログラムアラームの名前を選択すると、プログラムアラーム設定が[プロパティ]ウィンドウに表示されます。 このウィンドウでは、アラームクラスや優先度などを選択し、アラームテキストを編集することが可能です。

ここで行われた設定は、プロジェクトツリー内で編集できます。 [PLC アラーム]に移動し、[プログラムアラーム]タブを開きます。 [アラームタイプ]テーブルに、既存のすべてのプログラムアラームが表示されます。

パラメータ

次の表に、「関連値を含むプログラムアラームを生成」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SIG	Input	BOOL	I、Q、M、D、L、T、C、または定数	モニタされる信号。 • 信号立ち上がりエッジ: 着信プログラムアラームが生成されます。 • 信号立ち下がりエッジ: 発信プログラムアラームが生成されます。
TIMESTAMP	Input	LDT	M、D、L、または定数	このパラメータを使用し、たとえば配布されたタイムスタンプが付けられた入力信号からのタイムスタンプをアラームに割り当てます。時間値は必ずシステム時刻(つまり UTC)で指定する必要があります。これは、システム時刻がプラント全体の時刻同期に使用されるためです。 • 「割り当てなし」は、信号の切り替わりが発生すると CPU のシステム時刻が割り込みタイムスタンプとして使用されることを意味します(既定)。 • 信号の切り替わりが発生した際の割り込みタイムスタンプとして、任意のシステム時刻を使用できます。 注記: 割り込みのタイムスタンプをローカルタイムにするには、ローカルタイムをシステム時刻に変換する変換モジュールをシステムに接続する必要があります。 これ以外に、タイムスタンプを割り込みディスプレイ内に正しく表示させる方法はありません。
SD_i	Input	VARIANT	I、Q、M、D、L	i 番目の関連する値(1 ≤ i ≤ 10) 2 進数、整数、浮動小数点数、または文字列を関連値として使用できます。
Error	Output	BOOL	I、Q、M、D、L	ステータスパラメータ Error Error = TRUE は、処理中にエラーが発生したことを示します。 可能性のあるエラー原因が Status パラメータに表示されます。

Status	Output	WORD	I、Q、M、D、L	ステータスパラメータ Status エラー情報の表示(「Error および Status パラメータ」を参照)。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

Error および Status パラメータ

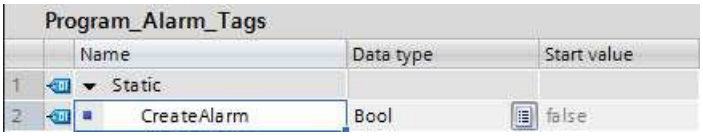
次の表に、Error および Status パラメータで出力できるすべての特定のエラー情報を示します。

Error	Status*	説明
0	0000	エラーなし、またはパラメータ SIG に信号のエッジがないため、命令が処理されませんでした。
1	0085	「情報のみ」アラームタイプ
1	8001	無効な静的アラーム情報
1	8002	有効な静的アラーム情報なし
1	8004	アラームの関連値が、最大サイズ 512 バイトに達しました。
1	8005	SIG パラメータで信号立ち上がりエッジが保留中であり、まだ確認済みでない保留中のアラームが存在します。
1	8007	先行する着信アラームが存在しなかった発信アラーム。
1	8087	静的アラームが無効です。
1	8089	アラームが長すぎます。
1	80Ax	SD_i パラメータの値が無効です。
1	80C1	初期化ルーチンが実行中のため、CPU が現在アラームを生成できません (RUN でのダウンロード後の場合など)。 後で再度実行してみてください。
1	80C2	時間単位当たりの最大数のアラームが送信されました。 後で再度実行してみてください。
1	80C3	動的アラームインスタンスがすべて使用中です。 後で再度実行してみてください。
1	80C4	アラームが出力されており、上書きできません。 後で再度実行してみてください。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

例

次の例では、信号切り替えの関連値と共にプログラムアラームを生成します。

モニタする信号値を保存するために、グローバルデータブロックに 1 つのタグを作成します。

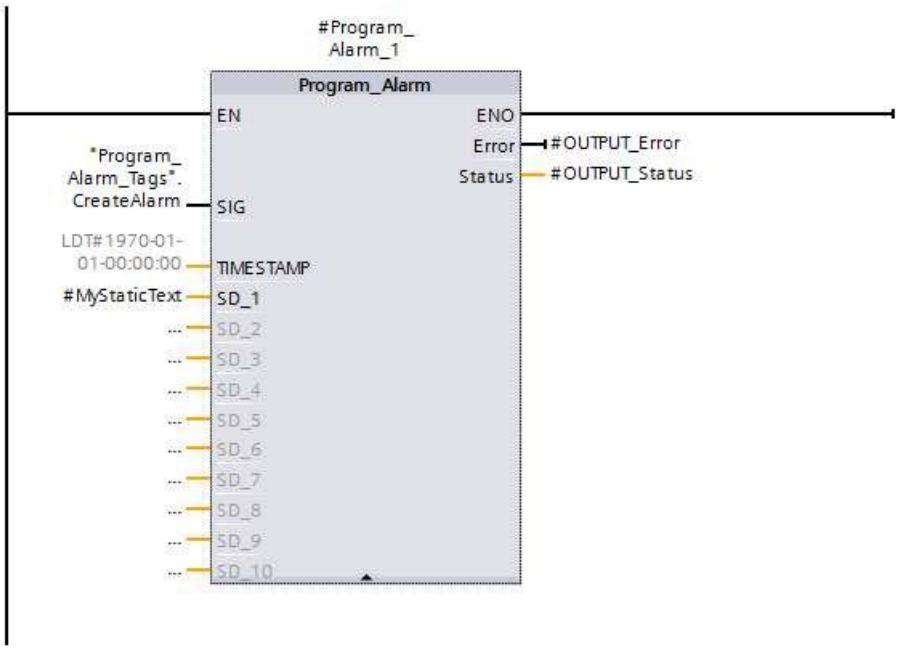


この命令はファンクションブロックで呼び出されます。 命令を相互接続するファンクションブロックに 4 つのパラメータを作成します。

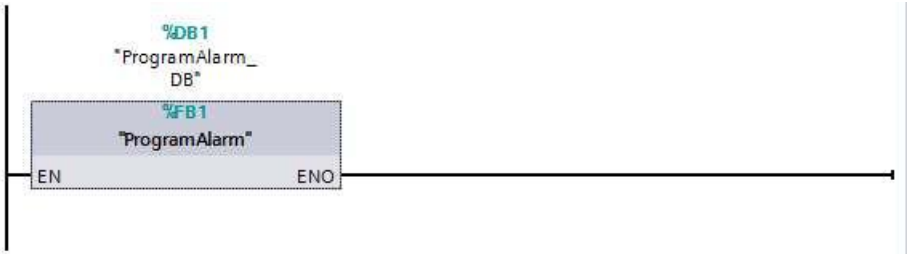
ProgramAlarm

		Name	Data type	Default value
1		Input		
2		<Add new>		
3		Output		
4		<Add new>		
5		InOut		
6		<Add new>		
7		Static		
8		MyStaticText	String	'MyStaticText'
9		Program_Alarm_1	Program_Alarm	
10		Temp		
11		OUTPUT_Error	Bool	
12		OUTPUT_Status	Word	

この命令のパラメータを以下のように相互接続します。



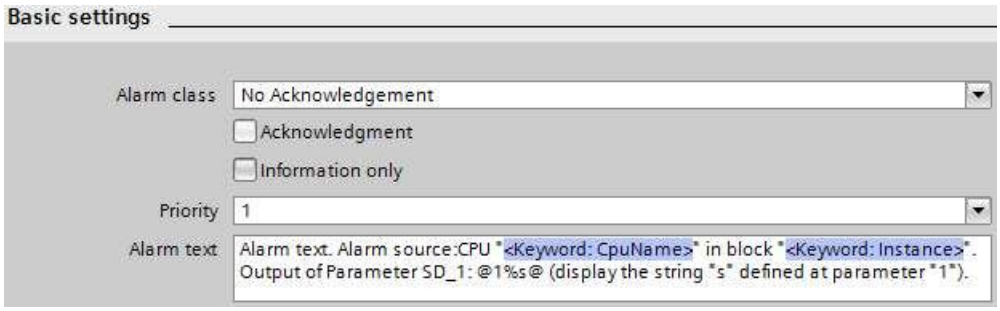
OB でファンクションブロックを呼び出します。



命令が作成されると、PLC アラームが自動的に生成されます。 PLC アラームダイアログを開き、[プログラムアラーム]タブで処理するアラームを選択します。 2 つのキーワードを含む、アラームテキストを作成します。

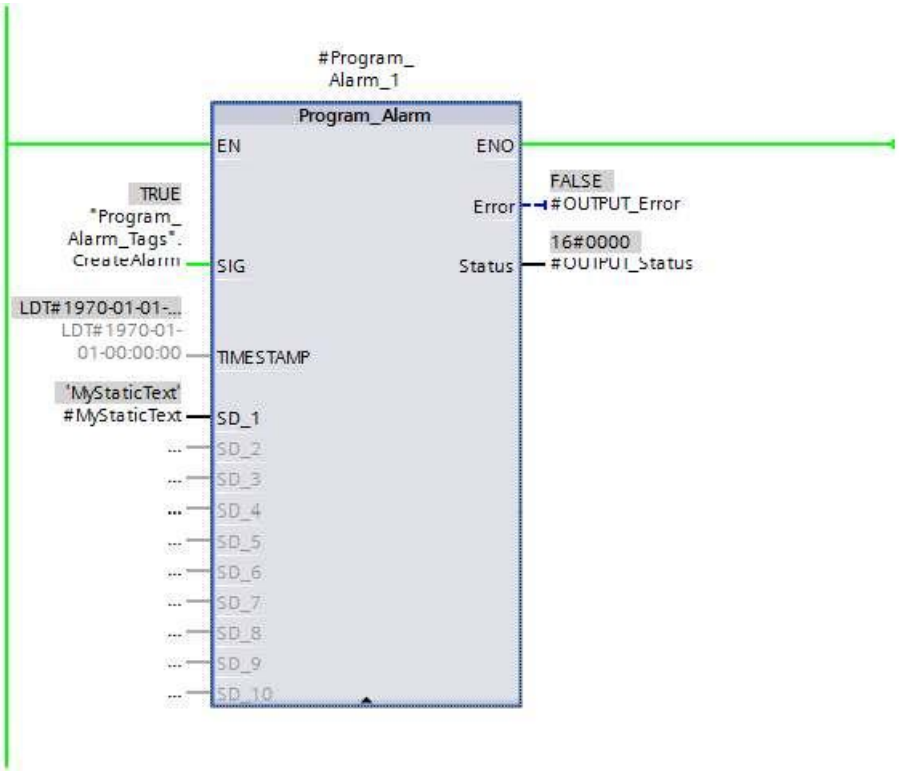
注記: テキストボックスを右クリックすると、キーワード、タグ、またはテキストリストを挿入できます。

文字列「@1%s@」の結果として、パラメータ SD_1(「#MyStaticText」)の値が読み出され、文字列として出力されます。



パラメータ SIG(「CreateAlarm」)がシグナル状態「TRUE」を持つ場合、PLC アラームが出力されます。 出力パラメータ STATUS(「OUTPUT_Status」)で、値「0001」は、信号の切り替えが行われたことを示します。 また、それ以外の処理が行われていないことを示します(値は「0000」)。 パラメータ SD_1(「#MyStaticText」)で、PLC アラームの関連値が出力されます。

タイムスタンプをパラメータ TIMESTAMP に転送できます。 相互接続されない場合、パラメータ TIMESTAMP は CPU クロックのローカル時刻を出力します。 出力パラメータ ERROR(「OUTPUT_Error」)で、命令の処理がエラーなしで実行されていることを示します。



PLC アラームの出力には、CPU の Web サーバーを使用します。 Web サーバーを使用するためには、以下のことが必要となります。

- Web サーバーは CPU の設定で有効にする必要があります。

インターネットブラウザを使用して(CPU の IP アドレス経由で)Web サーバーを開き、Web サーバーメニューにログオンします。 CPU は信号がモニタされている限りアラームテキストを出力し(「CreateAlarm」)、値「TRUE」を表示します。

Time	Message text
02:32:09.226 pm	Alarm text. Alarm source:CPU "PLC_2" in block "ProgramAlarm_DB". Output of Pa

例

以下の FAQ ID の Siemens Industry Online Support に、詳細なアプリケーション例が記載されています。 98210758.

Get_AlarmState: アラームステータスの出力

説明

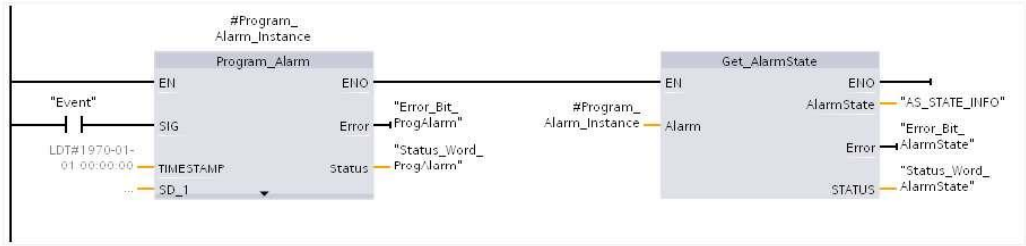
「アラームステータスの出力」命令を使用して、プログラムアラームのアラームステータスを出力します。

以下の 3 つのアラームステータスの可能性があります。

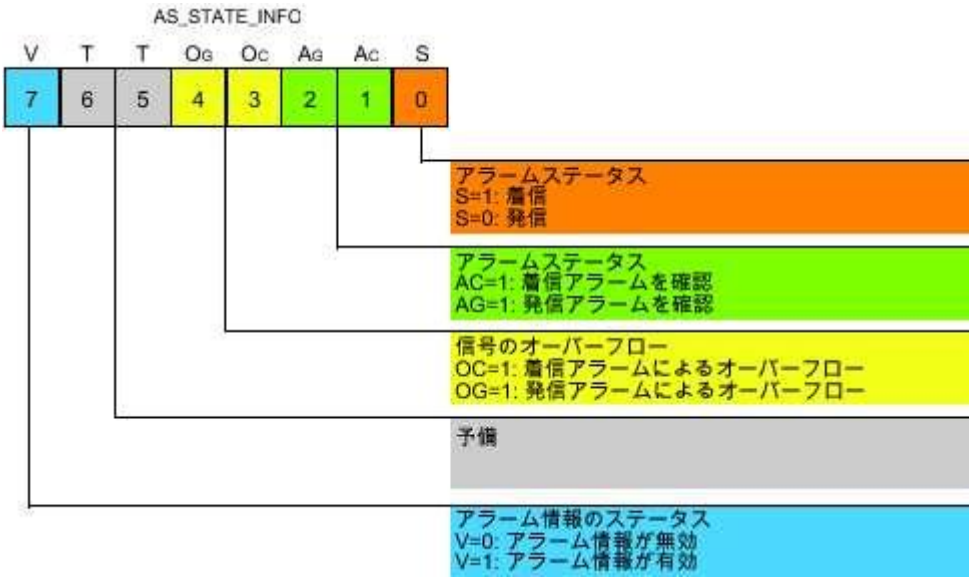
- 着信
- 発信
- 確認応答あり

アラームステータスの出力は、常に「関連値を含むプログラムアラームを生成」命令を使用して作成されたプログラムアラームを参照します。

プログラムアラームは、Alarm 入力パラメータを使用して選択されます。「関連値を含むプログラムアラームを生成」命令のインスタンス DB をパラメータ Alarm で指定します。



アラームステータスは、AlarmState 出力パラメータによってバイトで返されます。 次の図に、個々のビットの意味を示します。



命令の実行ステータスは、出力パラメータ Error および STATUS で示されます。

パラメータ

次の表に、「アラームステータスの出力」命令のパラメータをリストします。

パラメータ	宣言	データタイプ	メモリ領域	説明
Alarm	Input	ALARM_BASE	D	「関連値を含むプログラムアラームを生成」命令のインスタンス • Alarm.Messagetype = Alarm_AP、ビット Ac のシグナル状態が 0 または 1 であり、ビット Ag のシグナル状態が 1 です ○ 非アクティブ: 0x86 (1000 0110) ○ アクティブ/確認応答なし: 0x85 (1000 0101) ○ アクティブ/確認済み: 0x87 (1000 0111) ○ 発信/確認応答なし: 0x84 (1000 0100) • Alarm.Messagetype = Notify_AP、ビット Ac のシグナル状態が 0 または 1 であり、ビット Ag のシグナル状態が 1 です ○ 非アクティブ: 0x86 (1000 0110) ○ アクティブ: 0x85 (1000 0101) • Alarm.Messagetype = Inforeport_AP、ビット Ac と Ag のシグナル状態が両方とも 1 です ○ 静的: 0x86 (1000 0110) • アラームが非アクティブであるか、または Inforeport である場合、ビット S のシグナル状態は 0 です。
AlarmState	Output	BYTE	I、Q、M、D、L	ビット配列としてのアラームのステータス
Error	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ

				このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS を空きデータ領域にコピーする必要があります。

有効なデータタイプに関する追加情報については、「関連項目」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
8001	無効な静的アラームインスタンス
8002	アラームの ID が無効です。
8003	アラームクラス内に有効なアラームがありません。 • Alarm_AP: アラームが発信中または確認済みです。 • Notify_AP: アラームが発信中です。 • Inforeport ビット V は、シグナル状態 0 に設定されます。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例では、プログラムアラームのアラームステータスを出力します。

グローバルデータブロックにデータを保存するために、4 つのタグを作成します。

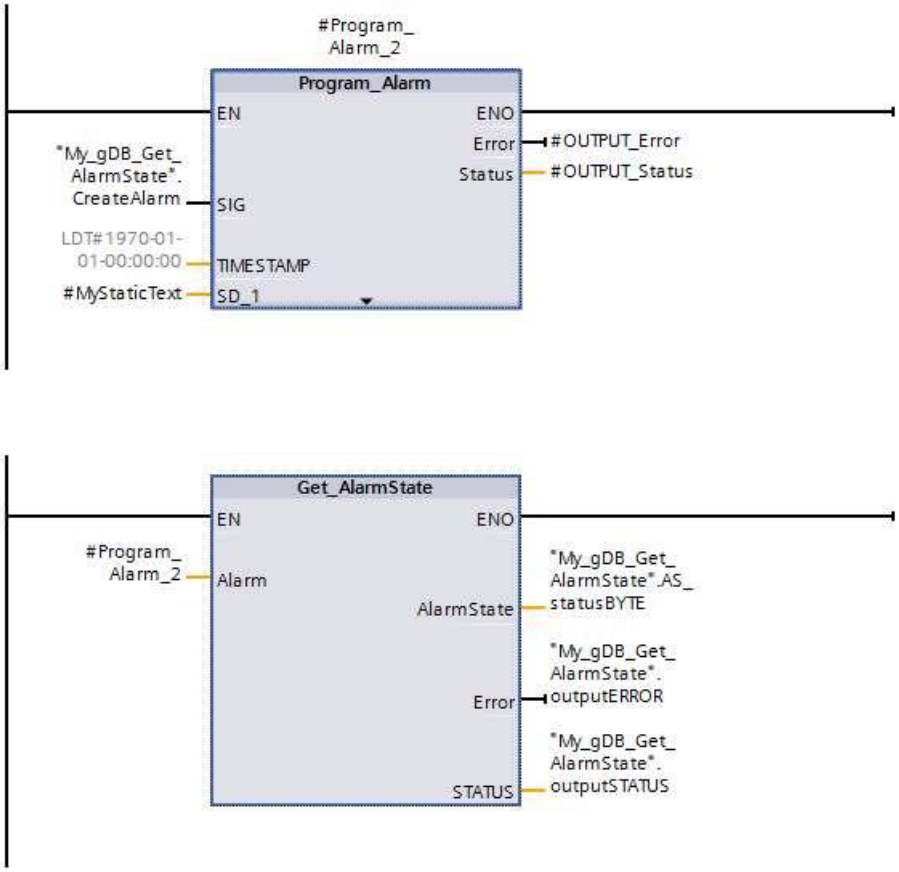
My_gDB_Get_AlarmState

	Name	Data type	Start value
1	Static		
2	AS_statusBYTE	Byte	16#0
3	outputERROR	Bool	false
4	outputSTATUS	Word	16#0
5	CreateAlarm	Bool	false

「Get_AlarmState」命令は、「Program_Alarm」命令と共にファンクションブロックで呼び出されます。 命令を相互接続するファンクションブロックに 4 つのパラメータを作成します。

7	Static		
8	MyStaticText	String	'MyStaticText'
9	Program_Alarm_2	Program_Alarm	
10	Temp		
11	OUTPUT_Error	Bool	
12	OUTPUT_Status	Word	

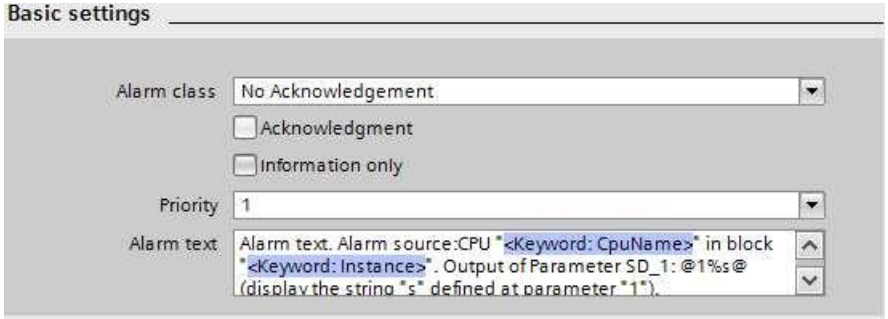
この命令のパラメータを以下のように相互接続します。



OB でファンクションブロックを呼び出します。

「Program_Alarm」命令が作成されると、PLC アラームが自動的に生成されます。 PLC アラームダイアログを開き、[プログラムアラーム]タブで処理するアラームを選択します。 2 つのキーワードを含む、アラームテキストを作成します。

アラームの以下の設定を選択します。 アラームはアラームクラス「Notify_AP」に属しています。



「Program_Alarm」命令の入力パラメータ SIG(「CreateAlarm」)がシグナル状態「TRUE」を持つ場合、PLC アラームが出力されます。

「Get_AlarmState」命令の場合は、以下が発生します。 PLC アラームは、入力パラメータ ALARM を使用して「Get_AlarmState」命令に通知されます。 出力パラメータ AlarmState(「AS_statusBYTE」)で、アラームクラス「Notify_AP」に対応しているアラームがアクティブであることを示します。

出力パラメータ ERROR(「outputERROR」)および STATUS(「outputSTATUS」)で、命令の処理がエラーなしで実行されていることを示します。

My_gDB_Get_AlarmState

	Name	Data type	Start value	Monitor value
1	Static			
2	AS_statusBYTE	Byte	16#0	16#85
3	outputERROR	Bool	false	FALSE
4	outputSTATUS	Word	16#0	16#0000
5	CreateAlarm	Bool	false	TRUE

PLC アラームの出力には、CPU の Web サーバーを使用します。

例

以下の FAQ ID の Siemens Industry Online Support に、詳細なアプリケーション例が記載されています。 98210758.

Gen_UsrMsg:ユーザー診断アラームの生成

説明

「Gen_UsrMsg」命令を使用して、診断バッファに入力するアラームを生成します。

Mode パラメータを使用して、着信アラームまたは発信アラームのいずれを生成するかを選択します。

- Mode = 1 の場合: 着信アラームを作成します。
- Mode = 2 の場合: 発信アラームを作成します。
- 着信アラームまたは発信アラームのいずれが生成されるかに関係なく、アラームは常に属性「情報のみ」を持ちます。

診断バッファのエントリは同期的に作成されます。 アラーム送信は非同期的に行われます。

命令の実行中にエラーが発生した場合、このエラーはパラメータ RET_VAL によって出力されます。

アラームの内容

アラームの内容は、テキストリストで定義されます。

- TextListID パラメータで、使用するテキストリストを指定します。 そのためには、プロジェクトナビゲーションでダイアログ[テキストリスト]を開きます。 ダイアログ[テキストリスト]で列[ID]を表示します。 TextListID パラメータで ID を適用します。
- TextID パラメータを使用して、診断バッファに書き込みたいテキストリストエントリを選択します。 そのためには、TextID パラメータで列[範囲開始/範囲終了]から番号を適用することによって、[テキストリストエントリ]ダイアログからエントリを選択します。 テキストリストエントリについて[範囲開始]列と[範囲終了]列の両方で同じ番号を使用する必要があります。
- テキストリストに関する追加情報については、以下を参照してください。 テキストリスト

関連値の定義

新しい関連値は、テキストエントリで定義してからアラームに追加することができます。

- 関連値を定義するには、テキストリストエントリに次の情報を追加します。

  @<関連値の番号><エレメントタイプ><形式の指定>@
- システムデータタイプ AssocValues を使用して、アラームの生成時にどの関連値を追加するかを指定します。
- 関連付けされた値の構造に関する追加情報については、以下を参照してください。 関連する値の構造

パラメータ

以下の表に、「Gen_UsrMsg」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
Mode	Input	UInt	I、Q、M、D、L、または定数	アラームのステータスを選択するためのパラメータ: • 1: 着信アラーム • 2: 発信アラーム

TextID	Input	UInt	I、Q、M、D、L、または定数	アラームテキストに使用するテキストリストエントリの ID。
TextListID	Input	UInt	I、Q、M、D、L、または定数	テキストリストエントリを含むテキストリストの ID。
Ret_Val	Return	Int	I、Q、M、D、L	命令のエラーコード。
AssocValues	InOut	VARIANT	D、L	関連値の定義を可能にするシステムデータタイプ AssocValues へのポインタ。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

AssocValues パラメータ

システムデータタイプ AssocValues を使用して、どの関連値が送信されるかを定義します。 最大 8 つの関連値が可能です。 システムデータタイプの構造体を作成するには、データブロックとしてデータタイプ「AssocValues」を入力します。

関連値を選択するには、Value[x]パラメータに対して関連値の番号を入力します。 以下のことに注意してください。

- TextID および TextListID の値は、命令によって、送信される関連値として扱われます。 その結果、「1」および「2」は、関連値をアドレス指定するための番号として既に割り当てられています。 番号「1」または「2」を関連値をアドレス指定するために使用しないでください。
- 関連値をパラメータ Value [1]で番号「3」として、パラメータ Value [2]で番号「4」としてなどアドレス指定します。

Byte (バイト)	パラメータ	データタイプ	開始値	説明	関連値の番号
0..1	Value[1]	UINT	0	アラームの最初の関連値。	3
2..3	Value[2]	UINT	0	アラームの 2 番目の関連値。	4
4..5	Value[3]	UINT	0	...	5
6..7	Value[4]	UINT	0	...	6
8..9	Value[5]	UINT	0	...	7
10..11	Value[6]	UINT	0	...	8
12..13	Value[7]	UINT	0	...	9
14..15	Value[8]	UINT	0	アラームの 8 番目の関連値。	10

RET_VAL パラメータ

次の表に、RET_VAL パラメータで出力できるすべての特定のエラー情報を示します。

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。

8080	Mode パラメータの値がサポートされていません。
80C1	多すぎる並列呼び出しによるリソースのボトルネック。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

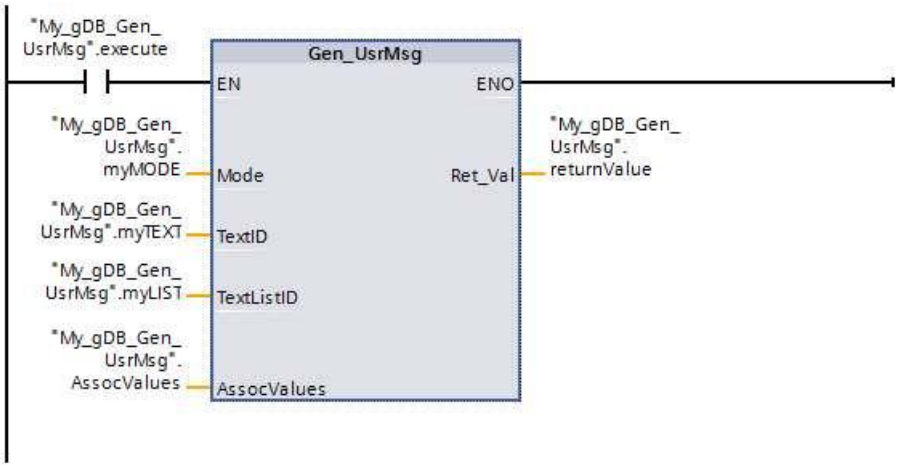
次の例では、診断バッファに入力されるアラームを生成します。

グローバルデータブロックにデータを保存するために、5 つのタグと 1 つの「AssocValues」構造体 (AssocValues データタイプ)を作成します。

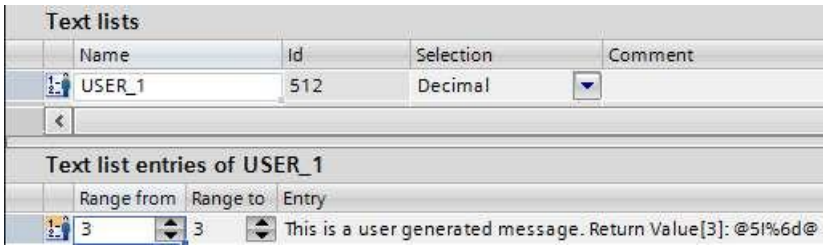
My_gDB_Gen_UsrMsg

	Name	Data type	Start value
1	Static		
2	myMODE	UInt	1
3	myTEXT	UInt	3
4	myLIST	UInt	512
5	returnValue	Int	0
6	execute	Bool	false
7	AssocValues	AssocValues	
8	Value	Array[1..8] of UInt	
9	Value[1]	UInt	33
10	Value[2]	UInt	411
11	Value[3]	UInt	578
12	Value[4]	UInt	6122
13	Value[5]	UInt	722
14	Value[6]	UInt	8111
15	Value[7]	UInt	9829
16	Value[8]	UInt	10457

この命令のパラメータを以下のように相互接続します。



「テキストリスト」エントリを使用して、アラームのテキストリストおよびテキストリストエントリを作成します。 パラメータ TextListID(「myLIST」)でテキストリストの ID を適用します。 パラメータ TextID(「myTEXT」)でテキストリストエントリの ID(範囲...)を適用します。 アラームパラメータを次のように割り当てます。

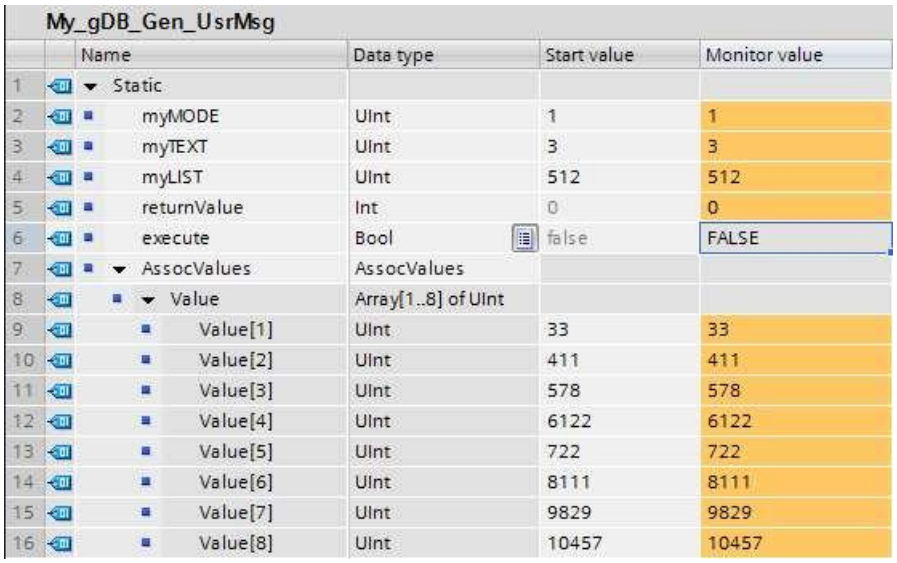


ノーマルオープン(「execute」)がシグナル状態「TRUE」を表示する場合、「Gen_UsrMsg」命令が実行されます。 パラメータ Mode(「myMODE」)の値に従って、着信アラームを生成します。 出力するアラームは、パラメータ TextListID(「myLIST」)および TextID(「myTEXT」)を使用して命令に伝えられます。 アラームの関連値は、パラメータ AssocValues(「AssocValues」)を使用して転送されます。

アラームが生成されると、アラームテキストに含まれる文字列「@5I%6d@」が次のように解釈されます。

- 番号「5」の関連値は、INT データタイプで読み出されます。 番号は、「AssocValues」構造体のパラメータ Value[3]に対応します。
- 関連値は 10 進数として出力されます。 10 進数は 6 桁までに制限されています。

出力パラメータ Ret_Val(「returnValue」)で、命令の処理がエラーなしで実行されていることを示します。



S7-1500 シリーズの CPU の場合は、アラームの出力にはエントリ[オンライン&診断|診断バッファ]を開きます。

例

以下の FAQ ID の Siemens Industry Online Support に、詳細なアプリケーション例が記載されています。 98210758.

診断

この章には下記に関する情報が記載されています：

RD_SINFO: 現在の OB 起動情報読み出し

説明

「RD_SINFO」命令を使用し、以下の開始情報を読み出します。

- まだ完了していない、呼び出された最後の OB、および
- 開始された最後の起動 OB。

いずれの場合もタイムスタンプはありません。 呼び出しが OB 100、OB 101 または OB 102 にある場合、2 つの同一の開始情報メッセージが返されます。

パラメータ

次の表に、「RD_SINFO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
TOP_SI	Output	VARIANT	D、L	現在の OB の開始情報
START_UP_SI	Output	VARIANT	D、L	最後に開始された起動 OB の開始情報

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

TOP_SI パラメータの SDT

次の表に、TOP_SI パラメータに使用可能な SDT を示します。

オーガニゼーションブロック (OB)	システムデータタイプ(SDT)	システムデータタイプ番号
任意	SI_classic	592
	SI_none	593
ProgramCycleOB	SI_ProgramCycle	594
TimeOfDayOB	SI_TimeOfDay	595
TimeDelayOB	SI_Delay	596
CyclicOB	SI_Cyclic	597
ProcessEventOB	SI_HWInterrupt	598
ProfileEventOB StatusEventOB UpdateEventOB	SI_Submodule	601
SynchronousCycleOB	SI_SynchCycle	602
IOredundancyErrorOB	SI_IORedundancyError	604
CPUredundancyErrorOB	SI_CPURedundancyError	605

TimeErrorOB	SI_TimeError	606
DiagnosticErrorOB	SI_DiagnosticInterrupt	607
PullPlugEventOB	SI_PlugPullModule	608
PeripheralAccessErrorOB	SI_AccessError	609
RackStationFailureOB	SI_StationFailure	610
ServoOB	SI_Servo	611
IpoOB	SI_Ipo	612
StartupOB	SI_Startup	613
ProgrammingErrorOB IOaccessErrorOB	SI_ProgIOAccessError	614

START_UP_SI パラメータの SDT

次の表に、START_UP_SI パラメータに使用可能な SDT を示します。

システムデータタイプ(SDT)	システムデータタイプ番号
SI_classic	592
SI_none	593
SI_Startup	613

構造体

次の表に、個々の構造体の構造体エレメントの意味を示します。

SI_classic 構造体

構造体エレメント	データタイプ	説明
EV_CLASS	BYTE	• ビット 0～3: イベント ID • ビット 4～7: イベントクラス
EV_NUM	BYTE	イベント番号
PRIORITY	BYTE	優先度クラス番号 (B#16#FE の意味: OB が使用不可、または無効、または現在の動作モードでは開始できない)
NUM	BYTE	DB 番号
TYP2_3	BYTE	データ ID 2_3: 次に入力された情報を特定 ZI2_3
TYP1	BYTE	データ ID 1: 次に入力された情報を特定 ZI1
ZI1	WORD	追加情報 1
ZI2_3	DWORD	追加情報 2_3

SI_none 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)

SI_ProgramCycle 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 1	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Initial_Call	BOOL	OB_Class = 1、30、52、61、65 の場合
Remanence	BOOL	OB_Class = 1 の場合

SI_TimeOfDay 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 10	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
CaughtUp	BOOL	OB_Class = 10 の場合
SecondTime	BOOL	OB_Class = 10 の場合

SI_Delay 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 20	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Sign	WORD	OB_Class = 20 の場合

SI_Cyclic 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 30	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Initial_Call	BOOL	OB_Class = 1、30、52、61、65 の場合
Event_Count	INT	OB_Class = 30、51、52、61、65、91、92 の場合

SI_HWInterrupt 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 40	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LADDR	HW_IO	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
USI	WORD	OB_Class = 40 の場合
IChannel	USINT	OB_Class = 40 の場合
EventType	BYTE	OB_Class = 40 の場合

SI_Submodule 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LADDR	HW_IO	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
Slot	UINT	OB_Class = 55、56、57 の場合
Specifier	WORD	OB_Class = 55、56、57 の場合

SI_SynchCycle 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報

OB_Class	USINT := 61	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Initial_Call	BOOL	OB_Class = 1、30、52、61、65 の場合
PIP_Input	BOOL	OB_Class = 61、91、92 の場合
PIP_Output	BOOL	OB_Class = 61、91、92 の場合
IO_System	USINT	OB_Class = 61、91、92 の場合
Event_Count	INT	OB_Class = 30、51、52、61、65、91、92 の場合
SyncCycleTime	LTIME	計算されたサイクルタイム

SI_IORedundancyError 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 70	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LADDR	HW_ANY	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
Event_Class	BYTE	OB_Class = 70、83、85、86 の場合
Fault_ID	BYTE	OB_Class = 70、80、83、85、86 の場合

SI_CPURedundancyError 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 72	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Switch_Over	BOOL	OB_Class = 72 の場合

SI_TimeError 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 80	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Fault_ID	BYTE	OB_Class = 70、80、83、85、86 の場合
Csg_OBnr	OB_ANY	OB_Class = 80 の場合

Csg_Prio	UINT	OB_Class = 80 の場合

SI_DiagnosticInterrupt 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 82	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
IO_State	WORD	OB_Class = 82 の場合
LADDR	HW_ANY	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
Channel	UINT	OB_Class = 82 の場合
MultiError	BOOL	OB_Class = 82 の場合

SI_PlugPullModule 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 83	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LADDR	HW_IO	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
Event_Class	BYTE	OB_Class = 70、83、85、86 の場合
Fault_ID	BYTE	OB_Class = 70、80、83、85、86 の場合

SI_AccessError 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 85	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LADDR	HW_IO	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
Event_Class	BYTE	OB_Class = 70、83、85、86 の場合
Fault_ID	BYTE	OB_Class = 70、80、83、85、86 の場合
IO_Addr	UINT	OB_Class = 85 の場合

IO_LEN	UINT	OB_Class = 85 の場合

SI_StationFailure 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 86	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LADDR	HW_IO	OB_Class = 40、51、55、56、57、70、82、83、85、86、91、92 の場合
Event_Class	BYTE	OB_Class = 70、83、85、86 の場合
Fault_ID	BYTE	OB_Class = 70、80、83、85、86 の場合

SI_Servo 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 91	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Initial_Call	BOOL	OB_Class = 1、30、52、61、65 の場合
PIP_Input	BOOL	OB_Class = 61、91、92 の場合
PIP_Output	BOOL	OB_Class = 61、91、92 の場合
IO_System	USINT	OB_Class = 61、91、92 の場合
Event_Count	INT	OB_Class = 30、51、52、61、65、91、92 の場合
Synchronous	BOOL	

SI_Ipo 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 92	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
Initial_Call	BOOL	OB_Class = 1、30、52、61、65 の場合
PIP_Input	BOOL	OB_Class = 61、91、92 の場合
PIP_Output	BOOL	OB_Class = 61、91、92 の場合
IO_System	USINT	OB_Class = 61、91、92 の場合

Event_Count	INT	OB_Class = 30、51、52、61、65、91、92 の場合
Reduction	UINT	OB_Class = 92 の場合

SI_Startup 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT := 100	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
LostRetentive	BOOL	OB_Class = 100 の場合
LostRTC	BOOL	OB_Class = 100 の場合

SI_ProgIOAcessError 構造体

構造体エレメント	データタイプ	説明
SI_Format	USINT	• 16#FF = 情報なし • 16#FE = 最適化された開始情報
OB_Class	USINT	「情報なし」または「最適化された開始情報」の OB クラス
OB_Nr	UINT	DB 番号(1～32767)
BlockNr	UINT	OB_Class = 121、122 の場合
Reaction	USINT	OB_Class = 121、122 の場合
Fault_ID	BYTE	OB_Class = 121、122 の場合
BlockType	USINT	OB_Class = 121、122 の場合
Area	USINT	OB_Class = 121、122 の場合
DBNr	DB_ANY	OB_Class = 121、122 の場合
Csg_OBNr	OB_ANY	OB_Class = 121、122 の場合
Csg_Prio	USINT	OB_Class = 121、122 の場合
Width	USINT	OB_Class = 121、122 の場合

注記
SI_classic 構造体に指定された構造体エレメントの内容は、ブロックプロパティ「Default」で作成されたあらゆる OB の一時タグと同一です。

ただし、個々の OB の仮タグの名前とデータタイプが異なる場合があることに注意してください。また、各 OB の呼び出しインターフェースには、OB 要求の日付と時刻に関する追加情報が含まれます。

EV_CLASS 構造体エレメントのビット 4～7 には、イベントクラスが含まれます。 ここでは、次の値を使用できます。

- 1: 標準 OB からのイベントの開始
- 2: 同期エラー OB からのイベントの開始
- 3: 非同期エラー OB からのイベントの開始

PRIORITY 構造体エレメントは、現在の OB に属する優先度クラスを提供します。

これら 2 つのエレメント以外に、NUM も関連します。NUM には、現在の OB または最後に開始された起動 OB の番号が含まれます。

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
8080	現在の OB の開始情報が、指定されたユーザー定義データタイプと一致しません。
8081	現在の OB の開始情報が、指定されたシステムデータタイプと一致しません。
8082	開始された最後のスタートアップ OB の開始情報が、指定されたユーザー定義データタイプと一致しません。
8083	開始された最後のスタートアップ OB の開始情報が、指定されたシステムデータタイプと一致しません。

例

OB 80 は最後に呼び出された OB で、まだ処理が完了していません。OB 100 は最後に開始された起動 OB です。

次の表は、「RD_SINFO」命令の TOP_SI パラメータの構造体エレメントと、関連する OB 80 のローカルタグの割当てを示します。

TOP_SI 構造体エレメント	データタイプ	OB 80 - 関連するローカルタグ	データタイプ
EV_CLASS	BYTE	OB80_EV_CLASS	BYTE
EV_NUM	BYTE	OB80_FLT_ID	BYTE
PRIORITY	BYTE	OB80_PRIORITY	BYTE
NUM	BYTE	OB80_OB_NUMBR	BYTE
TYP2_3	BYTE	OB80_RESERVED_1	BYTE
TYP1	BYTE	OB80_RESERVED_2	BYTE
ZI1	WORD	OB80_ERROR_INFO	WORD
ZI2_3	DWORD	OB80_ERR_EV_CLASS	BYTE
		OB80_ERR_EV_NUM	BYTE
		OB80_OB_PRIORITY	BYTE
		OB80_OB_NUM	BYTE

次の表は、「RD_SINFO」命令の START_UP_SI パラメータの構造体エレメントと、関連する OB 100 のローカルタグの割当てを示します。

START_UP_SI 構造体エレメント	データタイプ	OB 100 - ローカルタグ	データタイプ
EV_CLASS	BYTE	OB100_EV_CLASS	BYTE
EV_NUM	BYTE	OB100_STRTUP	BYTE
PRIORITY	BYTE	OB100_PRIORITY	BYTE
NUM	BYTE	OB100_OB_NUMBR	BYTE
TYP2_3	BYTE	OB100_RESERVED_1	BYTE
TYP1	BYTE	OB100_RESERVED_2	BYTE
ZI1	WORD	OB100_STOP	WORD
ZI2_3	DWORD	OB100_STRT_INFO	DWORD

RT_INFO: ランタイム統計情報の読み出し

説明

命令「RT_INFO」を使用して、特定のオーガニゼーションブロック、通信、またはユーザープログラムのランタイムに関する統計情報を生成します。

MODE パラメータを使用して、出力する情報を選択します。

- MODE 1 ~ 3 は、OB パラメータで番号が指定された特定のオーガニゼーションブロックのランタイムに関するデータを返します。
- MODE 10 および 11 は、通信およびユーザープログラムで使用されたランタイムの割合を返します。
- MODE 20 および 21 は、通信およびユーザープログラムに割り当てられたランタイムのパーセンテージを返します。
- MODE 23 ~ 25 は、ユーザープログラムの最短、最長、および現在のサイクルタイムを出力します。
- MODE 30 ~ 32 は、ユーザープログラムのコンフィグレーションされた設定に関するデータを返します。

すべての測定は、CPU がスタートアップから RUN に移行するとき、再び開始されます。

パラメータ

以下の表に、「RT_INFO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	UINT	I、Q、M、D、L、または定数	MODE パラメータを使用して、読み出す情報を選択します(表「MODE パラメータ」を参照)。
OB	Input	OB_ANY	I、Q、M、D、L、または定数	OB パラメータを使用して、情報を読み出す OB を選択します。
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報(「 RET_VAL パラメータ」を参照)。
INFO	InOut	VARIANT	I、Q、M、D、L	読み出しデータの書き込み先の領域へのポインタ。 INFO で必要なデータタイプは、MODE パラメータに応じて異なります(表「MODE パラメータ」を参照)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ MODE

次の表に、MODE パラメータの値で返される情報を示します。

MODE (10進数)	説明	注	OB パラメータの値	INFO のデータタイプ	それ以降使用可能になる CPU バージョン
1	特定の OB のランタイム	ランタイム情報は、常に、指定された OB に関する情報です。高優先度の OB による割り込みまたは通信などのプロセスは含まれません。 この命令「RUNTIME」を使用して、プログラム全体のランタイムを測定できます。	DB 番号	LTIME	S7-1500 V1.5
2	特定の OB の最大ランタイム		DB 番号	LTIME	S7-1500 V1.5
3	特定の OB の最小ランタイム		DB 番号	LTIME	S7-1500 V1.5
10	高優先度の OB によって使用される合計ランタイムのパーセンテージ ユーザープログラムで使用されるすべての OB (サイクリックなプログラム OB より高優先度の OB)のランタイムが出力されます(ProgramCycle)。 これらには、スタートアップ OB を除くすべての OB タイプが含まれます。 CPU が、使用できる OB とそれらの優先度を識別します。 この情報は、プログラミングに関する基本的な章に含まれています。	-	関連なし	UINT	S7-1500 V1.5
11	通信によって使用される合計ランタイムのパーセンテージ ユーザープログラムの合計ランタイムに対する、通信プロセスのパーセンテージが出力されます。		関連なし	UINT	S7-1500 V1.5
20	評価が実行された最後のプログラムサイクルに基づく場合を除き、モード 10 と同一。	• サイクルタイムが高すぎる(1秒より長い)場合、計算を実行できません。 この命令は、値 65535 (0xFFFF)を出力します。 • サイクルタイムが低すぎる(1ミリ秒より短い)場合、計算を実行できません。 この場合、1ミリ秒以上の持続時間を持つ最後のサイクルが評価されます。 CPU のプロパティの最小サイクルタイムを割り当てると、サ	関連なし	UINT	S7-1500 V1.7
21	評価が実行された最後のプログラムサイクルに基づく場合を除き、モード 11 と同一。		関連なし	UINT	S7-1500 V1.7

		イクルタイムが 1 ms 未満に低下することを防止できます。			
23	最長サイクルタイム 最後に STOP から RUN に切り替わってからの最長のサイクルタイム。	この時間は、TIA ポータルの[サイクルタイム]ダイアログの[測定されたサイクルタイム]の値に対応しています。 [オンライン&診断|診断|サイクルタイム]を使用してダイアログを開くことができます。	関連なし	LTIME	S7-1500 V1.7
24	最短サイクルタイム 最後に STOP から RUN に切り替わってからの最短のサイクルタイム。		関連なし	LTIME	S7-1500 V1.7
25	現在/最後のサイクルタイム 最後のサイクルタイム。		関連なし	LTIME	S7-1500 V1.7
30	サイクルモニタリングタイム CPU プログラムの最大許容時間。 サイクルタイムがサイクルモニタ時間を超えた場合、CPU は STOP モードに切り替わるか、時刻エラー OB を呼び出します。	この時間は、TIA ポータルの[サイクルタイム]ダイアログの[割り当てられたサイクルタイム]の値に対応しています。 [オンライン&診断|診断|サイクルタイム]を使用してダイアログを開くことができます。	関連なし	LTIME	S7-1500 V1.5
31	ユーザープログラムの設定された最小サイクルタイムの出力。 最小サイクルタイムが CPU のプロパティに割り当てられている場合、最小サイクルタイムに達するまでオペレーティングシステムが新しいサイクルの開始を遅延します。		関連なし	LTIME	S7-1500 V1.5
32	設定された最大通信負荷のパーセンテージでの出力	通信に割り当てられたサイクル負荷のパーセンテージは、[通信負荷]の CPU プロパティで指定されます。	関連なし	UINT	S7-1500 V1.5

RET_VAL パラメータ

次の表に、RET_VAL パラメータの値の意味を示します。

エラーコード* (W#16#...)	説明
0	エラーは発生していません。

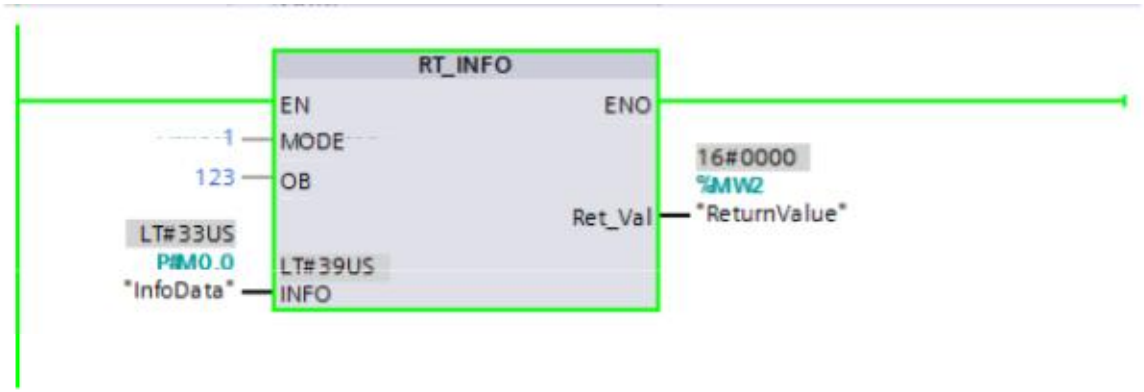
8080	MODE パラメータの値がサポートされていません。
8081	OB パラメータで選択されたオーガニゼーションブロックがユーザープログラムに存在しません。
8092	選択された MODE パラメータは、この CPU バージョンではサポートされていません。
80C3	不十分なリソース。 時間が経過してから、再び命令の呼び出しを試みます。
8452	INFO パラメータのデータタイプが不正です。 MODE パラメータに応じて、正しいデータタイプが選択されたかどうかをチェックします。

例

次の例では、サイクルオーガニゼーションブロックのランタイムを読み出します。

- タイプ「Program cycle」のブロックを新規作成します。 OB パラメータで DB 番号を指定します。
- MODE パラメータに「1」(特定 OB のランタイムを読み出す)を入力します。
- INFO パラメータ(この場合は「InfoData」)にデータタイプ LTIME のタグを指定します。
- Ret_Val パラメータにデータタイプ INT のタグを指定し、命令のエラーメッセージを出力します。

命令の呼び出し後、実際に測定されたランタイムが「InfoData」タグに書き込まれます。



LED : LED ステータス読み出し

説明

「LED」命令を使用して、特定のモジュール LED のステータス(たとえば「オン」または「オフ」)を読み出すことができます。

- LADDR パラメータを使用し、CPU またはインターフェースをアドレス指定します。
- LED パラメータを使用し、命令で現在のステータスを読み出すモジュール LED を選択します。
- RET_VAL パラメータは、命令が呼び出されたときに選択された LED のステータスを出力します。選択された LED に基づいて、特定のステータス情報のみを表示できます。 たとえば、色情報のみを持つ LED があります。 特定の LED で使用できるステータス情報については、それぞれのモジュールのハードウェア文書を参照してください。

パラメータ

以下の表に、「LED」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IO	I、Q、M、L、または定数	CPU またはインターフェースのハードウェア識別子 番号が自動的に割り当てられ、ハードウェアコンフィグレーションの CPU またはインターフェースのプロパティに保存されます(*CPU 名 + ~Common*)。
LED	Input	UINT	I、Q、M、D、L、または定数	LED の識別番号: • 1: STOP/RUN • 2: ERROR • 3: MAINT (保守) • 4: 冗長化 • 5: Link (緑) • 6: Rx/Tx (黄)
RET_VAL	Return	INT	I、Q、M、D、L	LED ステータス

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

RET_VAL パラメータ

RET_VAL	説明
0 ~ 9	LED ステータス: • 0 = LED が存在しない • 1 = 常にオフ

	• 2 = 色 1 (たとえば、LED STOP/RUN: 緑) 常にオン • 3 = 色 2 (たとえば、LED STOP/RUN: オレンジ色)常にオン • 4 = 色 1 が 2 Hz で点滅 • 5 = 色 2 が 2 Hz で点滅 • 6 = 色 1 および 2 が 2 Hz で交互に点滅 • 7 = LED が点灯、色 1 • 8 = LED が点灯、色 2 • 9 = LED が存在するが、ステータス情報が使用不可
8091	LADDR パラメータを使用してアドレス指定されたハードウェアコンポーネントが存在しません。
8092	LED をサポートしていないハードウェアコンポーネントが、LADDR パラメータを使用してアドレス指定されました。
8093	LED パラメータで指定されたハードウェア識別子が未定義です。
80Bx	LADDR パラメータで指定された CPU が「LED」命令をサポートしていません。

例

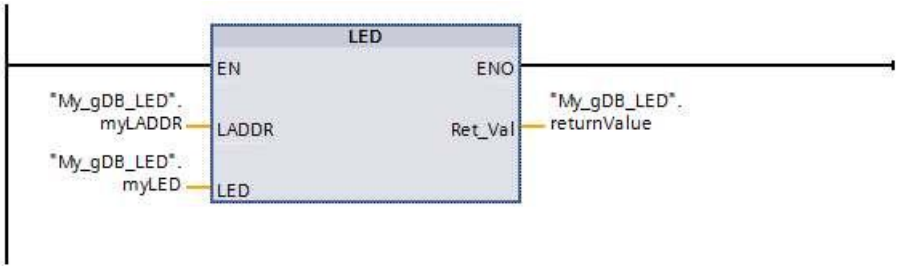
次の例では、CPU LED のステータスを読み出します。

グローバルデータブロックにデータを保存するために、3 つのタグを作成します。

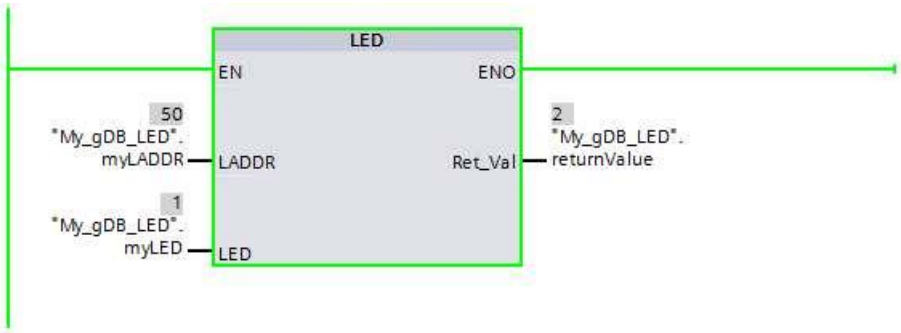
My_gDB_LED

	Name	Data type	Start value
1	Static		
2	myLADDR	HW_IO	50
3	myLED	UInt	1
4	returnValue	Int	0

この命令のパラメータを以下のように相互接続します。



CPU の HW 識別子は、パラメータ LADDR(「myLADDR」)を通じて「LED」命令に伝えられます。 モニタする CPU LED がパラメータ LED(「myLED」)に伝えられます。 CPU LED (STOP/RUN)のステータスが照会されます。 CPU が STOP から RUN モードになる場合、値「6」(緑色とオレンジ色の交互点灯)が出力パラメータ RET_VAL(「returnValue」)に表示されます。 次に、値「2」(緑色の常時点灯)が LED ステータス(「returnValue」)として表示されます。



注記: STOP/RUN LED の色は、次のような意味を持ちます。

色	意味
赤色	停止
緑色	実行
緑色とオレンジ色の交互点灯	CPU はロード中
赤と緑色の交互点灯	プログラムの処理が実行中

LED の色の意味に関する追加情報は、CPU のハードウェア説明を参照してください。

Get_IM_Data: 識別およびメンテナンスデータの読み出し 

説明

「Get_IM_Data」命令は、デバイスから識別およびメンテナンスデータ(I&M)を読み取ります。 LADDR パラメータを使用して、ハードウェア識別子によって I&M を読み取るデバイスを選択します。

IM_TYPE パラメータを使用して命令によって読み出すデータを選択します。

- IM_TYPE = 0: I&M 0 データ

  I&M 0 データは、デバイスのデバイス固有の基本情報であり、製造元 ID、注文番号、シリアル番号、ハードウェアおよびファームウェアのバージョンなどが含まれています。 I&M 0 データへの読み取りアクセスのみが可能です。 この情報は、TIA ポータルでは、デバイスの「オンライン&診断」ビューにも表示されます。

- IM_TYPE = 11: CPU のパラメータ割り当てデータからの I&M 1 データ

  I&M 1 データには、デバイスのファンクションの説明や、ロケーション ID、つまり、プラント内でのデバイスの指定方法に関する情報が含まれています。

- IM_TYPE = 12: CPU のパラメータ割り当てデータからの I&M 2 データ

  I&M 2 データには、据付日、つまり、プラント内にデバイスがいつ据え付けられたかに関する情報が含まれています。

- IM_TYPE = 13: CPU のパラメータ割り当てデータからの I&M 3 データ

  I&M 3 データには、設置済みデバイスに関する追加情報が含まれています。 追加情報はフリーテキストであり、希望通りに割り当てることができます。

読み出される I&M データは、DATA パラメータで定義されたアドレス領域に書き込まれます。

読み取りジョブの実行ステータスは、BUSY、DONE、ERROR 出力パラメータおよび STATUS 出力パラメータの 2 つの中央のバイトによって表示されます。

定義: 識別およびメンテナンスデータ(I&M)

識別および保守(I&M)データは、モジュールに保存された情報を参照します。これは、プラント設定のチェック、プラントでのハードウェア変更場所の特定、および工ラーの除去を支援します。

- 識別データ(I データ)は、読み取り専用のデバイスに関する静的情報です。
- 保守データ(M データ)は、インストール場所または日付など、プラント依存の情報を参照します。保守データは、設定中に作成され、モジュールに書き込まれます。

パラメータ

以下の表に、「Get_IM_Data」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	デバイスのハードウェア ID。番号は自動的に割り当てられ、デバイスのプロパティまたはハードウェアコンフィグレーションに保存されます。
IM_TYPE	Input	UINT	I、Q、M、D、L、または定数	識別およびメンテナンスデータ番号 次の値が返されます。

				• 0: I&M 0 データ • 11: CPU のパラメータ割り当てデータからの I&M 1 データ • 12: CPU のパラメータ割り当てデータからの I&M 2 データ • 13: CPU のパラメータ割り当てデータからの I&M 3 データ
DATA	InOut	VARIANT	I、Q、M、D、L	読み出された識別およびメンテナンスデータの保存のための領域(下記を参照)。
DONE	Output	BOOL	I、Q、M、D、L	命令が正常に実行されました。 I&M データは DATA パラメータに転送されました。
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 命令の実行が完了したか、またはまだ開始されていません。 • 1: 命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ。 このパラメータは、1 回の呼び出しの間のみ設定されます。 したがって、これを表示するには、STATUS を空きデータ領域にコピーしてください。

データタイプに関する詳細情報については、「有効なデータタイプの概要」を参照してください。

I&M 0 データの DATA パラメータ

配列(ARRAY of BYTE)または特殊なデータ構造を使用して、I&M 0 データを保存することができます。

- パラメータ DATA で配列(ARRAY of BYTE)をアドレス指定する場合、読み出された I&M 0 データは、DATA へのバイトシーケンスとしてコピーされます。 アドレス指定された配列が読み出されたデータよりも長い場合、不要なバイトがゼロで満たされます。
- I&M 0 データの場合、次の「IM0_Data」構造体を DATA パラメータで使用することもできます。

パラメータ		データタイプ	Byte (バイト)	説明
Manufacturer_ID		UINT	2	製造元 ID(シーメンスの場合は「42」など)
Order_ID		CHAR[20]	20	注文番号
Serial_Number		CHAR[16]	16	シリアル番号
Hardware_Revision		UNIT	2	ハードウェアリビジョン
Software_Revision		STRUCT	4	ファームウェアリビジョン
	Type	CHAR	1	-
	Functional	UNIT8	1	-

	Bugfix	UNIT8	1	-
	Internal	UINT8	1	-
Revision_Counter		UINT	2	リビジョンカウンタ
Profile_ID		UNIT	2	プロファイル
Profile_Specific_Type		UNIT	2	デバイスクラス
IM_Version		UNIT	2	I&M バージョン
I&M_Supported		UNIT	2	デバイス終端でサポートされている I&M データ(I&M 0-I&M 4)

異なるデータタイプが DATA パラメータで使用される場合、STATUS パラメータエラーコード 8093 を出力します。

I&M 1、I&M 2、および I&M 3 データの DATA パラメータ

文字列(STRING)、配列(ARRAY of CHAR/BYTE)またはデータ構造体(STRUCT)を使用して、I&M データを保存することができます。

- DATA パラメータで文字列(STRING データタイプ)をアドレス指定する場合、文字列の長さによって自動的に、読み取られた I&M データの長さが調節されます(最大 254 文字)。
- DATA パラメータでデータ構造体(ARRAY of CHAR/BYTE または STRUCT)をアドレス指定した場合、読み取られた I&M データが、使用されるデータタイプの個々のコンポーネントに書き込まれます。 アドレス指定されたデータ構造体が読み出されたデータよりも長い場合、残りのコンポーネントがゼロで満たされます。
- DATA パラメータで STRUCT データタイプを使用してデータ構造体を作成した場合には、最適化したブロックアクセスのないデータブロックを使用します(ブロックプロパティのカテゴリ「属性」を参照)。

DATA パラメータで STRING、ARRAY of BYTE/CHAR、または STRUCT 以外のデータタイプが使用された場合、STATUS パラメータによってエラーコード 8093 が生成されます。

注記

I&M データに関する追加情報

I&M データに関する追加情報は、たとえば、PROFIBUS & PROFINET International の Web サイト(リンク:http://www.profibus.com)を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生していません。
7000	進行中のジョブはありません。
7001	非同期的命令「Get_IM_Data」の最初の呼び出し。 命令の実行が未完了です(BUSY = 1、DONE = 0)。
7002	非同期的命令「Get_IM_Data」の追加呼び出し。 命令の実行が未完了です(BUSY = 1、DONE = 0)。
807F	インスタンスの最大 10 の並列実行を超えています。 • 10 を超える命令の並列呼び出しを避ける。 • これは一時的なエラーであるため、後で呼び出しを繰り返す。

8091	LADDR パラメータでアドレス指定されたデバイスが存在しません。
8092	LADDR が、I&M データの出力をサポートしていないデバイスをアドレス指定しています。
8093	DATA パラメータのデータタイプはサポートされません。
80B1	「Get_IM_Data」命令は、使用される CPU によってサポートされません。
80B2	IM_TYPE パラメータの値が無効であるか、選択された IM_TYPE が CPU またはアドレス指定されたデバイスによってサポートされません。
8752	DATA パラメータで指定されたメモリ領域が小さすぎて、すべての I&M データを保存できません。 読み取られた I&M データは、指定されたメモリ領域の最大長までのみが保存されます。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

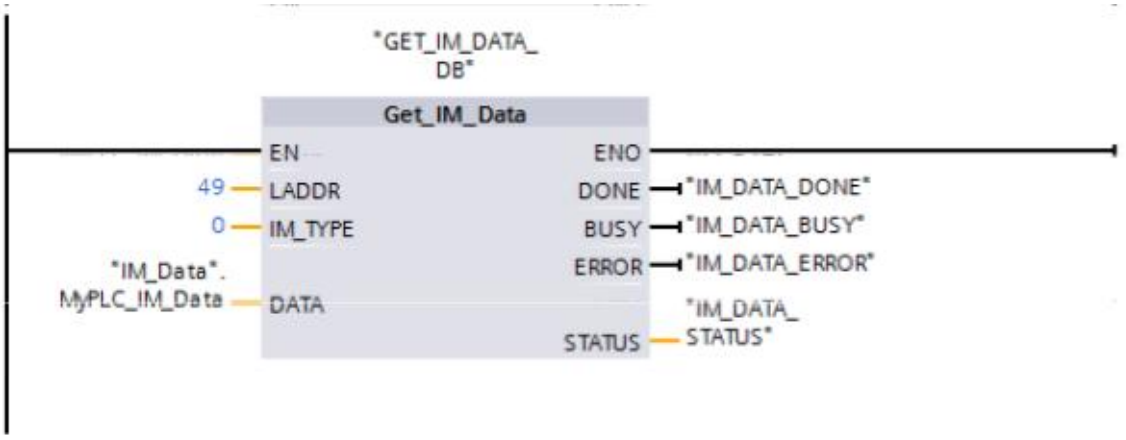
次の例では、S7-1500 CPU の IM0 データを読み出します。 IM0 データは、デバイスの基本情報であり、製造元 ID、注文番号、シリアル番号、ハードウェアおよびファームウェアのバージョンなどが含まれています。

IM0 データを保存するには、グローバルデータブロックに IM0_Data データタイプの構造体を作成します。 任意の名前を構造体(この場合は「MyPLC_IM_Data」)に割り当てることができます。

IM_Data

	Name	Datentyp	Startwert
1	Static		
2	MyPLC_IM_Data	IM0_Data	
3	Manufacturer_ID	UInt	0
4	Order_ID	String[20]	''
5	Serial_Number	String[16]	''
6	Hardware_Revision	UInt	0
7	Software_Revision	IM0_Version	
8	Type	Char	''
9	Functional	USInt	0
10	Bugfix	USInt	0
11	Internal	USInt	0
12	Revision_Counter	UInt	0
13	Profile_ID	UInt	0
14	Profile_Specific_Type	UInt	0
15	IM_Version	Word	16#0
16	IM_Supported	Word	16#0

LADDR パラメータに CPU のハードウェア識別子を入力します。 ハードウェア識別子は製品を一意に識別します。 CPU のハードウェア識別子を判別するには、PLC タグテーブルおよび[システム定数]タブを開きます。 次に、[名前]列で CPU を検索します。 関連値は、LADDR パラメータに入力するハードウェア識別子です。



命令が正常に実行されると、IM0 データがデータブロックに書き込まれます。

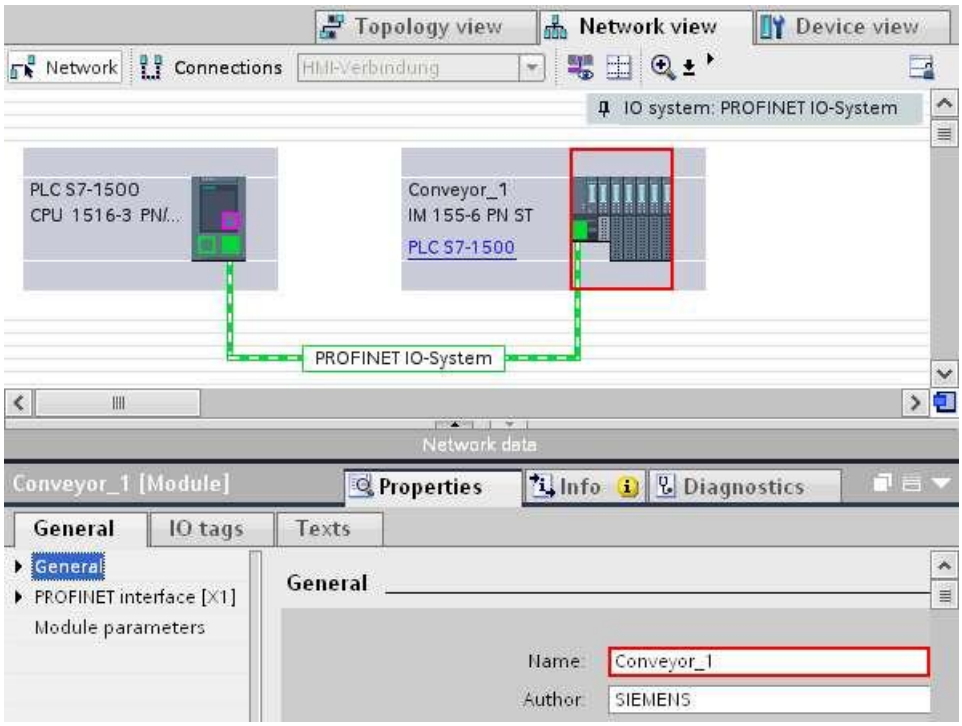
IM_Data

	Name	Datentyp	Startwert	Beobachtungswert
1	Static			
2	MyPLC_IM_Data	IM0_Data		
3	Manufacturer_ID	UInt	0	42
4	Order_ID	String[20]	''	'6ES7 511-1AK00-0AB0 '
5	Serial_Number	String[16]	''	'S C-DOS710132013'
6	Hardware_Revision	UInt	0	3
7	Software_Revision	IM0_Version		
8	Type	Char	' '	'V'
9	Functional	USInt	0	1
10	Bugfix	USInt	0	5
11	Internal	USInt	0	0
12	Revision_Counter	UInt	0	0
13	Profile_ID	UInt	0	0
14	Profile_Specific_Type	UInt	0	0
15	IM_Version	Word	16#0	16#0101
16	IM_Supported	Word	16#0	16#001E

GET_NAME: モジュールの名前の読み取り

説明

「GET_NAME」命令は、IO デバイスの名前を読み取ります。 この名前は、ネットワークビューおよび IO デバイスのプロパティに表示されます。



PROFINET IO システムのハードウェア識別子(LADDR パラメータで)および IO デバイスのデバイス番号(STATION_NR パラメータ)を使用することによって、IO デバイスを選択します。

命令が実行されると、IO デバイスの名前が DATA パラメータでアドレス指定された領域に書き込まれます。

読み出される名前は、IO デバイスのタイプに応じて異なります。

- DP スレーブまたは IO デバイスの場合、ヘッドモジュールの名前が出力されます。
- I スレーブまたは I デバイスの場合、インターフェースモジュールの名前が出力されます。
- HMI パネルの場合、インターフェースの名前が出力されます。
- PC ステーションの場合、インターフェースモジュールの名前が出力されます。
- GSD デバイスの場合、デバイスアクセスポイント(DAP)の名前が表示されます(インターフェースモジュールまたはヘッドモジュールの名前)。

名前の長さは、LEN パラメータに出力されます。 名前が DATA パラメータで指定された領域よりも長い場合、アドレス領域の最大長に対応するセクションのみが書き込まれます。

名前の最大長は、128 文字です。

注記

CPU 読み出しの名前(バージョン 1.1)

LADDR および STATION_NR パラメータのそれぞれに「0」を割り当てると、この命令は CPU の名前を出力します。

パラメータ

以下の表に、「GET_NAME」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IOSYS-TEM	I、Q、M、D、L、または定数	PROFINET IO システムのハードウェア識別子。 この番号は、システム定数、または、IO システムのプロパティから取得できます。
STA-TION_NR	Input	UINT	I、Q、M、D、L、または定数	IO デバイスのデバイス番号。 このデバイス番号は、ネットワークビューで[イーサーネットアドレス]の下の IO デバイスのプロパティから適用できます。
DATA	InOut	VARIANT	I、Q、M、D、L	モジュールの名前が書き込まれる領域へのポインタ。
DONE	Output	BOOL	I、Q、M、D、L	命令が正常に実行されました。 DA-TA パラメータでモジュールの名前が領域に転送されました。
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 命令の実行が完了しました。 • 1: 命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
LEN	Output	DINT	I、Q、M、D、L	IO デバイスの名前の長さ(文字数)。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS を空きデータ領域にコピーする必要があります。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

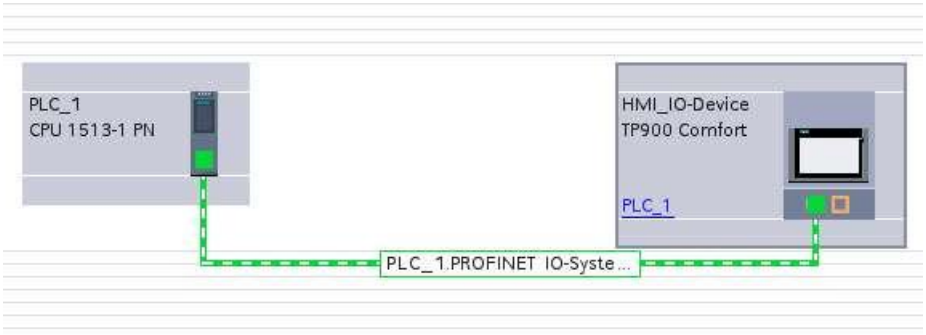
エラーコード* (W#16#...)	説明
0	エラーは発生していません。
7000	進行中のジョブはありません。
7001	非同期的命令「GET_NAME」の最初の呼び出し。 命令の実行が未完了です(BUSY = 1、DONE = 0)。
7002	非同期的命令「GET_NAME」の追加呼び出し。 命令の実行が未完了です(BUSY = 1、DONE = 0)。
8090	• LADDR 入力パラメータに無効な値が含まれています。 • LADDR パラメータで指定されたハードウェア識別子が、プロジェクトに存在しません。
8092	LADDR パラメータの値が、PROFINET IO システムをアドレス指定していません。
8093	命令が、DATA パラメータのデータタイプをサポートしていません。
8095	デバイス番号(STATION_NR パラメータ)が、選択されている PROFINET IO システム内に存在していないか、IO デバイスをアドレス指定していません。
80B1	使用している CPU がこの命令をサポートしていません。
80C3	一時的なリソースエラー: CPU は現在、可能な最大数の同時ブロック呼び出しを処理しています。 "GET_NAME」は、少なくとも 1 つのブロック呼び出しを終了しない限り、実行できません。
8852	DATA パラメータで指定された領域に、IO デバイスの名前すべてが収まりません。 名前が、可能な最大長まで書き込まれます。 名前全体を読み込むには、DATA パラメータにより長いデータ領域を使用します。 この領域には、LEN パラメータ以上の文字数が必要です。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例は、HMI パネルのステーション名を読み出す方法を示します。

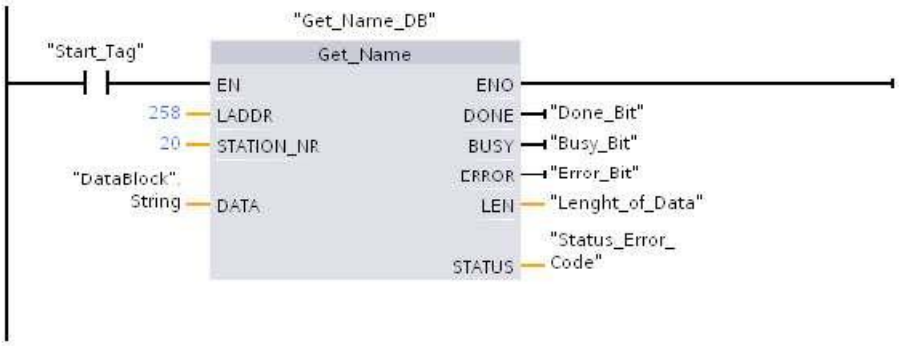
HMI パネルの設定

- ステーション名「HMI_IO-Device」を持つ HMI パネルはネットワークビューで作成され、CPU と同一の PROFINET IO システムに割り当てられます。
- HMI パネルの動作モード「IO デバイス」がハードウェアコンフィグレーションのプロパティで有効にされ、CPU が IO コントローラとして割り当てられました。
- デバイス番号「20」が、プロパティの「イーサネットアドレス」で割り当てられました。



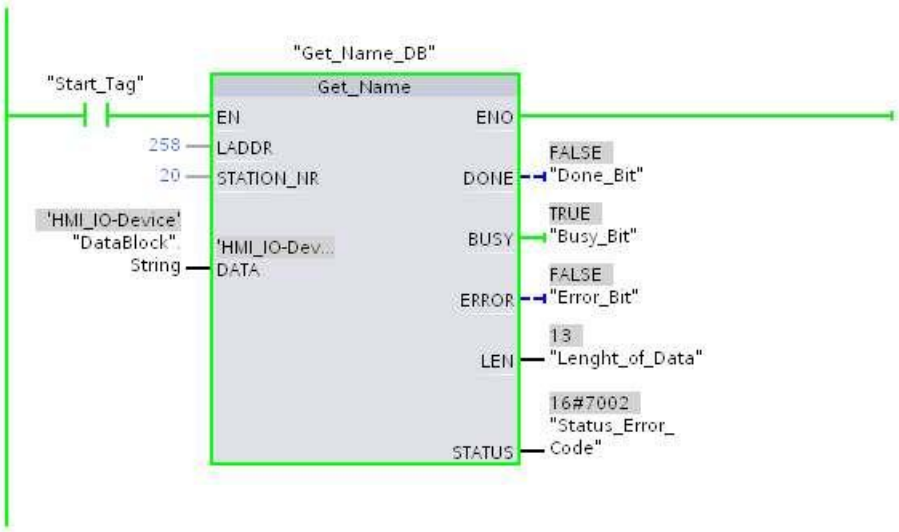
命令「GET_NAME」へのパラメータの割り当て:

- IO システムのハードウェア ID (258)が、パラメータ LADDR で指定されました。
- HMI パネルのデバイス番号(20)が、パラメータ STATION_NR で指定されました。
- タグが、パラメータ DATA のデータブロックのデータタイプ STRING と接続されました。
- 命令の出力パラメータとして、PLC タグ(メモリ領域、フラグ)が定義されました。



この命令の実行

- 命令を実行した後、HMI パネルのステーションの名前(HMI_IO-Device)が、パラメータ DATA に書き込まれます。
- この名前の文字数が、パラメータ LEN に出力されます(13)。



GetStationInfo: IO デバイスの情報の読み取り

説明

「GetStationInfo」命令を使用し、PROFINET IO デバイスの情報を読み取ることができます。 この命令は、下位の IO システムに配置された IO デバイス(CP/CM 経由で接続済み)の情報の読み取りも可能にします。

この IO デバイスは、LADDR パラメータでステーションのハードウェア識別子によってアドレス指定されます。 ハードウェア識別子は、「デバイス&ネットワーク」ビューでステーションプロパティに表示されます。

MODE パラメータを使用し、読み取る情報を選択します。

DATA パラメータで、読み取られたアドレスデータを書き込む行うデータ領域を指定します。 IP アドレスの保存には、「IF_CONF_v4」構造体を使用します。 MAC アドレスの保存には、「IF_CONF_MAC」構造体を使用します。

REQ 制御パラメータを使用して、アドレスデータの読み出しを有効にします。 これには、アクセス可能な IO デバイスが必要です。

読み取りジョブの実行ステータスは、BUSY、DONE、ERROR 出力パラメータおよび STATUS 出力パラメータによって表示されます。

注記

ステーションのハードウェア識別子のみを使用する IO デバイスのアドレス指定

ステーション、IO デバイス、および PROFINET インターフェースには、独自のハードウェア識別子があります。 「GetStationInfo」命令には、ステーションのハードウェア識別子のみを使用します。

たとえば、PROFINET が LADDR パラメータによってアドレス指定された場合、アドレスデータは読み取られず、エラーコード 8092 が生成されます。

集中構成で統合 PROFINET インターフェースまたは CM/CP のアドレスデータを読み取るには、「RDREC」命令を使用します。

パラメータ

以下の表に、「GetStationInfo」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	制御パラメータ要求 REQ = "1"で情報の読み取りを有効にします。
LADDR	Input	HW_DEVICE	I、Q、M、D、L、または定数	IO デバイスのステーションのハードウェア識別子 この数は、ネットワークビューのステーションのプロパティから、または標準タグテーブルの[システム定数]タブから取得することができます。

DETAIL	Input	HW_SUB-MODULE	I、Q、M、D、L、または定数	DETAIL パラメータは使用されません。 パラメータを未接続のままにします。
MODE	Input	UNIT	I、Q、M、D、L、または定数	読み出されるアドレスデータの選択: • MODE = 1: IPv4 に従ったアドレスパラメータ(ファームウェアバージョン V1.1 以降の S7-1500 CPU) • MODE = 2: MAC アドレス(ファームウェアバージョン V1.5 以降の S7-1500 CPU)
DATA	InOut	VARIANT	D、L	IO デバイスのアドレスデータが書き込まれる領域へのポインタ。 MODE = 1 の場合は「IF_CONF_v4」構造体を使用し、MODE = 2 の場合は「IF_CONF_MAC」構造体を使用します。
DONE	Output	BOOL	I、Q、M、D、L	命令が正常に実行されました。 アドレスデータは DATA パラメータに転送されました。
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 命令の実行が完了しました。 • 1: 命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ。 このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS を空きデータ領域にコピーする必要があります。

データタイプに関する詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ DATA

- DATA パラメータの「IF_CONF_v4」構造体を使用して、IPv4 に従ってアドレスパラメータを保存します。

Byte (バイト)	パラメータ	データタイプ	開始値	説明
0 … 1	Id	UINT	30	「IF_CONF_v4」構造体の ID。
2 … 3	Length	UNIT	18	BYTE で読み取られたデータの長さ。
4 ... 5	Mode	UNIT	0	「GetStationInfo」命令には関連なし(「0」では左)。

6 ... 9	InterfaceAddress	ARRAY [1..4] of BYTE	-	IP_V4 形式での IO デバイスの IP アドレス、たとえば 192.168.3.10 の場合: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 10
10 … 13	SubnetMask	ARRAY [1..4] of BYTE	-	IP_V4 形式での IO デバイスのサブネットマスク、たとえば 255.255.255.0 の場合: • addr[1] = 255 • addr[2] = 255 • addr[3] = 255 • addr[4] = 0
14 ... 17	DefaultRouter	ARRAY [1..4] of BYTE	-	IP_V4 形式でのルータの IP アドレス、たとえば 192.168.3.1 の場合: • addr[1] = 192 • addr[2] = 168 • addr[3] = 3 • addr[4] = 1

- パラメータ DATA の「IF_CONF_MAC」構造体を使用して、MAC アドレスを保存します。

Byte (バイト)	パラメータ	データタイプ	開始値	説明
0 … 1	Id	UINT	3	「IF_CONF_MAC」構造体の ID。
2 … 3	Length	UNIT	12	BYTE で読み取られたデータの長さ。
4 ... 5	Mode	UNIT	0	「GetStationInfo」命令には関連なし(「0」では左)。
6 ... 11	MACAddress	ARRAY [1..6] of BYTE	-	IO デバイスの MAC アドレス、たとえば、08-00-06-12-34-56 の場合 • Mac[1] = 8 • Mac[2] = 0 • Mac[3] = 6 • Mac[4] = 12 • Mac[5] = 34 • Mac[6] = 56

STATUS パラメータ

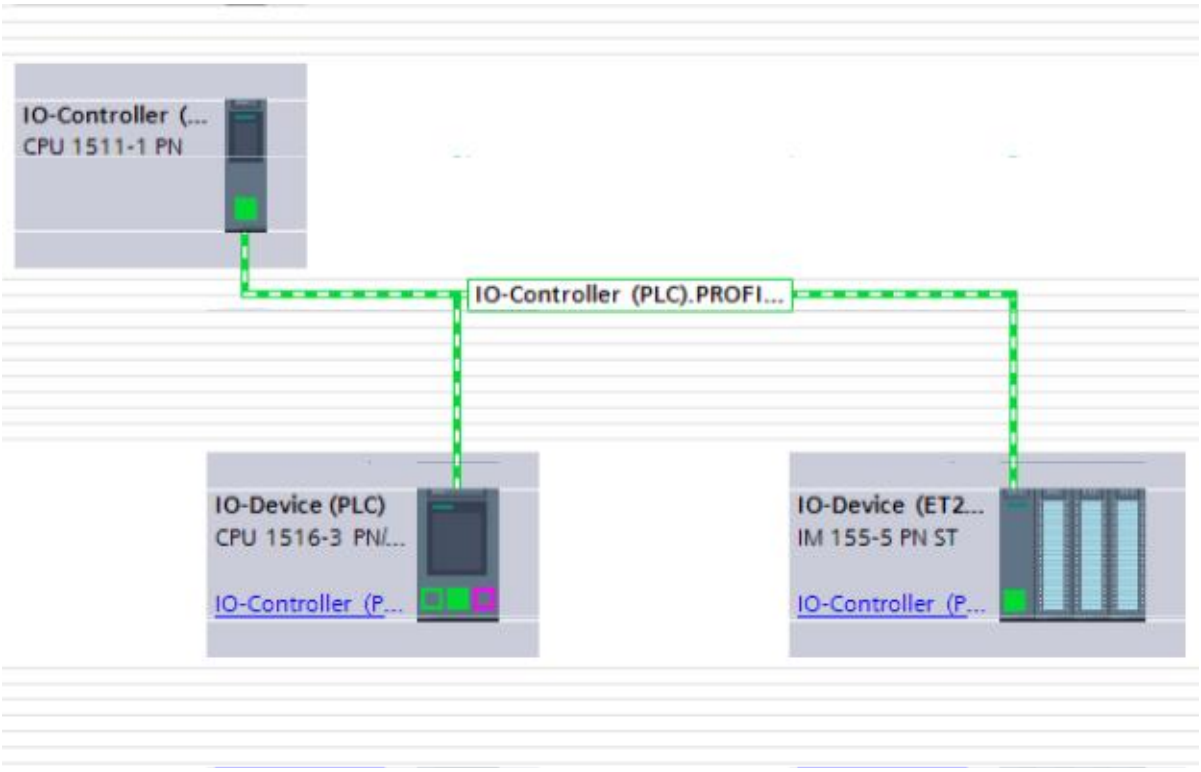
エラーコード* (W#16#...)	説明
0	エラーは発生していません。

7000	進行中のジョブはありません。
7001	非同期的命令「GetStationInfo」の最初の呼び出し。 命令の実行が未完了です(BUSY = 1、DONE = 0)。
7002	非同期的命令「GetStationInfo」の追加呼び出し。 命令の実行が未完了です(BUSY = 1、DONE = 0)。
8080	MODE パラメータの値がサポートされていません。
8090	LADDR パラメータで指定されたハードウェア識別子が未設定です。
8092	LADDR パラメータが、PROFINET IO デバイスをアドレス指定していません。
8093	DATA パラメータの無効なデータタイプ。
80A0	要求された情報を読み出せません。
80C0	アドレス指定された IO デバイスに到達できません。
80C3	「GetStationInfo」命令の同時呼び出しの最大数(10 インスタンス)に達しています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

下記では、„GetStationInfo“命令を使用して、IO デバイスの IP アドレスデータを読み出し、この情報をデータブロックに書き込みます。 IP アドレスデータには、IP アドレス、サブネットマスク、およびルータ(使用する場合)のアドレスデータが含まれます。

命令は IO コントローラ上で実行され、下位レベルの IO デバイスの IP アドレス情報(この例では ET200MP)を読み出します。



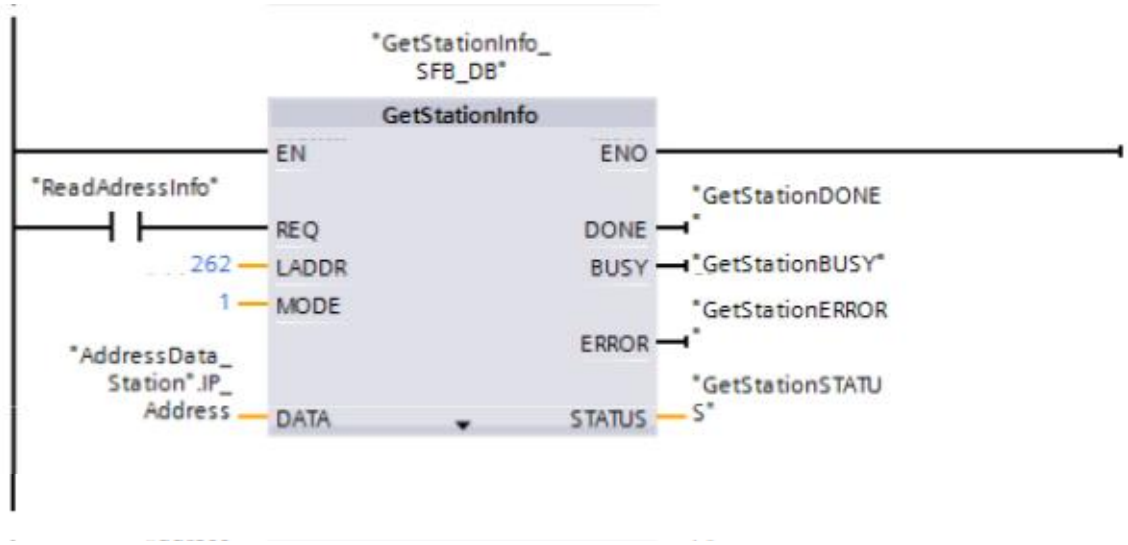
アドレスデータを保存するには、グローバルデータブロックに IF_CONF_v4 データタイプの構造体を作成します。 任意の名前を構造体(この場合は「IP_Address」)に割り当てることができます。

AddressData_Station

	Name	Datentyp	Startwert	Beobachtungswert	Kommentar
1	Static				
2	IP_Address	IF_CONF_v4			
3	Id	UInt	30		
4	Length	UInt	18		
5	Mode	UInt	0		
6	InterfaceAddress	IP_V4			
7	ADDR	Array[1..4] of Byte			IPv4 address
8	ADDR[1]	Byte	0		
9	ADDR[2]	Byte	0		
10	ADDR[3]	Byte	0		
11	ADDR[4]	Byte	0		
12	SubnetMask	IP_V4			
13	ADDR	Array[1..4] of Byte			IPv4 address
14	ADDR[1]	Byte	0		
15	ADDR[2]	Byte	0		
16	ADDR[3]	Byte	0		
17	ADDR[4]	Byte	0		
18	DefaultRouter	IP_V4			
19	ADDR	Array[1..4] of Byte			IPv4 address
20	ADDR[1]	Byte	0		
21	ADDR[2]	Byte	0		
22	ADDR[3]	Byte	0		
23	ADDR[4]	Byte	0		

次に GetStationInfo 命令を呼び出します。

- DATA パラメータで IF_CONF_v4 構造体を使用します。
- LADDR パラメータに IO デバイスのハードウェア識別子を入力します。 ハードウェア識別子は製品を一意に識別します。 IO デバイスのハードウェア識別子を判別するには、PLC タグテーブルおよび[システム定数]タブを開きます。 次に、名前列でデバイスを検索し、データタイプ列で „Hw_Device“を検索します。 関連値は、LADDR パラメータに入力するハードウェア識別子です。
- MODE パラメータで「1」を選択します(IPv4 に従ったアドレスパラメータの読み取り)。



REQ= 1 (TRUE)を使用して、アドレスデータの読み出しを開始します。 命令が正常に実行されると、IP アドレスデータがデータブロックに書き込まれます。

DeviceStates: IO システムのモジュールステータス情報の読み取り

説明

IO システムのすべてのモジュールに対して特定のステータス情報を照会するには、命令「DeviceStates」を使用します。つまり、

- PROFINET IO システムのすべての IO デバイス
- または DP マスタシステムのすべての DP スレーブに対してです。

出力されるブール値は、選択されたステータスが適用されるモジュールを示します。 たとえば、PROFINET IO システムで現在無効になっている IO デバイスを読み取ることができます。

読み取られるステータス情報が 1 つ以上の IO デバイスまたは DP スレーブに適用されるかどうかに関する情報も表示されます。

この命令は、サイクリック OB でも割り込み OB でも呼び出すことができます(OB82 - 診断割り込みなど)。

パラメータ

以下の表に、「DeviceStates」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IOSYSTEM	I、Q、M、L、または定数	PROFINET IO または DP マスタシステムのハードウェア識別子(以下の説明を参照)
MODE	Input	UINT	I、Q、M、D、L、または定数	読み出されるステータス情報の選択(以下の説明を参照)
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス(以下の説明を参照)
STATE	InOut	VARIANT	I、Q、M、D、L	IO デバイスまたは DP スレーブのステータス用バッファ(以下の説明を参照)

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ LADDR

ハードウェア識別子によって、LADDR パラメータで PROFINET IO または DP マスタシステムを選択します。

以下のハードウェア識別子が使用できます。

- ネットワークビューで、PROFINET IO または DP マスタシステムのいずれかのプロパティに入力します。
- または、PLC タグテーブルで、リストされたデータタイプ HW_IOSYSTEM のシステム定数に入力します。

パラメータ MODE

MODE パラメータを使用して、ステータス情報を読み出します。 PROFINET IO または DP マスタシステム全体に対して、以下のステータス情報項目の 1 つを読み出すことができます。

- 1: IO デバイス/DP スレーブが設定されます
- 2: IO デバイス/DP スレーブに障害があります
- 3: IO デバイス/DP スレーブが無効です
- 4: IO デバイス/DP スレーブが存在します
- 5: 問題が発生した IO デバイス/DP スレーブ。 たとえば、
  - メンテナンス要求または推奨
  - アクセス不可能
  - 使用不可
  - エラーが発生しました。

パラメータ STATE

STATE パラメータを使用し、MODE パラメータで選択した IO デバイス/DP スレーブのステータスが出力されます。

MODE を使用して選択されたステータスが IO デバイス/DP スレーブに適用された場合、STATE パラメータで以下のビットが「1」にセットされます。

- ビット 0 = 1: グループ表示。 少なくとも 1 つの IO デバイス/DP スレーブのビット n が、「1」に設定されました。
- ビット n = 1: MODE で選択されたステータスが、IO デバイス/DP スレーブに適用されます。
  - PROFINET IO システムの場合、ビット n はそれぞれの IO デバイスのデバイス番号に一致します(デバイスビューおよびネットワークビューの PROFINET インターフェースのプロパティを参照)
  - PROFIBUS DP システムの場合、ビット n が DP スレーブの PROFIBUS アドレスに一致します(デバイスビューおよびネットワークビューの DP スレーブのプロパティを参照)

データタイプ:として、「BOOL」または「Array of BOOL」を使用します。

- ステータス情報のグループ表示のビットのみを表示するには、STATE パラメータでデータタイプ BOOL を使用することができます。
- すべての IO デバイス/DP スレーブのステータス情報を出力するには、以下の長さの Array of BOOL を使用します。
  - PROFINET IO システムの場合: 1024 ビット
  - DP マスタシステムの場合: 128 ビット

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生していません。
8091	LADDR パラメータのハードウェア識別子が存在しません。 プロジェクトに LADDR の値が存在するかどうかを(システム定数などで)チェックします。
8092	LADDR が、PROFINET IO または DP マスタシステムをアドレス指定していません。
8093	STATE パラメータの無効なデータタイプ。
80B1	命令「DeviceStates」は、CPU によってサポートされていません。

80B2	選択された MODE パラメータが、LADDR パラメータで指定された IO システムに使用される CPU によってサポートされません。
8452	完全なステータス情報が、STATE パラメータに設定されたタグに収まりません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例 - PROFINET IO マスタシステムでの IO デバイスの存在の読み出し

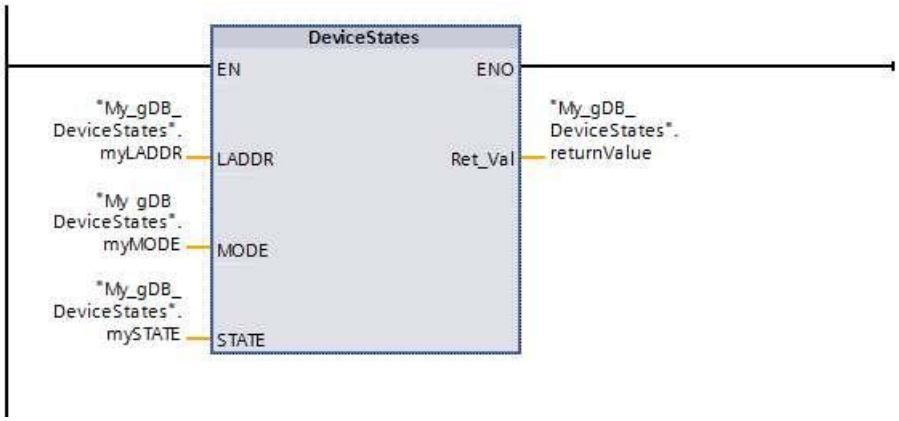
次の例では、IO システムに IO デバイスが存在するかを照会します。 IO システムは S7-1500 シリーズの 2 つの CPU で構成されます。 「PLC_14」CPU には「DeviceStates」命令を含む、プログラムが含まれます。 「PLC_13」CPU は、IO デバイスとして構成されます。

「PLC_14」CPU で次の手順を行います。 グローバルデータブロックにデータを保存するために、3 つのタグと 1 つの「mySTATE」構造体(Array of BOOL データタイプ)を作成します。

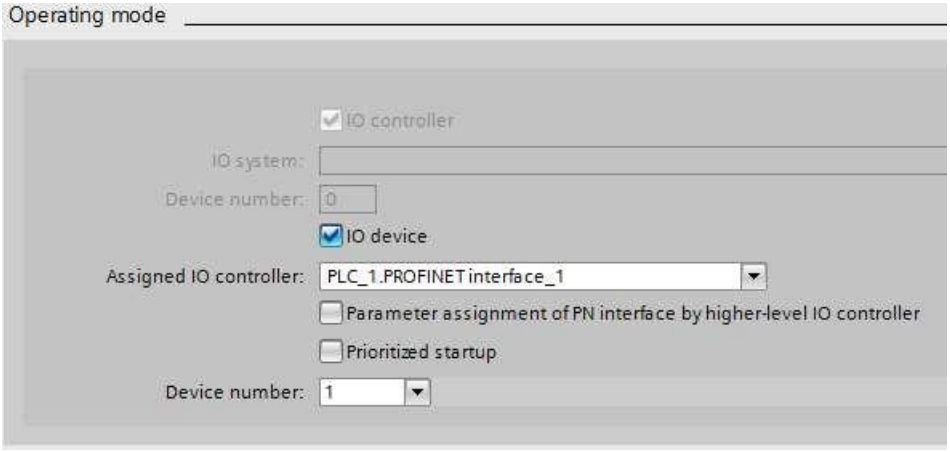
My_gDB_DeviceStates

	Name	Data type	Start value
1	Static		
2	myLADDR	HW_IOSYSTEM	258
3	myMODE	UInt	4
4	returnValue	Int	0
5	mySTATE	Array[0..1023] ...	
6	mySTATE[0]	Bool	false
7	mySTATE[1]	Bool	false
8	mySTATE[2]	Bool	false
9	mySTATE[3]	Bool	false
10	mySTATE[4]	Bool	false
11	mySTATE[5]	Bool	false
12	mySTATE[6]	Bool	false

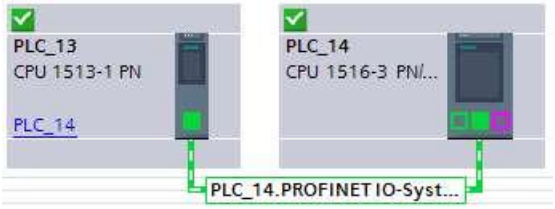
「PLC_14」CPU で次の手順を行います。 命令はサイクリック OB で呼び出されます。命令のパラメータを次のように相互接続します。



「PLC_13」CPU で次の手順を行います。 CPU のプロパティを使用して、この CPU「PLC_13」を IO デバイスとしてセットアップします。 IO デバイスはデバイス番号 1 を受け取ります。



IO システムがネットワークビューに表示されます。



「PLC_14」CPU で次の手順を行います。 IO システムの HW 識別子は、パラメータ LADDR(「myLADDR」)を通じて「DeviceStates」命令に伝えられます。 パラメータ MODE(「myMODE」)の値「4」に従って、IO デバイスで IO システムを検索します。

パラメータ STATE(「mySTATE」)で、(パラメータ MODE の値に基づき)IO デバイスの存在が出力されます。 ビット 0 はグループ値として機能し、IO デバイスが存在することを示します。 ビット 1 は、デバイス番号 1 の IO デバイスが存在することを示します。

出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われたことを示します。

My_gDB_DeviceStates

	Name	Data type	Start value	Monitor value
1	Static			
2	myLADDR	HW_IOSYSTEM	258	16#0102
3	myMODE	UInt	4	4
4	returnValue	Int	0	0
5	mySTATE	Array[0..1023] ...		
6	mySTATE[0]	Bool	false	TRUE
7	mySTATE[1]	Bool	false	TRUE
8	mySTATE[2]	Bool	false	FALSE
9	mySTATE[3]	Bool	false	FALSE
10	mySTATE[4]	Bool	false	FALSE
11	mySTATE[5]	Bool	false	FALSE
12	mySTATE[6]	Bool	false	FALSE

例 - PROFINET IO マスタシステムの障害ステーションの読み取り

PROFINET IO システムは、デバイス番号 1、2、3、および 4 の 4 つの IO デバイスで構成されます。番号 2 の IO デバイスは障害です。

命令「DeviceStates」は、MODE = 2 の PROFINET IO システムに対して実行されます(障害/障害でない)。

以下のビットが、STATE パラメータにセットされます。

- ビット 0 = 1: 少なくとも 1 つの IO デバイスに故障が存在します。
- ビット 1 = 0: デバイス番号 1 の IO デバイスは障害ではありません。
- ビット 2 = 1: デバイス番号 2 の IO デバイスは障害です。
- ビット 3 = 0: デバイス番号 3 の IO デバイスは障害ではありません。
- ビット 4 = 0: デバイス番号 4 の IO デバイスは障害ではありません。
- ビット 5 = 0: 対象外
- ビット 6 = 0: 対象外
- ...

例 - PROFIBUS DP マスタシステムの障害ステーションの読み取り

DP マスタシステムは、PROFIBUS アドレス 3、4、5、および 6 の 4 つの DP スレーブで構成されます。アドレス 4 の DP スレーブは障害です。

命令「DeviceStates」は、MODE = 2 の DP マスタシステムに対して実行されます(障害/障害でない)。

以下のビットが、STATE パラメータにセットされます。

- ビット 0 = 1: 少なくとも 1 つの DP スレーブに故障が存在します。
- ビット 1 = 0: 対象外
- ビット 2 = 0: 対象外
- ビット 3 = 0: アドレス 3 の DP スレーブは障害ではありません。
- ビット 4 = 1: アドレス 4 の DP スレーブに障害があります。
- ビット 5 = 0: アドレス 5 の DP スレーブは障害ではありません。
- ビット 6 = 0: アドレス 6 の DP スレーブは障害ではありません。
- ビット 7 = 0: 対象外
- ビット 8 = 0: 対象外
- ...

ModuleStates: モジュールのモジュールステータス情報読み出し

説明

「ModuleStates」命令を使用し、PROFINET IO デバイスまたは PROFIBUS DP スレーブのモジュールのステータス情報を読み取ることができます。

出力されるブール値は、選択されたステータスが適用されるモジュールを示します。 たとえば、PROFINET IO デバイスで現在無効になっているモジュールを読み取ることができます。

読み取られるステータス情報が 1 つ以上のモジュールに適用されるかどうかに関する情報も表示されます。

この命令は、サイクリック OB でも割り込み OB でも呼び出すことができます(OB82 - 診断割り込みなど)。

パラメータ

以下の表に、「ModuleStates」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_DEVICE	I、Q、M、D、L、または定数	ステーションのハードウェア識別子(以下の説明を参照)
MODE	Input	UINT	I、Q、M、D、L、または定数	読み出されるモジュールステータス情報の選択(以下の説明を参照)
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス(以下の説明を参照)
STATE	InOut	VARIANT	I、Q、M、D、L	モジュールステータス用バッファ(以下の説明を参照)

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ LADDR

ステーションのハードウェア識別子によって、LADDR パラメータで IO デバイスまたは DP スレーブを選択します。

以下のハードウェア識別子が使用できます。

- ネットワークビューで、IO デバイスステーションまたは DP スレーブステーションのいずれかのプロパティに入力します。
- または、PLC タグテーブルで、リストされたデータタイプ HW_DEVICE(IO デバイス用)またはデータタイプ HW_DPSLAVE(DP スレーブ用)のシステム定数に入力します。

パラメータ MODE

MODE パラメータを使用して、ステータス情報を読み出します。 モジュールに対して、以下のステータス情報項目の 1 つを読み取ることができます。

- 1: モジュールが設定されます
- 2: モジュールに障害があります

- 3: モジュールが無効です
- 4: モジュールが存在します
- 5: モジュールに問題があります。 たとえば、
  - メンテナンス要求または推奨
  - アクセス不可能
  - 使用不可
  - エラーが発生しました。

パラメータ STATE

STATE パラメータは、MODE パラメータで選択されたモジュールのステータスを出力します。

MODE を使用して選択されたステータスがモジュールに適用された場合、以下のビットが「1」にセットされます。

- ビット 0 = 1: グループ表示。 少なくとも 1 つのモジュールのビット n が、「1」に設定されました。
- ビット n = 1: MODE で選択されたステータスが、スロット n-1 のモジュールに適用されます(例: ビット 3 = スロット 2)。

データタイプ:として、「BOOL」または「Array of BOOL」を使用します。

- ステータス情報のグループ表示のビットのみを表示するには、STATE パラメータでデータタイプ BOOL を使用することができます。
- すべてのモジュールのステータス情報を出力するには、長さが 128 ビットの Array of BOOL を使用します。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生していません。
8091	LADDR パラメータのハードウェア識別子が存在しません。 プロジェクトに LADDR の値が存在するかどうかを(システム定数などで)チェックします。
8092	LADDR が、IO デバイスまたは DP スレーブをアドレス指定していません。
8093	STATE パラメータの無効なデータタイプ。
80B1	命令「ModuleStates」は、CPU によってサポートされていません。
80B2	選択された MODE パラメータが、LADDR パラメータで IO デバイス/DP スレーブに使用される CPU によってサポートされません。
8452	完全なステータス情報が、STATE パラメータに設定されたタグに収まりません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

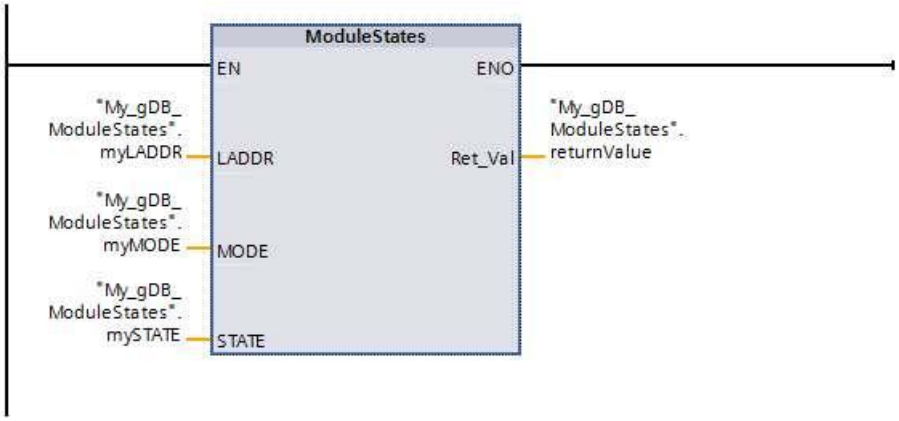
次の例では、PROFINET IO デバイスのモジュールの存在を照会します。 IO システムは S7-1500 シリーズの 2 つの CPU で構成されます。 「PLC_14」CPU には「ModuleStates」命令を含む、プログラムが含まれます。 「PLC_13」CPU は、IO デバイスとして構成されます。

「PLC_14」CPU で次の手順を行います。 グローバルデータブロックにデータを保存するために、3 つのタグと 1 つの「mySTATE」構造体(Array of BOOL データタイプ)を作成します。

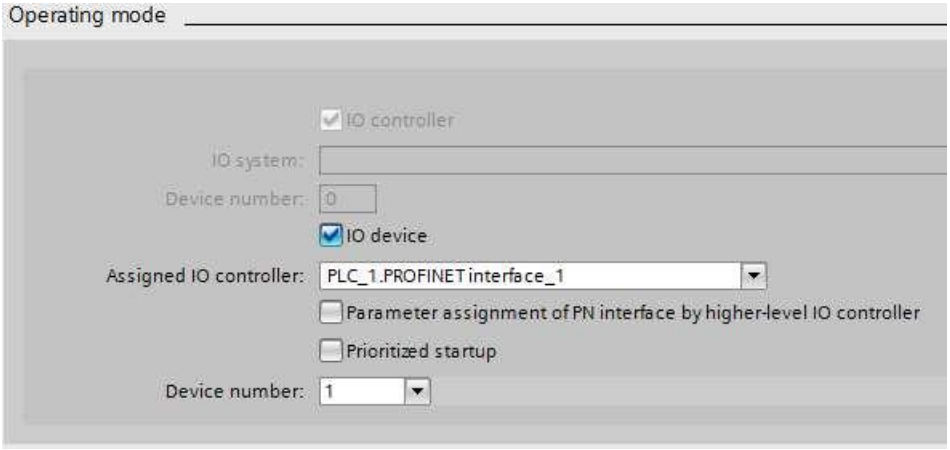
My_gDB_ModuleStates

	Name	Data type	Start value
1	Static		
2	myLADDR	HW_DEVICE	260
3	myMODE	UInt	4
4	returnValue	Int	0
5	mySTATE	Array[0..127] of Bool	
6	mySTATE[0]	Bool	false
7	mySTATE[1]	Bool	false
8	mySTATE[2]	Bool	false
9	mySTATE[3]	Bool	false
10	mySTATE[4]	Bool	false
11	mySTATE[5]	Bool	false
12	mySTATE[6]	Bool	false

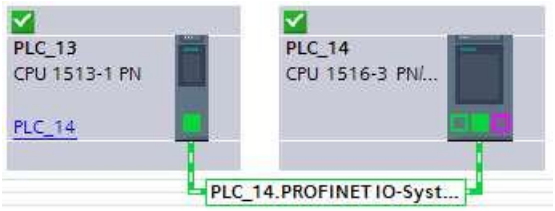
「PLC_14」CPU で次の手順を行います。 命令はサイクリック OB で呼び出されます。命令のパラメータを次のように相互接続します。



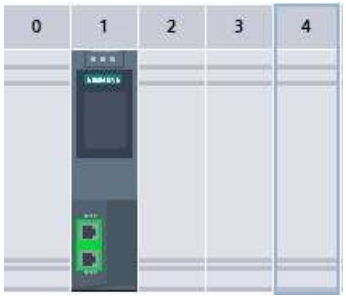
「PLC_13」CPU で次の手順を行います。 CPU のプロパティを使用して、この CPU「PLC_13」を IO デバイスとしてセットアップします。



IO システムがネットワークビューに表示されます。



「PLC_14」CPU で次の手順を行います。 モジュールが IO デバイスのスロット 1 にあります。



「PLC_14」CPU で次の手順を行います。 IO デバイスの HW 識別子は、パラメータ LADDR(「myLADDR」)を通じて「ModuleStates」命令に伝えられます。 パラメータ MODE(「myMODE」)の値「4」に従って、モジュールで IO デバイスを検索します。

パラメータ STATE(「mySTATE」)で、(パラメータ MODE の値に基づき)モジュールの存在が出力されます。 ビット 0 はグループ値として機能し、モジュールが存在することを示します。 ビット 2 はスロット 1 にモジュールが存在することを示します。

出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われたことを示します。

My_gDB_ModuleStates

	Name	Data type	Start value	Monitor value
1	Static			
2	myLADDR	HW_DEVICE	260	16#0104
3	myMODE	UInt	4	4
4	returnValue	Int	0	0
5	mySTATE	Array[0..127] of Bool		
6	mySTATE[0]	Bool	false	TRUE
7	mySTATE[1]	Bool	false	FALSE
8	mySTATE[2]	Bool	false	TRUE
9	mySTATE[3]	Bool	false	FALSE
10	mySTATE[4]	Bool	false	FALSE
11	mySTATE[5]	Bool	false	FALSE
12	mySTATE[6]	Bool	false	FALSE

例

IO デバイスは、スロット 1～4 に 4 つのモジュールが含まれています。スロット番号 2 のモジュールが障害です。

命令「ModuleStates」は、MODE = 2 の IO デバイスに対して実行されます(障害/障害でない)。

以下のビットが、STATE パラメータにセットされます。

- ビット 0 = 1: 少なくとも 1 つのモジュールに故障が存在します。
- ビット 1 = 0: スロット番号 0 (IO デバイスによって使用)
- ビット 2 = 0: スロット番号 1 のモジュールは障害ではありません。

- ビット 3 = 1: スロット番号 2 のモジュールに障害があります。
- ビット 4 = 0: スロット番号 3 のモジュールは障害ではありません。
- ビット 5 = 0: スロット番号 4 のモジュールは障害ではありません。
- ビット 6 = 0: 対象外
- ビット 7 = 0: 対象外

GEN_DIAG: 診断情報の生成

説明

「GEN_DIAG」命令は、他のメーカーのハードウェアコンポーネントの TIA ポータル診断で使用可能な診断情報を生成します。 この命令を使用するには、メーカーが提供する GSD(GSDL/GSDML)ファイルを先にインストールする必要があります。

この命令は、すべての診断イベント(メンテナンス用診断イベントを含む)を生成します。

- LADDR パラメータを使用して、診断イベントを生成するハードウェアコンポーネントを選択します。
- MODE パラメータを使用して、イベントが送信イベントなのか、または受信イベントなのかを指定します。
- DiagEvent パラメータを使用して、DiagnosticDetail 構造体内で診断イベントを定義します。 DiagEvent パラメータでタグを定義すると、ブロックのローカルインターフェース内で自動的に構造体が作成されます。

診断情報は同期的に提供されます。 診断情報の転送とアラーム出力は非同期です。

通知
フェールセーフ固有のエラーメッセージは無効です DiagEvent パラメータでフェールセーフ固有ン診断情報を定義すると、命令がこの情報をチェックし、エラーコード 80A1 を出力します。

パラメータ

次の表に、「GEN_DIAG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_ANY	I、Q、M、D、L、または定数	ハードウェアコンポーネントの識別番号
MODE	Input	UINT	I、Q、M、D、L、または定数	受信/送信情報の選択: • 1: 指定された診断イベントは受信イベントです。 • 2: 指定された診断イベントは送信イベントです。 • 3: すべての診断イベントが送信イベントです。 このため、ハードウェアコンポーネント障害はありません(緑の診断シンボル)。 The DiagEvent パラメータは評価されません(MODE = 3 の場合)。
DiagEvent	InOut	Diagnostic-Detail	L	診断イベントを指定します(「DiagEvent パラメータ」を参照)。
RET_VAL	Return	INT	I、Q、M、D、L	命令/エラーメッセージのステータス(「RET_VAL」パラメータを参照)

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

DiagEvent パラメータ

構造体 DiagnosticDetail は、診断イベント指定のためのシステムデータタイプです。 この構造体は、次のとおりです。

パラメータ		データタイプ	説明
Diagnostic-Detail		Struct	
	Chan-nelInfo	WORD	チャンネルプロパティ(0～7)
	ALID	UINT	アラームのローカル ID。 ID はアラームを一意に定義します。
	TextID	UINT	テキストリスト内のアラームテキストの ID。
	Chan-nel-Num-ber	UINT	メーカー固有のチャンネル番号(0x0000～0x7FFF)
	Addv-al_0	DWORD	追加情報のためのプレースホルダ。値/値のリストは接続エラーによって異なります。
	Tex-tID2	UINT	CPU 応答用のテキスト(モード、OB 呼び出しなど)。
	LADD R	HW_ANY	LADDR パラメータと同じ。
	Text-ListId	UINT	• 0: テキストリストなし • ≠0: テキストリストの ID
	Chan-nelDir-ection	UINT	• 0000: 対象外 • FFF1: 入力 • FFF2: 出力 • FFF3: 入力/出力
	Addv-al_1	DWORD	チャンネルエラー時の追加情報のためのプレースホルダ(GSD ファイルによって異なります)。 チャンネルエラーの種類については、 IEC 61158 (PROFINET IO Type 10 および PROFIBUS DP Type 3)も参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0	エラーは発生していません。
8080	MODE パラメータの値がサポートされていません。
8090	LADDR パラメータではハードウェアコンポーネントの識別はできません。
8091	LADDR パラメータでアドレス指定されたハードウェアコンポーネントに対して診断情報を生成できません。

80A1	• DiagEvent パラメータの DiagnosticsDetail 構造体の内容が、無効であるか矛盾しています。 • DiagEvent パラメータで、フェールセーフ固有の診断情報が定義されています(無効)。
80A4	アドレス指定されたハードウェアコンポーネントにアクセスできません。
80C1	並行実行にはリソースが不足しています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

GET_DIAG: 診断情報読み出し

説明

「GET_DIAG」命令を使用して、ハードウェアオブジェクトの診断情報を読み出します。 ハードウェアオブジェクトは、パラメータ LADDR を使用して選択されます。 MODE パラメータを使用し、読み出される診断情報を選択します。

パラメータ

以下の表に、「GET_DIAG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MODE	Input	UINT	I、Q、M、D、L、または定数	MODE パラメータを使用し、出力される診断データを選択します。
LADDR	Input	HW_ANY (WORD)	I、Q、M、L、または定数	デバイスのハードウェア識別子。
RET_VAL	Return	INT	I、Q、M、D、L	命令のステータス
CNT_DIAG	Output	UINT	I、Q、M、D、L	予約済み(常時「0」)。
DIAG	InOut	VARIANT	I、Q、M、D、L	診断情報は、選択されたモードに対応します。下の表を参照してください
DETAIL	InOut	VARIANT	I、Q、M、D、L	パラメータは非表示です。 このパラメータは使用しないでください。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ MODE

MODE パラメータの値に応じて、DIAG、CNT_DIAG および DETAIL 出力パラメータで異なる診断データが出力されます。

MODE	説明	DIAG	CNT_DIAG
0	DWORD としてモジュールのすべてのサポートされている診断情報の出力。ビット X=1 はモード X がサポートされていることを示します。	DWORD データタイプのビット: • ビット 0 = 1: MODE 0 がサポートされています • ビット 1 = 1: MODE 1 がサポートされています • ビット 2 = 1: MODE 2 がサポートされています • ビット 3 = 1:関連なし	0

1	アドレス指定されたハードウェアオブジェクトの診断ステータスの出力	構造体 DIS(説明は下記を参照): • MaintenanceState • ComponentStateDetail • OwnState • IOState • OperatingState	0
2	アドレス指定されたハードウェアオブジェクトのすべての従属モジュールのステータス出力。	構造体 DNN(説明は下記を参照): • SubordinateState • SubordinateIOState • DNNmode	0

DIS 構造体

パラメータ MODE = 1 の場合、DIS 構造体に基づいて診断情報が出力されます。 この場合、タグ宣言のデータタイプとして、システムデータタイプ「DIS」を入力します。

次の表に、個々のパラメータ値の意味を示します。

パラメータ	データタイプ	値	説明
MaintenanceS-tate	DWORD	ENUM	
		0	メンテナンス不要
		1	モジュールまたはデバイスが無効です。
		2	-
		3	-
		4	-
		5	メンテナンス必須
		6	メンテナンス要求
		7	エラー
		8	ステータス不明/従属モジュールのエラー
		9	-
		10	入力/出力が使用不可。
Component-StateDetail	DWORD	ビット配列	モジュールのサブモジュールのステータス: • ビット 0～15: モジュールのステータスメッセージ • ビット 16～31: CPU のステータスメッセージ
		0～2 (enum)	追加情報: • ビット 0: 追加情報なし • ビット 1: 転送が許可されていません。
		3	ビット 3 = 1: 少なくとも 1 つのチャンネルが診断の修飾子をサポートしています。
		4	ビット 4 = 1: 少なくとも 1 つのチャンネルまたは 1 つのコンポーネントのメンテナンス必須。

		5	ビット 5 = 1: 少なくとも 1 つのチャンネルまたは 1 つのコンポーネントのメンテナンス要求。
		6	ビット 6 = 1: 少なくとも 1 つのチャンネルまたは 1 つのコンポーネントでエラー。
		7 ~ 10	予約済み(常時= 0)
		11 ~ 14	• ビット 11 = 1: PNIO - サブモジュールが適正 • ビット 12 = 1: PNIO - 交換モジュール • ビット 13 = 1: PNIO - 不正なモジュール • ビット 14 = 1: PNIO - モジュール切断
		15	予約済み(常時= 0)
		16 ~ 31	CPU に生成されたモジュールのステータス情報: • ビット 16 = 1: モジュールが無効 • ビット 17 = 1: CiR 操作有効 • ビット 18 = 1: 入力使用不可 • ビット 19 = 1: 出力使用不可 ビット • 20 = 1: オーバーフロー診断バッファ ビット • 21 = 1: 診断使用不可 ビット • 22 - 31: 予約済み(常時 0)
OwnState	UINT	ENUM	パラメータ Ownstate の値によって、モジュールの保守ステータスが記述されます。
		0	障害なし
		1	モジュールまたはデバイスが無効です。
		2	メンテナンス必須
		3	メンテナンス要求
		4	エラー
		5	モジュールまたはデバイスが CPU にアクセスできません(CPU の下にあるモジュールとデバイスについて有効)。
		6	入力/出力が使用不可。
		7	-
IOState	WORD	ビット配列	モジュールの I/O ステータス
		0	ビット 0 = 1: メンテナンス不要
		1	ビット 1 = 1: モジュールまたはデバイスが無効です。
		2	ビット 2 = 1: メンテナンス必須
		3	ビット 3 = 1: メンテナンス要求
		4	ビット 4 = 1: エラー
		5	ビット 5 = 1: モジュールまたはデバイスが CPU にアクセスできません(CPU の下にあるモジュールとデバイスについて有効)。
		6	入力/出力が使用不可。
		7	修飾子。ビット 0、2、または 3 が設定されている場合、ビット 7= 1
		8 ~ 15	予約済み(常時= 0)

OperatingState	UINT	ENUM	
		0	-
		1	STOP /ファームウェアの更新
		2	STOP /メモリのリセット
		3	STOP /セルフ起動
		4	STOP
		5	メモリリセット
		6	START
		7	RUN 中
		8	-
		9	HOLD
		10	-
		11	-
		12	モジュール不良
		13	-
		14	電力なし
		15	CiR
		16	STOP / ODIS なし
		17	IN
		18	
		19	
		20	

DNN 構造体

パラメータ MODE = 2 の場合、DNN 構造体に基づいて診断情報詳細が出力されます。 この場合、タグ宣言のデータタイプとして、システムデータタイプ「DNN」を入力します。

次の表に、個々のパラメータ値の意味を示します。

パラメータ	データタイプ	値	説明
SubordinateState	UINT	Enum	従属モジュールのステータス(DIS 構造体のパラメータ OwnState を参照)
SubordinateIOS-tate	WORD	Bitarray	従属モジュールの入力および出力のステータス(DIS 構造体のパラメータ IO State を参照)
DNNmode	WORD	Bitarray	• ビット 0 = 0: 診断が有効 • ビット 0 = 1: 診断が無効 • ビット 1 ~ 15: 予約済み

RET_VAL パラメータ

エラーコード*	説明

(W#16#...)	
0	エラーは発生していません。
n	DETAIL パラメータのデータ領域が小さすぎます。 出力できない診断データの詳細もあります。
8080	MODE パラメータの値がサポートされていません。
8081	DIAG パラメータのタイプが選択したモードでサポートされていません(パラメータ MODE)。
8082	DETAIL パラメータのタイプが選択したモードでサポートされていません(パラメータ MODE)。
8090	LADDR が存在しません。
80C1	並列実行にはリソースが不足しています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

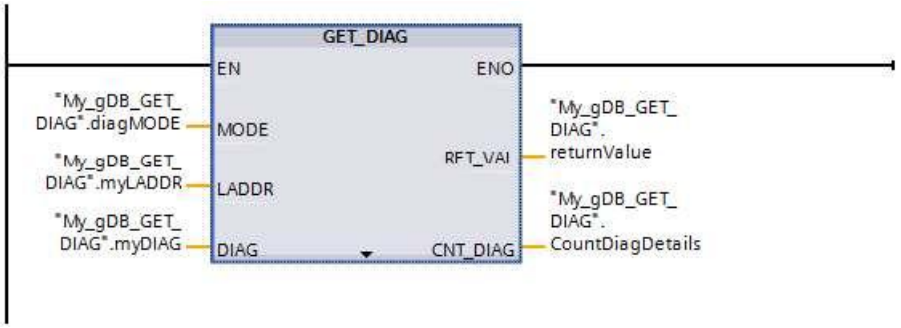
次の例では、CPU の診断情報を読み出します。

グローバルデータブロックにデータを保存するために、4 つのタグと 1 つの「myDIAG」構造体(DIS データタイプ)を作成します。

My_gDB_GET_DIAG

	Name	Data type	Start value
1	Static		
2	diagMODE	UInt	1
3	myLADDR	HW_ANY	50
4	returnValue	Int	0
5	CountDiagDetails	UInt	0
6	myDIAG	DIS	
7	MaintainanceState	DWord	16#0
8	ComponentStateDetail	DWord	16#0
9	OwnState	UInt	0
10	IOState	Word	16#0
11	OperatingState	UInt	0

この命令のパラメータを以下のように相互接続します。



CPU の HW 識別子は、パラメータ LADDR(「myLADDR」)を通じて「GET_DIAG」命令に伝達されます。 パラメータ MODE(「diagMODE」)の値「1」に従って、以下を適用します。

- この命令は、(CPU の)アドレス指定されたハードウェアオブジェクトのステータスを読み出します。
- パラメータ DIAG(「myDIAG」)で、診断情報が構造体(DIS データタイプ)に出力されます。

診断情報を理解するためには、16 進数値を 2 進コードに変換する必要があります。 パラメータ DIAG(「myDIAG」)に、以下が示されています。

- MaintenanceState: 値「0」に従って、CPU は保守を要求しません。
- ComponentStateDetail: 16 進数値「0000_8000」に従って、ビット 15 は有効です。
- OwnState: 値「0」に従って、障害は発生していません。
- IOState: 16 進数値「0001」に従って、保守は必要ありません。
- OperatingState: 出力「0」。

出力パラメータ RET_VAL(「returnValue」)は、処理がエラーなしで行われたことを示します。 出力パラメータ CNT_DIAG(「CountDiagDetails」)で、パラメータ DETAIL の「0」診断詳細が出力されています。

My_gDB_GET_DIAG

	Name	Data type	Start value	Monitor value
1	Static			
2	diagMODE	UInt	1	1
3	myLADDR	HW_ANY	50	16#0032
4	returnValue	Int	0	0
5	CountDiagDetails	UInt	0	0
6	myDIAG	DIS		
7	MaintainanceState	DWord	16#0	16#0000_0000
8	ComponentStateDetail	DWord	16#0	16#0000_8000
9	OwnState	UInt	0	0
10	IOState	Word	16#0	16#0001
11	OperatingState	UInt	0	0

注記: たとえば、ComponentStateDetail タグのビット 3(チャンネル診断、はい/いいえ) を個別に読み出すことができます。

- 次のようにビットをアドレス指定します。 ComponentStateDetail.%X3.

パルス

この章には下記に関する情報が記載されています：

- CTRL_PWM:パルス幅変調 (S7-1200, S7-1500)

CTRL_PWM:パルス幅変調

説明

「CTRL_PWM」命令を使用して、ソフトウェアを使って CPU がサポートするパルス出力を有効および無効にすることができます。

- 入力 PWM の命令で制御するパルスジェネレータのハードウェア ID を入力します。
- パルス出力は、命令の ENABLE 入力のビットが設定されたときに有効になります。
  - ENABLE の値が TRUE の場合、パルスジェネレータは、デバイス構成で定義されたプロパティを持つパルスを生成します。
  - ENABLE 入力のビットがリセットされるか、または CPU が STOP に移行すると、パルス出力が無効になり、それ以上パルスは生成されません。

S7-1200 は、命令「CTRL_PWM」が実行された場合、パルスジェネレータを有効にするため、S7-1200 では、BUSY の値は常に FALSE です。

ENO 許可出力は、EN 許可入力のシグナル状態が「1」で、命令の実行中にエラーが発生しなかった場合のみ設定されます。

注記

PWM および PTO でのフォーステーブルの使用

PWM および PTO に使用されるデジタル入力および出力は強制できません。 デバイス構成を使って割り当てられたデジタル入力および出力は、フォーステーブルおよびモニタリングテーブルのいずれでも制御できません。

要件

この命令の正しい実行の必要条件は、指定されたパルスジェネレータがハードウェアコンフィグレーションで有効であることです。

デバイスビューで、モジュールのプロパティを開きます。 [パルスジェネレータ(PTO/PWM)]に移動し、必要な PTO/PWM を開き、[全般]のファンクション[このパルスジェネレータを有効にする]を有効にします。

[パラメータの割り当て]に移動し、パルスオプションをセットします。

注記

パルス出力パラメータは排他的にデバイス構成に割り当てられ、「CTRL_PWM」命令は使用しません。 そのため、CPU への影響を目的としたパラメータの変更は、CPU が STOP モードに入っている間に行う必要があります。 パルスの持続時間の変更は例外です。

ユーザープログラムからのパルス持続時間の変更

ダイアログ[パルスオプション]で行ったパルス持続時間の設定は、ユーザープログラムから変更することができます。

[初期パルス持続時間]でセットされた値は、パルスジェネレータの出力バイトに書き込まれます。 開始アドレスと終了アドレスが、パルスジェネレータのプロパティの[I/O アドレス]に表示されます。

パルス持続時間を変更するには、デバイス構成で指定された出力ワードアドレスに必要な値を書き込みます。

例:

- 値 500 (10 進数)は、「初期パルス持続時間」で使用されます。 PTO/PWM の開始アドレスは「1000」で、終了アドレスは「1001」です。
- 2 進数値「0000000111110100」(10 進数の 500)は、出力バイトの両方に書き込まれます。
  - 開始アドレス(AB1000): 0000_0001 (BIN)
  - 終了アドレス(AB1001): 1111_0100 (BIN)

パルス持続時間は、パルス持続時間の形式としてセットされたパラメータに応じて異なることに注意してください(hundredths (100 分の 1)、thousandths (1000 分の 1)など)。

パラメータ

以下の表に、「CTRL_PWM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PWM	Input	HW_PWM	I、Q、M、D、L、または定数	パルスジェネレータのハードウェア ID ハードウェア ID は、デバイスビューのパルスジェネレータのプロパティにあります。 パルスジェネレータのハードウェア ID は、システム定数にも記載されています。
ENABLE	Input	BOOL	I、Q、M、D、L、または定数	パルス出力は、ENABLE = TRUE の場合に有効になり、ENABLE = FALSE の場合に無効になります。
BUSY	Output	BOOL	I、Q、M、D、L	処理ステータス
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス(下記を参照)。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません
80A1	パルスジェネレータのハードウェア ID が無効です。
80D0	指定されたハードウェア ID のパルスジェネレータは、有効になっていません。 [パルスジェネレータ(PTO/PWM)]の CPU プロパティでパルスジェネレータを有効にします。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

レシピおよびデータロギング

この章には下記に関する情報が記載されています：

- レシピファンクション (S7-1200, S7-1500)
- データロギング (S7-1200, S7-1500)

レシピファンクション

この章には下記に関する情報が記載されています：

- RecipeExport:レシピのエクスポート (S7-1200, S7-1500)
- RecipeImport:レシピのインポート (S7-1200, S7-1500)
- レシピ DB の構造 (S7-1200, S7-1500)

RecipeExport:レシピのエクスポート

説明

「RecipeExport」命令は、レシピデータをデータブロックから CPU のメモリカード内の CSV ファイルにエクスポートします。

エクスポートは「REQ」パラメータによってトリガされます。 エクスポート中、BUSY パラメータが「1」にセットされます。 エクスポート中にメモリカードのメインディレクトリの「Recipes」フォルダ内に CSV が生成されます。 データブロックの名前は、作製された CSV ファイルのファイル名として使用されます。同名の CSV ファイルが既に存在している場合、エクスポート時に既存のファイルが上書きされます。

命令の実行後、BUSY が「0」にリセットされ、DONE パラメータが「1」になることで操作の完了が示されます。 実行中にエラーが発生すると、パラメータ ERROR および STATUS によって通知されます。

パラメータ

以下の表に、「RecipeExport」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	制御パラメータ REQUEST: 立ち上がりエッジでエクスポートを有効にします。
REC-IPE_DB	InOut	VARIANT	D	レシピデータブロックへのポインタ。データブロックの構造体に関する情報については、次を参照してください: レシピ DB の構造
DONE	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 命令が実行されません。 • 1: 命令が実行されます。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 警告もエラーもなし。 • 1: エラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ

				「STATUS」パラメータの表を参照してください。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8090	CSV ファイルの名前に無効な文字が含まれています。 CSV ファイルのファイル名は、データブロックの名前と同一です。
8091	RECIPE_DB で参照されるデータ構造体を処理できません。
8092	RECIPE_DB パラメータのデータ構造体が 5000 バイトを超えています。
80B3	メモリカードまたは内部ロードメモリに十分なメモリ領域がありません。
80B4	メモリカードが書き込み禁止になっています。
80C0	CSV ファイルが一時的にロックされています。
80C1	データブロックが一時的にロックされています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

RecipeImport:レシピのインポート

説明

「RecipeImport」命令は、レシピデータをメモリカード内の CSV ファイルから RECIPE_DB パラメータのデータブロックにインポートします。 データブロック内の値は、処理中に上書きされます。

CSV ファイルのインポート時には、以下に注意してください。

- CSV ファイルはメモリカードの「Recipes」ディレクトリにある必要があります。
- CSV ファイルの名前は、RECIPE_DB パラメータのデータブロック名と一致している必要があります。
- CSV ファイルの最初の行(ヘッダー)は、レシピコンポーネントの名前を含んでいます(関連項目: レシピ DB の構造)。インポート中は最初の行は無視されます。 CSV ファイル内のレシピコンポーネントおよびデータブロックの名前は、インポート中は一致しません。
- CSV ファイルの各行の最初の値は、必ずレシピのインデックス番号です。 個々のレシピは、インデックス順にインポートされます。 レシピをインデックス順にインポートするには、CSV ファイル内のインデックスは連番で昇順にソートされている必要があります(それ以外の場合、エラーメッセージ 80B0 が STATUS パラメータに出力されます。
- CSV ファイルは、データブロック内で提供されている数より多いレシピデータレコードを含んではなりません。 データレコードの最大数は、データブロック内の配列制限値で示されています。

インポートは「REQ」パラメータによってトリガされます。 インポート中、BUSY パラメータが「1」にセットされます。命令の実行後、BUSY が「0」にリセットされ、DONE パラメータが「1」になることで操作の完了が示されます。 実行中にエラーが発生すると、パラメータ ERROR および STATUS によって通知されます。

パラメータ

次の表に、「RecipeImport」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	制御パラメータ REQUEST: 立ち上がりエッジでインポートを有効にします。
REC-IPE_DB	InOut	VARIANT	D	レシピデータブロックへのポインタ。データブロックの構造体に関する情報については、次を参照してください: レシピ DB の構造
DONE	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ

				• 0: 命令が実行されません。 • 1: 命令が実行されます。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 警告もエラーもなし。 • 1: エラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ 「STATUS」パラメータの表を参照してください。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8090	ファイル名に無効な文字が含まれています。
8092	インポート用の一致する CSV ファイルが見つかりません。 考えられる原因: CSV ファイル名がレシピ DB 名と一致していない。
80C0	CSV ファイルが一時的にロックされています。
80C1	データブロックが一時的にロックされています。
80B0	CSV ファイルのインデックスの番号が連続していない、昇順ではない、またはデータブロック内の最大数(配列制限値)を超過しています。
80B1	レシピデータブロックの構造と CSV ファイルが一致しません: CSV ファイルのフィールドが多すぎます。
80B2	レシピデータブロックの構造と CSV ファイルが一致しません: CSV ファイルのフィールドが少なすぎます。
80D0 +n	レシピデータブロックの構造と CSV ファイルが一致しません: フィールド n のデータタイプが一致しません(n<=46)。
80FF	レシピデータブロックの構造と CSV ファイルが一致しません: フィールド n のデータタイプが一致しません(n>46)。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

レシピ DB の構造

概要

次のセクションでは、簡単な例を使用してレシピ DB の構造体について説明します。 レシピ DB は 5 つのデータレコードで構成され、そのうちの 3 つが使用されます。 4 番目、および 5 番目のデータレコードは、今後の拡張のために確保されています。 各データレコードに 1 つのレシピが含まれています。レシピはレシピ名と 8 つの原料で構成されています。

product-name	water	barley	wheat	hops	yeast	wa-terTmp	mashTmp	mash-Time	QTest
Pils	10	9	3	280	39	40	30	100	0
Lager	10	9	3	150	33	50	30	120	0
BlackB-eer	10	9	3	410	47	60	30	90	1
Not_used	0	0	0	0	0	0	0	0	0
Not_used	0	0	0	0	0	0	0	0	0

レシピデータブロックの構造

レシピデータはグローバルデータブロック内に以下のように実装します。

- すべてのレシピのテンプレートは、PLC データタイプ「Beer_Recipe」、および対応するデータタイプのレシピコンポーネント「procutname」、「water」などです。
- グローバルデータブロック内では、PLC データタイプは Array [1.. 5] of "Beer_Recipe"として使用されます。 配列制限値(この例では 1～5)は、DB に含まれる可能性があるレシピの最大数を示します。
- レシピコンポーネントの値は、データブロック内に開始値として追加されます。
- グローバル DB は、入出力パラメータ RECIPE_DB によって命令と相互接続されます。

Recipe_DB

	Name	Data type	Offset	Start value
1	Static			
2	Products	Array [1 .. 5] of "Beer_Recipe"	...	
3	Products[1]	"Beer_Recipe"	...	
4	Products[2]	"Beer_Recipe"	...	
5	Products[3]	"Beer_Recipe"	...	
6	productname	String[20]	...	'BlackBeer'
7	water	UInt	...	10
8	barley	UInt	...	9
9	wheat	UInt	...	3
10	hops	UInt	...	410
11	yeast	UInt	...	47
12	waterTmp	UInt	...	60
13	mashTmp	UInt	...	30
14	mashTime	UInt	...	90
15	QTest	UInt	...	1
16	Products[4]	"Beer_Recipe"	...	
17	Products[5]	"Beer_Recipe"	...	

CSV ファイルへのエクスポート

「RecipeExport」命令の実行後、DB のデータは以下の構造の CSV ファイルに書き込まれます。

```
Recipe_DB.csv
index,productname,water,barley,wheat,hops,yeast,wa-
terTmp,mashTmp,mashTime,QTest
1,”Pils”,10,9,3,280,39,40,30,100,0
2,”Lager”,10,9,3,150,33,50,30,120,0
3,”BlackBeer”,10,9,3,410,47,60,30,90,1
4,”Not_used”,0,0,0,0,0,0,0,0,0
5,”Not_used”,0,0,0,0,0,0,0,0,0
```

Excel での表示

CSV ファイルを Excel で開くことで、より簡単に確認や編集ができます。 ファイルを開くとき、カンマがセパレータとして認識されない場合、Excel のインポートファンクションを使用してデータを構造化された形式で出力します。

	A	B	C	D	E	F	G	H	I	J	K
1	index	product	water	barley	wheat	hops	yeast	waterTmp	mashTmp	mashTime	QTest
2	1	"Pils"	10	9	3	280	39	40	30	100	0
3	2	"Lager"	10	9	3	150	33	50	30	120	0
4	3	"BlackBeer"	10	9	3	410	47	60	30	90	1
5	4	"Not_used"	0	0	0	0	0	0	0	0	0
6	5	"Not_used"	0	0	0	0	0	0	0	0	0

CSV ファイルの編集

CSV ファイルは、Web サーバーを使用して PC/プログラミングデバイスにアップロードし、編集することができます。 編集後、変更されたファイルを CPU に再読み込みすることができます。 これに関する既存の CSV ファイルを削除する必要があります。

「RecipeImport」命令を使用して、変更されたデータを CSV ファイルからデータブロックに再インポートすることが可能です。

変更されたデータは、まだ、このデータブロックと互換性を持たなければならないことに注意してください。 言い換えると、以下の通りです。

- 表の構造を変更することはできません(たとえば、新しい列に要素を加えることによって)。
- ファイルにデータレコードを追加する場合、データブロックへのインポート時に、データレコードの最大数を指定する配列制限値がデータレコード数と対応するよう注意してください。
- CSV ファイルへのエクスポート時に、インデックスが自動的に生成されます。 追加のデータレコードを作成する場合は、適宜連続するインデックス番号を追加します。
- 表のセルの値は、フォーマットと長さに関して、データブロックで使用されたデータタイプに対応していることが必要です。
  - 例 1: データタイプ INT がデータブロックで使用された場合、表では、整数のみを使用できます。
  - 例 2: データタイプ SINT がデータブロックで使用された場合、表では、値が-128 ~ +127 の整数のみを使用できます。

  表で変更を行うときは、下記の表に記載されている許容データタイプおよびデータ領域を常に銘記してください。

データタイプ		フォーマット	注
浮動小数点数	LReal	+9.9999999999999E+999	常に指数関数で書き込まれる
	Real	+9.9999999E+99	常に指数関数で書き込まれる

符号付き整数	LInt	+9999999999999999999	符号付整数の値の範囲: -9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	整数の値の範囲: -2147483648 ~ +2147483647
	Int	+99999	整数の値の範囲: -32768 ~ +32767
	SInt	+999	整数の値の範囲: -128 ~ +127
符号なし整数	ULInt	+9999999999999999999	値の範囲は 0 ~ +18446744073709551615 です。
	UDInt	+9999999999	整数の値の範囲: 0 ~ +4294967295
	UInt	+99999	整数の値の範囲:0 ~ +65535
	USInt	+999	整数の値の範囲: 0 ~ +255
2 進数	LWORD	+9999999999999999999	整数の値の範囲: 0 ~ +18446744073709551615
	DWord	+9999999999	整数の値の範囲:0 ~ +4294967295
	Word	+99999	整数の値の範囲: 0 ~ +65535
	Byte (バイト)	+999	整数の値の範囲:0 ~ +255
	Bool	9	値の範囲: 0 または 1
日付と時刻	LTIME	dddddd:hh:mm:ss.999_999_999	ミリ秒、マイクロ秒およびナノ秒付きの ISO フォーマット
	TIME	hhh:mm:ss.999	ミリ秒付きの ISO フォーマット
	S5TIME	hhh:mm:ss.999	ミリ秒付きの ISO フォーマット
	LDT	YYYY-MM-DD hh:mm:ss.999_999_999	ミリ秒、マイクロ秒およびナノ秒付きの ISO フォーマット
	DTL	YYYY-MM-DD hh:mm:ss.999_999_999	ミリ秒、マイクロ秒およびナノ秒付きの ISO フォーマット
	DT	YYYY-MM-DD hh:mm:ss.999	ミリ秒付きの ISO フォーマット
	DATE	YYYY-MM-DD	ISO フォーマット
	LTime_Of_Day	hh:mm:ss.999_999_999	ISO フォーマット
	TOD	hh:mm:ss.999	ISO フォーマット
文字	WString	"abcd"	• 個々の文字は、二重引用符記号で囲まなくてはいけません。 レシピ DB がインポートされる場合、二重引用符記号は省略することができます。 • 現在の長さ • データタイプ WString の文字列は、WChar データタイプの要素で構成されています。 レシピ DB が エクスポートされる場合、WChar データタイプの文字の内容は 16#FF(例: 16#1255 は 16#55 に変更されます)に制限されています。
	文字列	"abcd"	• 個々の文字は、二重引用符記号で囲まなくてはいけません。 レシピ DB がインポートされる場合、二重引用符記号は省略することができます。 • 現在の長さ

	WChar	"a"	個々の文字は、二重引用符記号で囲まなくてはいけません。 レシピ DB がインポートされる場合、二重引用符記号は省略することができます。 レシピ DB が エクスポートされる場合、WChar データタイプの文字の内容は 16#FF(例: 16#1255 は 16#55 に変更されます)に制限されています。
	Char	"a"	個々の文字は、二重引用符記号で囲まなくてはいけません。 レシピ DB がインポートされる場合、二重引用符記号は省略することができます。

データロギング

この章には下記に関する情報が記載されています：

データロギング - 概要

プロセス値の保存

データロギング命令は、ユーザープログラムでデータログにプロセス値を保存するために使用されます。 データログは、メモリカード(MC)または内部のロードメモリに保存できます。 データログは、CSV フォーマット(カンマ区切り値)で保存されます。

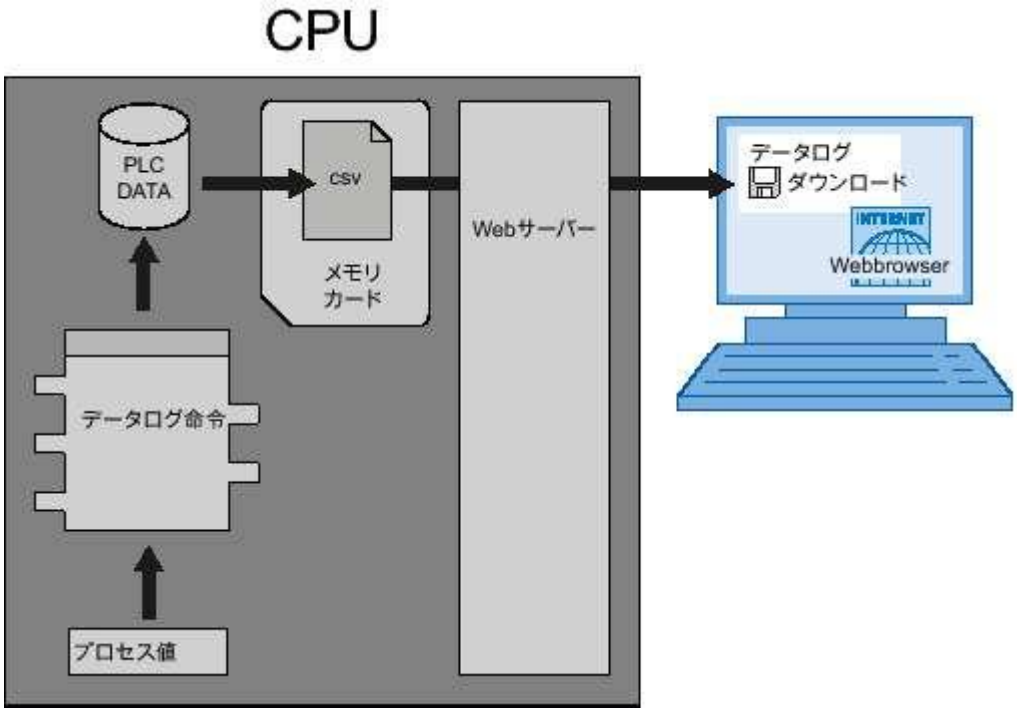
データタイプは、次のルールに従って文字列に変換されます。

データタイプ		フォーマット	注
浮動小数点数	LReal	+9.9999999999999E+999	常に指数関数で書き込まれる
	Real	+9.9999999E+99	常に指数関数で書き込まれる
符号付き整数	LInt	+9999999999999999999	符号付整数の値の範囲:-9223372036854775808 .. +9223372036854775807
	DInt	+9999999999	整数の値の範囲:-2147483648 ~ +2147483647
	Int	+99999	整数の値の範囲:-32768 ~ +32767
	SInt	+999	整数の値の範囲:-128 ~ +127
符号なし整数	ULInt	+9999999999999999999	値の範囲は 0 ~ +18446744073709551615 です。
	UDInt	+9999999999	整数の値の範囲:0 ~ +4294967295
	UInt	+99999	整数の値の範囲:0 ~ +65535
	USInt	+999	整数の値の範囲:0 ~ +255
2 進数	LWORD	+9999999999999999999	整数の値の範囲: +0000000000000000000 ~ +18446744073709551615
	DWord	+9999999999	整数の値の範囲: +0000000000 ~ +4294967295
	Word	+99999	整数の値の範囲: +00000 ~ +65535
	Byte (バイト)	+999	整数の値の範囲: +000 ~ +255
	Bool	9	値の範囲:0 または 1
日付と時刻	LTIME	dddddd:hh:mm:ss.999_999_999	ミリ秒、マイクロ秒およびナノ秒付きの ISO フォーマット
	TIME	hhh:mm:ss.999	ミリ秒付きの ISO フォーマット
	S5TIME	hhh:mm:ss.999	ミリ秒付きの ISO フォーマット
	LDT	YYYY-MM-DD hh:mm:ss.999_999_999	ミリ秒、マイクロ秒およびナノ秒付きの ISO フォーマット
	DTL	YYYY-MM-DD hh:mm:ss.999_999_999	ミリ秒、マイクロ秒およびナノ秒付きの ISO フォーマット
	DT	YYYY-MM-DD hh:mm:ss.999	ミリ秒付きの ISO フォーマット
	DATE	YYYY-MM-DD	ISO フォーマット

	LTime_Of_Day	hh:mm:ss.999_999_999	ISO フォーマット
	TOD	hh:mm:ss.999	ISO フォーマット
文字	WString	"abcd"	• 個々の文字は、二重引用符記号で囲まなくてはいけません。 • 現在の長さは、最大長までスペースで埋められています。 • データタイプ WString の文字列は、WChar データタイプの要素で構成されています。 データログが保存される場合、WChar データタイプの文字の内容は 16#FF(例: 16#1255 は 16#55 に変更されます)に制限されています。
	文字列	"abcd"	• 個々の文字は、二重引用符記号で囲まなくてはいけません。 • 現在の長さは、最大長までスペースで埋められています。
	WChar	"a"	個々の文字は、二重引用符記号で囲まなくてはいけません。 データログが保存される場合、WChar データタイプの文字の内容は 16#FF(例: 16#1255 は 16#55 に変更されます)に制限されています。
	Char	"a"	個々の文字は、二重引用符記号で囲まなくてはいけません。

使用しているプログラムでデータロギング命令を使用して、データログを作成または開く、エントリを書き込む、およびデータログファイルを閉じることができます。

データバッファを作成することで、データバッファの作成中にどのプログラム値がデータログに保存されるか決定します。 データバッファは、新規のデータログエントリ用のメモリとして使用します。新しい値は、「DataLogWrite」が呼び出される前にバッファに書き込まれる必要があります。 「DataLogWrite」命令の実行中に、データはバッファからデータログレコードに書き込まれます。



データログファイルは、次のように PC にコピーされます。

- PROFINET インターフェースが PC に接続されている場合、Web ブラウザを使用して Web サーバー経由でデータログにアクセスします。 これは、CPU の RUN または STOP モードで実行できます。 CPU が「RUN」モードの場合、Web サーバーによるデータの転送中もプログラムは実行を継続します。
- CPU にメモリカードがある場合、このカードを取り外し、PC やプログラミングデバイスの SD (セキュアデジタル)カードまたは MMC (MultiMediaCard)カード用の標準スロットに挿入します。 File Manager を使用して、メモリカードから PC へデータログファイルを転送します。 メモリカードを取り外すと、CPU が「STOP」に移行します。

データログのプロパティ

データログのデータレコードの書き込みは、リングバッファの原理に従って実行されます。 データレコードの最大数(RECORD パラメータ)に達するまで、新規データレコードが追加されます。 そして、次のデータレコードが、データログの「最も古い」データレコードを上書きします。

データレコードの上書きを防ぎたい場合は、「DataLogNewFile」命令を使用して現在のデータログに基づいて新規のデータログファイルを作成します。 この場合、新規のデータレコードは新規のデータログに書き込まれます。

データログの作成

「DataLogCreate」命令を使用して、ロードメモリの「"\DataLogs」ディレクトリに新規のデータログファイルを作成できます。

- NAME パラメータで割り当てられた名前がデータログの名称で、さらに CSV ファイルのファイル名にも使用されます。 このファイルは、ディレクトリ「DataLogs」に保存されます。
- ブロックパラメータ DATA は、新規のデータログオブジェクト、およびデータログの列およびデータタイプのデータバッファを指定します。 データログにあるデータレコードの列およびデータタイプは、このデータバッファの構造体宣言または配列宣言内の工レメントによって生成されます。 構造体または配列の各工レメントは、データログの 1 行の 1 列に対応します。
- HEADER ブロックパラメータを使用し、各列のヘッダーのヘッダーテキストを割り当てることができます。
- 「DataLogCreate」命令は ID を返します。この ID は、他のデータロギング命令が作成されたデータログの参照として使用します。

データログを開く

命令「DataLogOpen」(S7-1200 および S7-1500)を使用して、メモリカードの既存のデータログを開きます。 新規データレコードを書き込む前に、データログを開く必要があります。

データログは、「DataLogCreate」および「DataLogNewFile」命令を実行すると自動的に開きます。

同時に、10 個のデータログを開くことができます。 開くデータログは、データログの ID または名前を使用して指定できます。

- ID および NAME パラメータでデータログの ID および名前を指定した場合、それぞれデータログは ID を元に識別されます。データログの名前は比較されません。
- NAME パラメータを使いデータログを選択し、ID を指定しない場合、データログを開いたときに ID パラメータに ID が表示されます。
- ID パラメータを使いデータログを選択し、名前を指定しない場合、データログを開いたときに NAME パラメータに名前が表示されます。

MODE パラメータを使用し、開いたときにデータログのデータレコードを削除するかどうかを指定します。

データログへの書き込み

データレコードをデータログに書き込むためには、データログを(「DataLogOpen」命令で)開いておく必要があります。 「DataLogWrite」命令は、データレコードをデータログに書き込みます。

データログを閉じる

「DataLogClose」命令を使用し、開いているデータログを閉じます。 ID パラメータを使用してデータログを選択します。

CPU が STOP に切り替わった場合、および再起動された場合には、データログが自動的に閉じられます。

データログの削除

「DataLogDelete」命令(S7-1500)を使用し、メモリカードからデータログファイルを削除します。 ログが「DataLogCreate」命令で作成された場合のみ、ログファイルに含まれているデータログおよびデータレコードを削除することが可能です。

NAME および ID パラメータを使用して、削除するデータログを選択します。最初に ID パラメータが評価されます。 関連する ID のデータログが存在する場合、NAME パラメータは評価されません。 ID パラメータで値「0」が使用されている場合、NAME パラメータではデータタイプ STRING を使用する必要があります。

データログのクリア

「DataLogClear」命令(S7-1500)を使用し、既存のデータログ内のすべてのデータレコードを削除します。 この命令は、CSV ファイルのオプションヘッダーは削除しません(「DataLogCreate」命令の HEADER パラメータの説明を参照)。

ID パラメータを使用し、データレコードを削除するデータログを選択します。 データレコードを削除するには、データログを開いておく必要があります。

データログの新規ファイル

「DataLogNewFile」命令(S7-1200)、または「DataLogTypedNewFile」命令(S7-1500)を使用し、既存のデータログと同じプロパティの新規データログを作成します。 これにより、既存のデータログの内容を保持することが可能です。

この命令が呼び出されると、メモリカードまたは内部ロードメモリに NAME パラメータで定義された名前の新しいデータログを作成します。 ID パラメータを使用し、プロパティを新しいデータログに適用する古いデータログの ID を指定します。 その後、新しいデータログの ID は ID パラメータで出力されます。

命令の RECORDS パラメータを使用し、新しいデータログファイルサイズを指定します。

「DataLogTypedNewFile」命令(S7-1500)では、整合性チェックの実行が可能です。

DataLogCreate: データログ作成

説明

「DataLogCreate」命令を使用して、データログを作成します。

データログは、メモリカードまたは内部のロードメモリのディレクトリ「\DataLogs」に保存できます。 データログに保存できるデータ量は、メモリカードの空き領域または CPU の内部ロードメモリの記憶領域によって異なります。

データログに保存できるデータレコードの最大数は、RECORDS パラメータで指定します。 指定されたデータログのデータレコードの最大数に到達すると、最も古いデータレコードが上書きされます。既存のデータレコードの上書きを防ぐには、「DataLogNewFile」命令を使用します。 この命令を使用して、RECORDS パラメータで指定されている数に到達したとき(「DataLogWrite」命令の STATUS パラメータの戻り値が 1)、同じ構造体の新しいデータログを作成できます。 この場合、データレコードが新しいデータログに保存されます。

NAME パラメータで、データログの名前を指定できます。 データログは、CSV 形式(カンマ区切り値)で作成されます。 HEADER パラメータを使用し、データログのヘッダーを作成できます(オプション)。

データログが作成されたら、自動的に開きます。 これによって、データを書き込むことができます。

パラメータ

以下の表に、「DataLogCreate」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	命令の実行 データログは、パラメータ REQ での信号立ち上がりエッジで作成されます。
RECORDS	Input	UDInt	I、Q、M、L、D、または定数	データログのデータレコードの最大数 命令「DataLogWrite」がこのパラメータで指定されているよりも多くのレコードを書き込んだ場合、最も古いレコードが上書きされます。
FORMAT	Input	UInt	I、Q、M、L、D、または定数	データ形式: • 0: 内部(サポートなし) • 1: CSV (Comma separated values)
TIME-STAMP	Input	UInt	I、Q、M、L、D、または定数	タイムスタンプ: • 0: タイムスタンプなし • 1: 日付と時刻

				タイムスタンプが有効になった場合、ヘッダーの追加列が自動的に追加されます。
NAME	Input	VARIANT	L、D	データログの名前 指定された名前は、CSV ファイルのファイル名としても使用されます。 名前を割り当てる場合、Windows のファイル名の制限が適用されます。 次の文字は使用できません。 「\」、「/」、「,」、「:」、「*」、「?」、「<」、「>」、「|」、「空白文字」
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID (出力のみ) 作成されたデータログをアドレス指定するための追加のデータロギング命令には、データログの ID が必要です。
HEADER	InOut	VARIANT	L、D	データログのヘッダー(オプション) 命令が追加された後、このパラメータは非表示になります。 ヘッダーは、CSV ファイルに最初の行として書き込まれます。
DATA	InOut	VARIANT	L、D	命令「DataLogWrite」の実行時にデータレコードとして書き込まれるデータ構造体へのポインタ。
DONE	Output	BOOL	I、Q、M、L、D	ステータスパラメータ: • 0: 処理がまだ完了していません。 • 1: 命令の処理が正常に終了しました。
BUSY	Output	BOOL	I、Q、M、L、D	ステータスパラメータ: • 0: 命令の処理が開始されていないか、完了したか、またはキャンセルされました。 • 1: 命令の処理が進行中です。
ERROR	Output	BOOL	I、Q、M、L、D	ステータスパラメータ: • 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	詳細なステータス情報 詳細なエラー情報およびステータス情報は、パラメータ STATUS で出力されます。 このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

HEADER パラメータ

HEADER パラメータは、CSV ファイル(ヘッダー)のヘッダーを定義するデータブロックへの VARIANT ポインタです。 ヘッダーは、CSV ファイルの表記で必ず最初の行になります。

- ヘッダーの作成時には、各列がカンマ(S7-1200)またはセミコロン(S7-1500)で区切られている必要があることにご注意ください。
- 個々の列の名前には、STRING、Array of BYTE、または Array of CHAR データタイプを使用できます。 Array [...] of type データタイプでは、STRING データタイプよりも長い文字列を使用可能です。 STRING データタイプを使用する場合、長さは 254 バイトに制限されます。

ヘッダーを作成しない場合は、HEADER パラメータで値を指定しないでください。

パラメータ DATA

DATA パラメータは、データブロックの構造体または配列への VARIANT ポインタです。 構造体または配列のエレメントは、特定のデータタイプのデータログの列に対応します。

データブロックを作成する場合、次に注意してください。

- 列の数は、HEADER パラメータで定義された列の数と一致する必要があります。
- 構造体または配列の各エレメントは、CSV ファイルの列エントリに対応します。 したがって、データタイプ STRUCT を使用しているときは、ネスト構造(STRUCT 内の STRUCT)を使用できません。
- データ構造は、最大 256 エレメントを含むことができます。 256 を超えるエレメントがある場合、STATUS パラメータでエラーコード 8C52 が出力されます。
- データブロックのタグは、保持型または非保持型のタグとして設定できます。 ただし、保持設定は、データブロックのすべてのタグで同じであることが必要です。

STATUS (S7-1200)パラメータ

エラーコード * (W#16#...)	説明
0	エラーはありません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8070	内部インスタンスメモリ全部が割り当て済みです。
8090	無効なファイル名(NAME パラメータの説明を参照)。
8093	データログが既に存在します。
8097	ファイル長さがファイルシステム制限を超えました。
80A2	ファイルシステムから書き込みエラーが返されました。
80B3	メモリカードのメモリ領域が不十分です。
80B4	メモリカードが書き込み禁止になっています。
80C1	開いているデータログが多すぎます。
80C3	リソースが不足しています。 考えられる原因: • 異なるパラメータによる命令の複数呼び出し。

	• 10 を超えるデータログが同時に開いている。 命令「DataLogCreate」は作成中のログを自動的に開くため、続行するには、少なくとも 1 つのデータログを閉じる必要があります。
8253	RECORDS パラメータの値が無効です。
8453	形式の選択が無効です。
8553	無効なタイムスタンプ。
8B51	パラメータ HEADER のデータタイプが許可されていない、または長さが最大値を超過し得いる。
8C20	254 以外の長さ指定の文字列が使用されています。
8C24	パラメータ DATA の無効な割り当て(たとえば、ビットメモリをメモリ領域として使用)。
8C51	パラメータ DATA のデータタイプが許可されていない/ データ構造が使用できない。
8C52	パラメータ DATA の構造体に、256 を超えるエレメントが含まれています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

STATUS (S7-1500)パラメータ

エラーコード * (W#16#...)	**説明**
0	エラーはありません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
8070	内部インスタンスメモリ全部が割り当て済みです。
8090	無効なファイル名(NAME パラメータの説明を参照)。
8091	"NAME」パラメータが文字列ではありません。
8093	データログが既に存在します。
8097	ファイル長さがファイルシステム制限を超えました。
80A2	ファイルシステムから書き込みエラーが返されました。
80B3	メモリカードのメモリ領域が不十分です。
80B4	メモリカードが書き込み禁止になっています。
80C0	現在アクセスできません。
80C1	開いているデータログが多すぎます。
80C3	リソースが不足しています。考えられる原因: • 異なるパラメータによる命令の複数呼び出し。 • 10 を超えるデータログが同時に開いている。命令「DataLogCreate」は作成中のログを自動的に開くため、続行するには、少なくとも 1 つのデータログを閉じる必要があります。
8253	RECORDS パラメータの値が無効です。
8353	形式の選択が無効です。
8453	無効なタイムスタンプ。

8B24	パラメータ HEADER の無効な割り当て(たとえば、ビットメモリをメモリ領域として使用)。
8B51	パラメータ HEADER のデータタイプが許可されていない、または長さが最大値を超過し得いる。
8C24	パラメータ DATA の無効な割り当て(たとえば、ビットメモリをメモリ領域として使用)。
8C51	パラメータ DATA のデータタイプが許可されていない/ データ構造が使用できない。
8C52	パラメータ DATA の構造体に、256 を超えるエレメントが含まれています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

例

次の例では、タイムスタンプと 3 つのプロセス値を持つ単純なデータログが作成されています。

グローバルデータブロック内のタグ

データログの入力パラメータの値は、グローバルデータブロック「DataLogDB」に保存されます。

- DataLogName (String): このタグには、CSV ファイルのファイル名としても使用されるデータログの名前が含まれています。
- DataLogID (DInt): 命令が呼び出されたときに、データログの ID がこのタグに書き込まれます。
  - ID は命令によって自動的に割り当てられます。
  - このデータログをアドレス指定するには、他のデータログ命令でタグ「DataLogID」を使用します。
- MyHeader (String): このタグには、データログのヘッダー、つまり、プロセス値の列見出しが含まれています。 S7-1500 を使用しているときは、セミコロンが列の区切り文字として使用されます。
- MyData (Struct): このタグには、データログに書き込まれる 3 つのプロセス値が含まれています。 レコードが書き込まれるたびに(命令「DataLogWrite」)、現在値が新しいレコードに書き込まれます。

DataLogDB

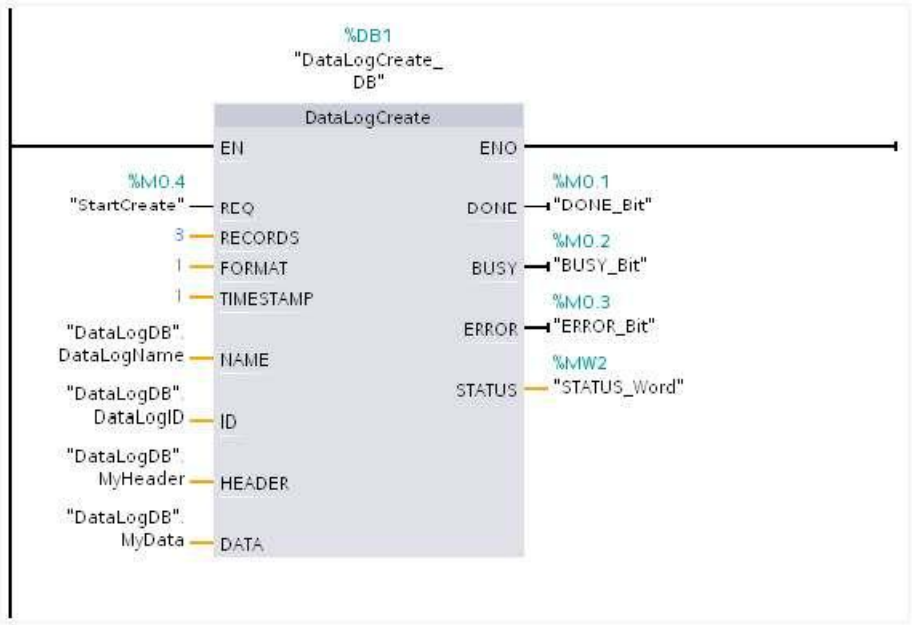
	Name	Data type	Start value	Retain
1	Static			
2	DataLogName	String	'MyDataLog'	
3	DataLogID	DInt	0	
4	MyHeader	String	'Value1;Value2;Value3'	
5	MyData	Struct		
6	ProcessValue1	Int	2	
7	ProcessValue2	Int	3	
8	ProcessValue3	Int	4	

「DataLogCreate」命令の呼び出し

この命令は、次の入力パラメータを使用して呼び出されます。

- REQ (BOOL): REQ = 「1」の場合、データログが作成されます。
- RECORD (3): 最大 3 つのデータレコードをデータログに書き込むことができます。 その後、最も古いデータレコードが上書きされます。
- FORMAT (1): データログは CSV ファイルとして作成されます。
- TIMESTAMP (1): 有効。 データログ用に 2 つの追加列(日付と時刻)が自動的に作成されます。 「DataLogWrite」が実行されるたびに、現在のタイムスタンプがデータレコードに書き込まれます。

- NAME (VARIANT): データブロック「DataLogDB」内のタグ「DataLogName」へのポインタ。
- ID (VARIANT): データブロック「DataLogDB」内のタグ「DataLogID」へのポインタ。
- HEADER (VARIANT): データブロック「DataLogDB」内のタグ「MyHeader」へのポインタ。
- DATA (VARIANT): データブロック「DataLogDB」内のタグ「MyData」へのポインタ。



Web サーバーを経由したデータログの読み出し

Web サーバーを使用して作成済みデータログを読み出すことができます。

- CPU のプロパティで Web サーバーを有効にします。 データログを読み出すには、Web サーバーに対して「ファイルの読み取り」権限を有効にする必要があります。
- インターネットブラウザで Web サーバーにアクセスするには、URL として CPU の IP アドレスを入力します。
- [Filebrowser]の下に、データログを含むディレクトリ「\DataLogs」が自動的に作成されています。
- 命令「DataLogWrite」がまだ実行されていない場合、データログには 1 つのエントリ「//END」しか含まれていません。 「DataLogWrite」の初期実行後に、最初のデータレコードが書き込まれます。

	A	B	C	D	E	F
1	SeqNo	Date	Time	ProcessValue1	ProcessValue2	ProcessValue3
2	1	01.07.2013	12:33:42.917	2	3	4
3						
4						
5						

DataLogOpen: データログを開く

この章には下記に関する情報が記載されています：

- DataLogOpen: データログを開く (S7-1200)
- DataLogOpen: データログを開く (S7-1500)

DataLogOpen: データログを開く

説明

「DataLogOpen」命令を使用して、メモリカードにある既存のデータログを開きます。 新規データレコードを書き込む前に、データログを開く必要があります。

データログは、「DataLogCreate」および「DataLogNewFile」命令を実行すると自動的に開きます。

同時に、10 個のデータログを開くことができます。 開くデータログは、データログの ID または名前を使って指定できます。

- ID および NAME パラメータでデータログの ID および名前を指定した場合、それぞれデータログは ID を元に識別されます。データログの名前は比較されません。
- NAME パラメータを使いデータログを選択し、ID を指定しない場合、データログを開いたときに ID パラメータに ID が表示されます。
- ID パラメータを使いデータログを選択し、名前を指定しない場合、データログを開いたときに NAME パラメータに名前が表示されます。

MODE パラメータを使用し、開いたときにデータログのデータレコードを削除するかどうかを指定します。

パラメータ

次の表に、「DataLogOpen」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、または定数	信号立ち上がりエッジでの命令の実行。
MODE	Input	UInt	I、Q、M、L、D、または定数	データログを開くモード: • MODE= "0" データログのデータレコードが保持されます • MODE= "1" データログのデータレコードは削除されますが、ヘッダは保持されます。
NAME	Input	VARIANT	L、D	データログの(ファイル)名。
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID。
DONE	Output	BOOL	I、Q、M、L、D	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、L、D	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、L、D	• 0: エラーなし。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	STATUS パラメータ

				このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプの追加情報については、「**有効なデータタイプの概要**」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません。
2	警告: このアプリケーションでデータログが既に開かれています。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8070	内部インスタンスメモリ全部が割り当て済みです。
8090	データログの定義と既存のデータログデータに矛盾があります。
8091	String 以外のデータタイプが NAME パラメータで使用されました。
8092	データログが存在しません。
80B4	メモリカードが書き込み禁止になっています。
80C0	データログファイルがロックされています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

DataLogOpen: データログを開く

説明

「DataLogOpen」命令を使用して、メモリカードにある既存のデータログを開きます。 新規データレコードを書き込むには、データログを開く必要があります。

データログは、「DataLogCreate」および「DataLogNewFile」命令を実行すると自動的に開きます。

同時に、10 個のデータログを開くことができます。 開くデータログは、データログの ID または名前を使用して指定できます。

- ID および NAME パラメータでデータログの ID および名前を指定した場合、それぞれデータログは ID を元に識別されます。データログの名前は比較されません。
- NAME パラメータを使いデータログを選択し、ID を指定しない場合、データログを開いたときに ID パラメータに ID が表示されます。
- ID パラメータを使いデータログを選択し、名前を指定しない場合、データログを開いたときに NAME パラメータに名前が表示されます。

MODE パラメータを使用し、開いたときにデータログのデータレコードを削除するかどうかを指定します。

DATA パラメータは、開くデータログと命令「DataLogCreate」のデータログの定義の間の整合性チェックを有効にします。 この整合性チェックは、データログが命令「DataLogCreate」によって作成された場合のみ実行できます。

- DATA パラメータで「DataLogCreate」命令の DATA パラメータと同じポインタを使用している場合、チェックを実行するとデータタイプが一致するかどうかが判定されます。 これ以外の場合、エラーコード W#16#8090 が STATUS パラメータに出力されます。
- 開くデータログが「DataLogCreate」によって作成されなかった場合、整合性チェックは使用できません。 この場合は、値「NULL」を DATA パラメータに入力します。

パラメータ

以下の表に、「DataLogOpen」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	信号立ち上がりエッジでの命令の実行。
MODE	Input	UInt	I、Q、M、L、D、または定数	データログを開くモード: • MODE= "0" データログのデータレコードが保持されます • MODE= "1"

				データログのデータレコードは削除されますが、ヘッダーは保持されます。
NAME	Input	VARIANT	L、D	データログの(ファイル)名。
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID。
DATA	InOut	VARIANT	L、D	整合性チェックあり: 「DataLogCreate」命令の DATA パラメータのデータ領域へのポインタ。
DONE	Output	BOOL	I、Q、M、L、D	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、L、D	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、L、D	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません。
2	警告: このアプリケーションでデータログが既に開かれています。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8070	内部インスタンスメモリ全部が割り当て済みです。
8090	データタイプの不整合。 ID パラメータのデータログが、DATA パラメータに指定されたデータタイプとは異なるデータタイプを使用します。
8091	String 以外のデータタイプが NAME パラメータで使用されました。
8092	データログが存在しません。
80B4	メモリカードが書き込み禁止になっています。
80C1	開いているファイルが多すぎます。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DataLogClear: データログを空にする

説明

「DataLogClear」命令は、既存のデータログ内のすべてのデータレコードを削除します。 この命令は、CSV ファイルのオプションヘッダーは削除しません(「DataLogCreate」命令の HEADER パラメータの説明を参照)。

ID パラメータを使用し、データレコードを削除するデータログを選択します。

要件

データレコードを削除するには、データログを開いておく必要があります(「DataLogOpen 命令」を参照)。

パラメータ

以下の表に、「DataLogClear」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	信号立ち上がりエッジでの命令の実行。
ID	InOut	DWORD	I、Q、M、D、L	データログのオブジェクト ID
DONE	Output	BOOL	I、Q、M、D、L	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8080	ID パラメータで選択したデータログファイルは、「DataLogClear」命令では処理できません。
8092	データログが存在しません。
80A2	ファイルシステムから書き込みエラーが返されました。
80B0	データログが開いていません。
80B4	メモリカードが書き込み禁止になっています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DataLogWrite: データログ書き込み

説明

「DataLogWrite」命令を使用して、既存のデータログにデータレコードを書き込みます。 データレコードを書き込むデータログの選択には、ID パラメータを使用します。 新規のデータレコードを作成するためには、データログが開いている必要があります。 この命令は、データログ作成時に DATA パラメータで指定された形式で新規のデータレコードを作成します。

「DataLogWrite」命令を呼び出す前に、データを「DataLogCreate」命令の DATA パラメータで相互接続したタグに転送します。 「DataLogWrite」命令が実行されると、転送されたデータはデータログにコピーされます。

通知
CPU への電源が切断された時点でデータログデータが消失 「DataLogWrite」命令の実行中に電源が切断されると、転送するデータレコードが失われます。

パラメータ

以下の表に、「DataLogWrite」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	信号立ち上がりエッジでの命令の実行。
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID
DONE	Output	BOOL	I、Q、M、L、D	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、L、D	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、L、D	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS (S7-1200)/パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません
0001	ファイルの最後に、最後の可能なデータレコードが作成されました。 別のデータレコードを作成すると、古いデータレコードが上書きされます。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8070	内部インスタンスメモリ全部が割り当て済みです。
8092	データログが存在しません。
80A2	ファイルシステムから書き込みエラーが返されました。
80B0	データログが開いていません。
80B3	メモリカードのメモリ領域が不十分です。
80B4	メモリカードが書き込み禁止になっています。
80C0	データログがロックされています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

STATUS (S7-1500)/パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません
0001	ファイルの最後に、最後の可能なデータレコードが作成されました。 別のデータレコードを作成すると、古いデータレコードが上書きされます。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
8070	内部インスタンスメモリ全部が割り当て済みです。
8092	データログが存在しません。
80A2	ファイルシステムから書き込みエラーが返されました。
80B0	データログが開いていません。
80B3	メモリカードのメモリ領域が不十分です。
80B4	メモリカードが書き込み禁止になっています。
80C0	データログがロックされています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DataLogClose: データログのクローズ

説明

「DataLogClose」命令を使用し、開いているデータログを閉じます。 ID パラメータを使用してデータログを選択します。

注記

データログを自動的に閉じる

CPU が STOP になった場合、または再起動された場合には、データログが自動的に閉じます。

パラメータ

次の表に、「DataLogClose」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	信号立ち上がりエッジで機能を実行します。
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID
DONE	Output	BOOL	I、Q、M、L、D	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、L、D	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、L、D	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	ステータスパラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS (S7-1200)パラメータ

エラーコード*	説明

(W#16#...)	
0	エラーはありません
1	データログが開いていません
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は無関係): 命令が既に有効です。BUSY の値は「1」です。
8092	データログが存在しません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

STATUS (S7-1500)パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません
1	データログが開いていません
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
8070	内部インスタンスメモリ全部が割り当て済みです。
8092	データログが存在しません。
80B4	メモリカードが書き込み禁止になっています。
80C0	現在アクセスできません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

DataLogDelete: データログの削除

説明

「DataLogDelete」命令を使用し、メモリカードからデータログファイルを削除します。 データログが「DataLogCreate」または「DataLogNewFile」命令で作成された場合のみ、データログとデータログに含まれているデータレコードを削除することが可能です。

パラメータ

以下の表に、「DataLogDelete」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	信号立ち上がりエッジでの命令の実行。
NAME	Input	VARIANT	L、D	データログのファイル名
DELFILE	Input	BOOL	I、Q、M、D、L	• 0: データログが保持されます。 • 1: データログが削除されます。
ID	InOut	DWORD	I、Q、M、D、L	データログのオブジェクト ID
DONE	Output	BOOL	I、Q、M、D、L	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	データログの削除が未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

NAME および ID パラメータ

NAME および ID パラメータを使用して、削除するデータログを選択します。 最初に ID パラメータが評価されます。 関連する ID のデータログが存在する場合、NAME パラメータは評価されません。 ID パラメータで値「0」が使用されている場合、NAME パラメータではデータタイプ STRING を使用する必要があります。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生していません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8091	STRING 以外のデータタイプが NAME パラメータで使用されています。
8092	データログが存在しません。
80A2	ファイルシステムから書き込みエラーが返されました。
80B4	メモリカードが書き込み禁止になっています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DataLogNewFile: 新規ファイルのデータログ

この章には下記に関する情報が記載されています：

- DataLogNewFile: 新規ファイルのデータログ (S7-1200)
- DataLogNewFile: 新規ファイルのデータログ (S7-1500)

DataLogNewFile: 新規ファイルのデータログ

説明

「DataLogNewFile」命令を使用し、既存のデータログと同じプロパティを持つ新しいデータログを作成します。 これにより、既存のデータログの内容の保持が可能になります。

この命令が呼び出されると、メモリカードまたは内部ロードメモリに NAME パラメータで定義された名前の新しいデータログを作成します。 ID パラメータを使用し、プロパティを新しいデータログに適用する古いデータログの ID を指定します。 その後、新しいデータログの ID は ID パラメータで出力されます。

命令の RECORDS パラメータを使用し、新しいデータログファイルサイズを指定します。

新しいデータログが作成されたら、自動的に開きます。 これによって、データを書き込むことができます。

パラメータ

以下の表に、「DataLogNewFile」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、または定数	信号立ち上がりエッジでの命令の実行。
RECORDS	Input	UDInt	I、Q、M、L、D、または定数	データログにあるデータレコードの数
NAME	Input	VAR-IANT	L、D	新しいデータログのファイル名.
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID • In: 既存のデータログの ID • OUT: 新しいデータログの ID
DONE	Output	BOOL	I、Q、M、L、D	命令が正常に実行されました。
BUSY	Output	BOOL	I、Q、M、L、D	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、L、D	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード * (W#16#...)	説明
0	エラーはありません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8070	内部インスタンスメモリ全部が割り当て済みです。
8090	無効なファイル名。
8091	NAME パラメータのデータタイプが STRING ではありません。
8092	ソースデータログが存在しません。
8093	新しいデータログが既に存在しています。
8097	ファイル長さがファイルシステム制限を超えました。
80A0	データタイプの不整合。 ID パラメータのデータログが、DATA パラメータに指定されたデータタイプとは異なるデータタイプを使用します。
80A2	ファイルシステムから書き込みエラーが返されました。
80B3	ロードメモリが不足しています。
80B4	メモリカードが書き込み禁止になっています。
80C1	開いているファイルが多すぎます。
8253	RECORDS パラメータの値が無効です。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DataLogNewFile: 新規ファイルのデータログ

説明

「DataLogNewFile」命令を使用し、既存のデータログと同じプロパティを持つ新しいデータログを作成します。 これにより、既存のデータログの内容の保持が可能になります。

この命令が呼び出されると、メモリカードまたは内部ロードメモリに NAME パラメータで定義された名前の新しいデータログを作成します。 ID パラメータを使用し、プロパティを新しいデータログに適用する古いデータログの ID を指定します。 その後、新しいデータログの ID は ID パラメータで出力されます。

命令の RECORDS パラメータを使用し、新しいデータログファイルサイズを指定します。

パラメータ DATA は、作成される新しいデータログと命令「DataLogCreate」のデータログの定義の間の整合性チェックを有効にします。 この整合性チェックは、データログが命令「DataLogCreate」によって作成された場合のみ実行できます。

- DATA パラメータで「DataLogCreate」命令の DATA パラメータと同じポインタを使用している場合、チェックを実行するとデータタイプが一致するかどうかが判定されます。 データタイプが一致しない場合、エラーコード W#16#80A0 が STATUS パラメータに出力されます。
- 開くデータログが「DataLogCreate」によって作成されなかった場合、整合性チェックは使用できません。 この場合は、値「NULL」を DATA パラメータに入力します。

新しいデータログが作成されたら、自動的に開きます。 これによって、データを書き込むことができます。

パラメータ

以下の表に、「DataLogNewFile」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、L、D、T、C、または定数 (S7-1500 では、T および C は LAD および FBD でのみ使用可能)	信号立ち上がりエッジでの命令の実行。
RECORDS	Input	UDInt	I、Q、M、L、D、または定数	データログにあるデータレコードの数
NAME	Input	VAR-IANT	L、D	新しいデータログのファイル名.
ID	InOut	DWORD	I、Q、M、L、D	データログのオブジェクト ID • In: 既存のデータログの ID • OUT: 新しいデータログの ID
DATA	InOut	VAR-IANT	L、D	整合性チェックを行うデータタイプ
DONE	Output	BOOL	I、Q、M、L、D	命令が正常に実行されました。

BUSY	Output	BOOL	I、Q、M、L、D	命令の実行が未完了です。
ERROR	Output	BOOL	I、Q、M、L、D	• 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、L、D	STATUS パラメータ このパラメータは、1 回の呼び出しの間のみ設定されます。 このため、ステータスを表示するには、STATUS パラメータを空きデータ領域にコピーします。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
0	エラーはありません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8070	内部インスタンスメモリ全部が割り当て済みです。
8090	無効なファイル名。
8091	NAME パラメータのデータタイプが STRING ではありません。
8092	ソースデータログが存在しません。
8093	新しいデータログが既に存在しています。
8097	ファイル長さがファイルシステム制限を超えました。
80A0	データタイプの不整合。 ID パラメータのデータログが、DATA パラメータに指定されたデータタイプとは異なるデータタイプを使用します。
80A2	ファイルシステムから書き込みエラーが返されました。
80B3	ロードメモリが不足しています。
80B4	メモリカードが書き込み禁止になっています。
80C0	現在アクセスできません。
80C1	開いているファイルが多すぎます。
8253	RECORDS パラメータの値が無効です。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

データログを使った作業のサンプルプログラム

概要

次のサンプルプログラムは、データログ命令の重要なファンクションを示しています。 個々の命令の詳細情報については、対応するリンクを使用して対応するヘルプの説明を参照してください。

データログの使用に関する一般的な注記

- 作成したデータログは、「DataLogCreate」および「DataLogNew」命令が実行されると自動的に開きます。
- データログは、CPU 再起動後に CPU が RUN から STOP に切り替わった後に自動的に閉じます。
- 「DataLogWrite」命令を実行する場合は、データログが開いている必要があります。
- S7-1200 CPU の場合は、最大 8 個のデータログを同時に開くことができます。 S7-1500 CPU の場合は、最大 10 個のデータログを同時に開くことができます。

サンプルプログラム

データログの内容は、このサンプルのデータブロック(DB)で定義されています。 DB はデータログを作成するために使用され(DataLogCreate)、データレコードを書き込むためのプロセス値を提供します(DataLogWrite)。

「MyData」構造体の 3 つのエントリは、プロセス値として使用されます。MyCount、MyTemerature および MyPressure。 データブロックには、これらの 3 つの値が一時的に保存され、次にデータレコードとして[DataLogWrite]命令によりデータログに転送されます。

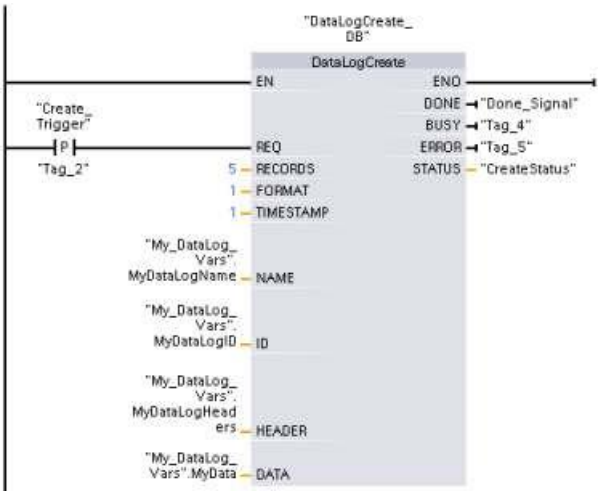
My_Datalog_Vars

		Name	Datentyp	Startwert
1		Static		
2		MyNewDataLogName	String	'MyNEWDataLog'
3		MyDataLogName	String	'MyDataLog'
4		MyDataLogID	DWord	0
5		MyDataLogHeaders	String	'Count,Temperature,Pressure'
6		MyData	Struct	
7		MyCount	Int	0
8		MyTemperature	Real	0.0
9		MyPressure	Real	0.0

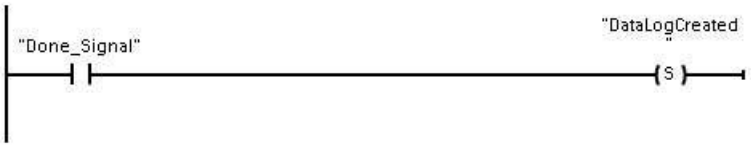
データレコードは後で次の 6 つのエントリから構成されます。

1. データレコード番号(自動割り当て)
2. 日付(TIMESTAMP パラメータで DataLogCreate に「1」が使用される場合、自動割り当て)。
3. 時刻(TIMESTAMP パラメータで DataLogCreate に「1」が使用される場合、自動割り当て)。
4. 「MyData」構造体の「MyCount」の現在値
5. 「MyData」構造体の「MyTemperature」の現在値
6. 「MyData」構造体の「MyPressure」の現在値

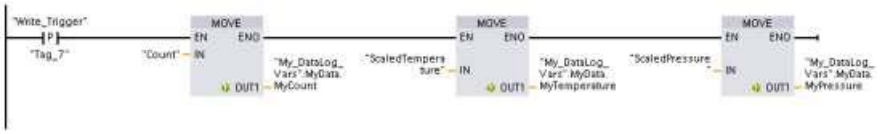
ネットワーク 1: REQ の立ち上がりエッジは、DataLogCreate 命令でデータログ作成を開始します。



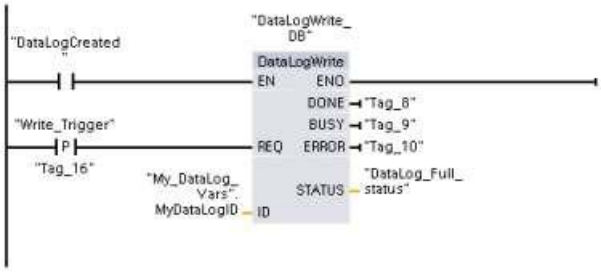
ネットワーク 2: 1 つのサイクルに対してのみ有効であるため、DataLogCreate の DONE 出力を取得します。



ネットワーク 3: 立ち上がりエッジは、MyData 構造体に新しいプロセス値が保存される時点をトリガします。 このステップは、データブロックに必要なプロセス値を一時的に保存するために使用されます。

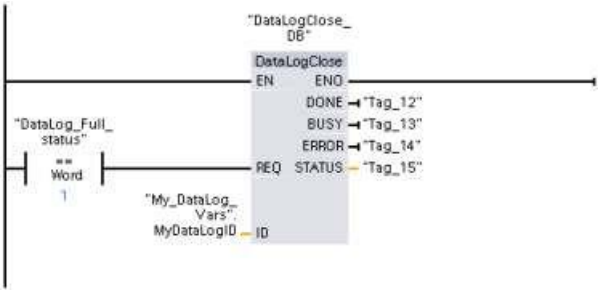


ネットワーク 4: DataLogCreate の実行が完了すると、(DONE パラメータ =1、ネットワーク 1 を参照)、DataLogWrite の EN 入力が設定されます。 この理由は、生成プロセスが複数のサイクルにわたって実行され、書き込み操作が実行されるために完了する必要があるからです。 書き込みは、REQ 入力の立ち上がりエッジによってデータレコードのためにトリガされます。

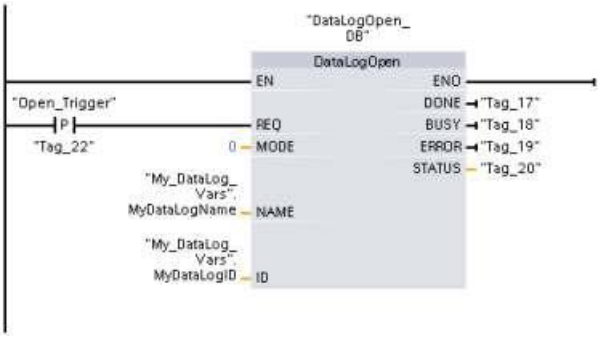


ネットワーク 5: 最後のデータレコードが書き込まれた後にデータログを閉じます。 データログは、5 個のデータレコードについて作成されました(ネットワーク 1 を参照)。 つまり、5 個のデータレコードの後に、DataLogWrite 命令の STATUS パラメータに 0001 が出力されます(ファイル末尾に最後のデータレコードを作成可能です。 別のデータレコードを作成すると、古いデータレコードが上書きさ

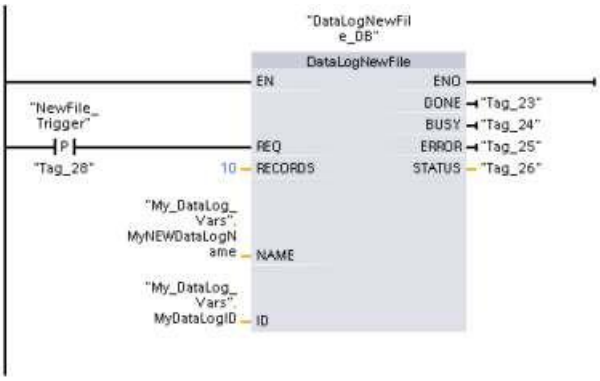
れます)。 この場合は、 REQ 入力が設定され、DataLogClose 命令が実行されます。 データログを閉じると、それ以上のデータレコードは書き込むことができません。



ネットワーク 6: データログは、後で再びデータレコードを書き込むために、DataLogOpen 命令を使用してもう一度開く必要があります。 別のデータレコードを DataLogWrite を使用して書き込む場合、最も古いデータレコードが上書きされます。



ネットワーク 7: 古いデータレコードを上書きしたくない場合は、DataLogNewFile 命令を使用して同じ構造を持つ新しいデータログを作成します。 これを行うには、構造をコピーしたい既存のデータログの ID を命令の ID パラメータに入力します。 DataLogNewFile 命令が実行されると、新しい一意の ID の値が新しいデータログに割り当てられます。

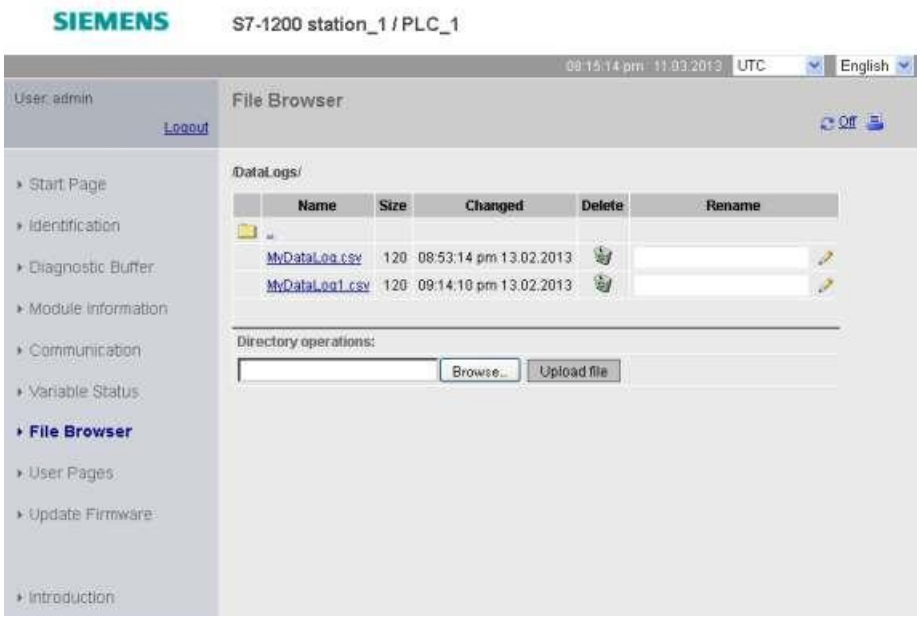


DataLogNewFile の呼び出しも複数のサイクルにわたって実行されることに注意してください。 したがって、DataLogCreate 命令と同様、DONE ビットも照会し、DataLogWrite の早期実行を防止してください(ネットワーク 1、2 および 4 を参照)。

結果

Web サーバー経由で書き込みデータを開く

サンプルプログラムで作成されたデータログは、Web サーバー経由で表示できます。 これを行うには、インターネットブラウザを使用して Web サーバーを開き、「\DataLogs」ディレクトリを開きます。



注記

[削除]および[名前の変更]オプションは、編集権限のあるユーザーアカウントでログオンしている場合のみ使用可能です。 CPU の Web サーバープロパティのハードウェアコンフィグレーションで権限を割り当てます。

CSV ファイルの内容

- DataLogCreate 命令でデータログを作成するとき、データレコードの最大数として「5」が設定されます。 この数を超えない場合、書き込まれたすべてのデータレコードはデータログに含まれます。

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	1	9/30/2010	20:26:56	1	9.86E+01	3.52E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

- 別のデータレコードがこれに追加される場合、最も古いデータレコード(Record 1)が上書きされます。

	A	B	C	D	E	F
1	Record	Date	UTC Time	Count	Temperature	Pressure
2	6	9/30/2010	20:32:03	6	9.86E+01	3.58E+01
3	2	9/30/2010	20:28:43	2	1.00E+02	3.73E+01
4	3	9/30/2010	20:29:03	3	9.99E+01	3.68E+01
5	4	9/30/2010	20:29:21	4	9.95E+01	3.64E+01
6	5	9/30/2010	20:30:19	5	9.92E+01	3.74E+01
7						

データブロックのファンクション

この章には下記に関する情報が記載されています：

- CREATE_DB:データブロックの作成 (S7-1500)
- READ_DBL: ロードメモリのデータブロックからの読み出し (S7-1200, S7-1500)
- WRIT_DBL: ロードメモリのデータブロックへの書き込み (S7-1200, S7-1500)
- ATTR_DB: データブロック属性読み出し (S7-1500)
- DELETE_DB:データブロック削除 (S7-1500)

CREATE_DB:データブロックの作成

説明

「CREATE_DB」命令を使用して、ロードメモリ、またはワークメモリ、またはその両方に新規データブロックを作製します。

「CREATE_DB」命令は、ユーザープログラムのチェックサムを変更しません。

データブロックの番号

作成されたデータブロックには、LOW_LIMIT (下限値)、および UP_LIMIT (上限値)パラメータで定義された範囲から番号を割り当てます。 "CREATE_DB」は、指定された範囲から割り当て可能な最小の番号を DB に割り当てます。 既にユーザープログラムに含まれる DB の番号を割り当てることはできません。

特定の番号の DB を作成するには、指定する範囲の上限値および下限値と同じ番号を入力します。 ワークメモリおよび/またはロードメモリに同じ番号の DB が存在する場合、または、コピーバージョンとして DB が存在する場合、この命令は終了し、エラーメッセージが RET_VAL パラメータで生成されます。

データブロックの開始値

SRCBLK パラメータを使用して、作成する DB の開始値を定義できます。 SRCBLK パラメータは、開始値を適用する DB または DB 領域へのポインタです。 SRCBLK パラメータでアドレス指定される DB は、標準アクセス(「最適化したブロックアクセス」属性が無効の状態)で生成されている必要があります。

- SRCBLK パラメータで指定された領域が、生成された DB よりも大きい場合、生成された DB の長さまでの値が開始値として適用されます。
- SRCBLK パラメータで指定された領域が、生成された DB よりも小さい場合、残りの値は「0」で埋められます。

データの整合性を保証するため、「CREATE_DB」の実行中にこのデータ領域を変更しないでください (つまり、BUSY パラメータの値が TRUE の間)。

ファンクションの説明

「CREATE_DB」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。REQ=1 で「CREATE_DB」を呼び出してジョブを開始します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: **非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味**

パラメータ

以下の表に、「CREATE_DB」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ= 1: データブロック作成の要求
LOW_LIMIT	Input	UINT	I、Q、M、D、L、または定数	「CREATE_DB」が DB 60000 への番号割り当てに使用する領域###の下限値

<table>
<tr><td>UP_LIM-IT</td><td>Input</td><td>UINT</td><td>I、Q、M、D、L、または定数</td><td>「CREATE_DB」が DB への番号割り当てに使用する領域###の上限値(使用可能な最大 DB 番号: 60999)</td></tr>
<tr><td>COUNT</td><td>Input</td><td>UDINT</td><td>I、Q、M、D、L、または定数</td><td>カウンタ値は、生成される DB に予約するバイト数を指定します。 バイト数は偶数であることが必要です。 最大長は 65534 バイトです。</td></tr>
<tr><td>ATTRIB</td><td>Input</td><td>BYTE</td><td>I、Q、M、D、L、または定数</td><td>ATTRIB パラメータのバイトの最初の 4 ビットを使用して、データブロック*のプロパティを定義します。
• ビット 0 = 0: 属性[ロードメモリにのみ保存]が設定されていません。
• ビット 0 = 1: 属性[ロードメモリにのみ保存]が設定されています。 この設定により、DB はワークメモリ内のスペースをまったく取らず、プログラムに含まれません。 DB はビットコマンドを使用してアクセスできません。 ビット 0 = 1 の場合、ビット 2 の選択は対象外です。
• ビット 1 = 0: 属性「デバイス内のデータブロックは書き込み禁止」が設定されていません。
• ビット 1 = 1: 属性「デバイス内のデータブロックは書き込み禁止」が設定されています。
• ビット 2 = 0: DB は保持型です(ロードメモリ内に生成された DB の場合のみ) 少なくとも 1 つの値が保持型として設定されている場合、DB は保持型とみなされます。
• ビット 2 = 1: DB は保持型ではありません。
• ビット 3 = 0: ロードメモリ内またはワークメモリ内のいずれかでの DB の作成(ビット 0 を使用した選択、上記を参照)
• ビット 3 = 1: ロードメモリ内およびワークメモリ内の両方での DB の作成(ビット 0 は対象外)
STEP 7 V5.x と確実に互換するように、ビット 0 および 3 を組み合わせて使用する必要があります。
<table>
<tr><th>ビット 0</th><th>ビット 3</th><th>DB の生成</th></tr>
<tr><td>0</td><td>0</td><td>ワークメモリ内のみ</td></tr>
<tr><td>1</td><td>0</td><td>ロードメモリ内のみ</td></tr>
<tr><td>0</td><td>1</td><td>ワークメモリおよびロードメモリ</td></tr>
<tr><td>1</td><td>1</td><td>ワークメモリおよびロードメモリ</td></tr>
</table>
• Bit 4 = 0 - 指定された開始値はありません (SRCBLK パラメータの入力値は無視されます)。
• Bit 4 = 1 - 開始値(SRCBLK パラメータによってアドレス指定された DB に対応する値)を指定します。</td></tr>
<tr><td>SRCBLK</td><td>Input</td><td>VAR-IANT</td><td>D</td><td>生成されるデータブロックを初期化するために値が使用されるデータブロックへのポインタ。</td></tr>
<tr><td>RET_VAL</td><td>Return</td><td>INT</td><td>I、Q、M、D、L</td><td>エラー情報</td></tr>
<tr><td>BUSY</td><td>Output</td><td>BOOL</td><td>I、Q、M、D、L</td><td>BUSY= 1:プロセスが未完了です。</td></tr>
</table>

DB_NUM	Output	DB_DYN (UINT)	I、Q、M、D、L	作成済みの DB 数。
* ここで選択したプロパティは、データブロックのプロパティの属性に対応します。				

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0081	ソース範囲よりも大きなターゲット範囲です。 ソース範囲すべてがターゲット範囲に書き込まれます。 ターゲット範囲の残りのバイトは 0 で埋められます。
7000	REQ=0 による最初の呼び出し: 有効なデータ転送がありません。BUSY の値は「0」です。
7001	REQ=1 による最初の呼び出し: データ転送がトリガされました。BUSY の値は 1 です。
7002	中間呼び出し(REQ は対象外): データ転送が既に有効です。BUSY の値は「1」です。
8081	ターゲット範囲よりも大きなソース範囲です。 ターゲット範囲すべてが書き込まれます。 ソース範囲の残りのバイトは無視されます。
8092	次の理由で、「データブロックの作成」機能が現在使用できません。 • 現在「ユーザーメモリ圧縮」機能が有効になっています。 • 使用している CPU のブロック最大数に既に達しています。
8093	SRCBLK パラメータにデータブロックが指定されていないか、ワークメモリにないデータブロックが指定されています。
8094	ATTRIB パラメータに、まだサポートされていない属性が指定されています。
80A1	DB 番号エラー: • 番号が「0」 • 下限値 > 上限値
80A2	DB 長エラー: • 長さが「0」 • 長さが奇数 • 長さが CPU で許可される長さを超えている
80A3	SRCBLK パラメータのデータブロックが標準アクセスで作成されていません。
80B1	空いている DB 番号がありません。
80B2	ワークメモリが不足しています。
80B4	メモリカードが書き込み禁止になっています。
80BB	ロードメモリが不足しています。
80C3	同時に有効な「CREATE_DB」命令の最大数に既に達しています。

* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

READ_DBL: ロードメモリのデータブロックからの読み出し

説明

この命令を使用して、DB またはロードメモリ(マイクロメモリカード)の DB の領域をコピー先の DB のデータ領域にコピーします。 宛先の DB は、実行に関連する必要があります。つまり、UNLINKED 属性付きで作成することはできません。 コピー処理中にロードメモリの内容は変更されません。

データの整合性を確保するため、「READ_DBL」の実行中(つまり、BUSY パラメータの値が TRUE の間)はターゲット範囲を変更しないでください。

SRCBLK および DSTBLK パラメータ(ソースおよび宛先ブロック)には、次の制限が適用されます。

- VARIANT ポインタの長さは、8 で割り切れる必要があります。
- タイプ STRING の VARIANT ポインタは、長さが 1 と等しいことが必要です。
- ソースおよび宛先ブロックは、同じブロックアクセスで作成されている必要があります。つまり、両方ともアクセスタイプ「最適化」または「標準」のいずれかを使用している必要があります。

注記

""「READ_DBL」は、非同期で処理されます。 したがって、ロードメモリのタグの頻繁な(周期的な)読み出しには適しません。

ジョブは一旦開始されると、必ず完了します。 同時に有効な「READ_DBL」命令の最大数に達し、この時点で再度さらに優先度の高いクラスで「READ_DBL」を呼び出すと、エラーコード W#16#80C3 が返されます。 その結果、直ちにさらに高い優先度のジョブを再開する意味はありません。

ファンクションの説明

「READ_DBL」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 REQ =1 で「READ_DBL」を呼び出してジョブを開始します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味

パラメータ

以下の表に、「READ_DBL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: 読み出し要求
SRCBLK	Input	VARIANT	D	読み出し元のロードメモリのデータブロックへのポインタ
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
BUSY	Output	BOOL	I、Q、M、D、L	BUSY = 1: 読み出しプロセスが未完了です。
DSTBLK	Output	VARIANT	D	書き込み先のワークメモリのデータブロックへのポインタ

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
0081	ソース範囲よりも大きなターゲット範囲です。 ソース範囲が完全にターゲット範囲に書き込まれます。ターゲット範囲の残りのバイトは変更されません。
7000	REQ=0 による最初の呼び出し:有効なデータ転送がありません。BUSY の値は「0」です。
7001	REQ=1 による最初の呼び出し: データ転送がトリガされました。BUSY の値は 1 です。
7002	中間呼び出し(REQ は対象外): データ転送が既に有効です。BUSY の値は「1」です。
8x51	注記: このエラーコードは S7-1200 CPU の場合のみ使用されます。 データブロックのデータタイプエラーです。
8081	ターゲット範囲よりも大きなソース範囲です。 ターゲット範囲すべてが書き込まれます。 ソース範囲の残りのバイトは無視されます。
8082	宛先 DB のタイプがソース DB のタイプと異なります(最適化/標準アクセス)。
8093	注記: このエラーコードは S7-1500 CPU の場合のみ使用されます。 DSTBLK パラメータにデータブロックが指定されていないか、ワークメモリにないデータブロックが指定されています。
80B1	注記: このエラーコードは S7-1500 CPU の場合のみ使用されます。 パラメータ DSTBLK では、ロードメモリ内にあるデータブロックのみが許可されています。
8xB1	注記: このエラーコードは S7-1200 CPU の場合のみ使用されます。 SRCBLK パラメータにデータブロックが指定されていない、または、そこで指定されたデータブロックがロードメモリオブジェクトではありません。
80B4	メモリカードが書き込み保護されているか、または DSTBLK が F 属性を持つ DB を指しています(書き込むことはできません)。
8xC0	注記: このエラーコードは S7-1200 CPU の場合のみ使用されます。 宛先の DB が、別の命令または通信機能によって現在処理中です。
80C3	同時に有効な「READ_DBL」命令の最大数に既に達しています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

WRIT_DBL: ロードメモリのデータブロックへの書き込み 

説明

「WRIT_DBL」命令を使用して、DB または DB 領域の内容をワークメモリからロードメモリ(マイクロメモリカード)の DB または DB 領域に転送します。 ソース DB は実行に関連する必要があります。つまり、[ロードメモリにのみ保存]属性付きで作成することはできません。

データの整合性を確保するため、「WRIT_DBL」の実行中(つまり、BUSY パラメータの値が TRUE の間)はソース範囲を変更しないでください。

SRCBLK および DSTBLK パラメータ(ソースおよび宛先ブロック)には、次の制限が適用されます。

- タイプ BOOL の VARIANT ポインタは、長さが 8 で割り切れることが必要です。
- タイプ STRING の VARIANT ポインタは、長さが 1 と等しいことが必要です。
- ソースおよび宛先ブロックは、同じブロックアクセスで作成されている必要があります。つまり、両方とも「最適化したブロックアクセス」または最適化したアクセスを無効にしている必要があります。

「WRIT_DBL」命令は、命令を使用して作成した DB に書き込む場合は、ユーザープログラムのチェックサムを変更しません。 ただし、読み込まれた DB が書き込まれる場合、この DB の最初のエントリがユーザープログラムのチェックサムが変更されます。

注記
"WRIT_DBL」は、ロードメモリのタグの頻繁な(周期的な)書き込みには適しません。 これは、マイクロメモリカードテクノロジーが、マイクロメモリカードへの書き込みアクセス数を制限しているためです。

ファンクションの説明

「WRIT_DBL」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。 REQ = 1 で「WRIT_DBL」を呼び出すことによって、ジョブを開始します。

出力パラメータ RET_VAL および BUSY は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

パラメータ

以下の表に、「WRIT_DBL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: 書き込み要求
SRCBLK	Input	VARIANT	D	読み出し元のワークメモリの DB へのポインタ
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
BUSY	Output	BOOL	I、Q、M、D、L	BUSY= 1: 書き込みプロセスが未完了です。
DSTBLK	Output	VARIANT	D	書き込み先のロードメモリのデータブロックへのポインタ

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生していません。
0081	ソース範囲よりも大きなターゲット範囲です。 ソース範囲が完全にターゲット範囲に書き込まれます。ターゲット範囲の残りのバイトは変更されません。
7000	REQ=0 による最初の呼び出し: 有効なデータ転送がありません。BUSY の値は「0」です。
7001	REQ=1 による最初の呼び出し: データ転送がトリガされました。BUSY の値は 1 です。
7002	中間呼び出し(REQ は対象外): データ転送が既に有効です。BUSY の値は「1」です。
8x51	注記: このエラーコードは S7-1200 CPU の場合のみ使用されます。 データブロックのデータタイプエラーです。
8081	ターゲット範囲よりも大きなソース範囲です。 ターゲット範囲すべてが書き込まれます。ソース範囲の残りのバイトは無視されます。
8082	宛先 DB のタイプがソース DB のタイプと異なります(最適化されたアクセス/最適化されていないアクセス)。
8093	注記: このエラーコードは S7-1500 CPU の場合のみ使用されます。 SRCBLK パラメータにデータブロックが指定されていないか、ワークメモリにないデータブロックが指定されています。
80B1	注記: このエラーコードは S7-1500 CPU の場合のみ使用されます。 パラメータ DSTBLK では、ロードメモリ内にあるデータブロックのみが許可されています。
8xB1	注記: このエラーコードは S7-1200 CPU の場合のみ使用されます。 DSTBLK パラメータにデータブロックが指定されていない、または、そこで指定されたデータブロックがロードメモリ内に存在しません。
80B4	• メモリカードが書き込み禁止になっています。 • F-属性の DB は読み出さないでください。
80BB	使用可能なロードメモリが不足。
8xC0	注記: このエラーコードは S7-1200 CPU の場合のみ使用されます。 宛先の DB が、別の命令または通信命令によって現在処理中です。
80C3	注記: このエラーコードは S7-1500 CPU の場合のみ使用されます。 同時に有効な「WRIT_DBL」命令の最大数に既に達しています。
一般エラーコード	関連項目: GET_ERR_ID: ローカルでエラー ID を取得

* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

ATTR_DB: データブロック属性読み出し

説明

「ATTR_DB」命令を使用して、CPU のワークメモリ内にあるデータブロック(DB)に関する情報を取得します。 この命令は、選択された DB の ATTRIB パラメータで設定された属性を識別します。

データブロックの長さが最適化されたアクセスで読み出せません。DB_LENGH パラメータには、最適化されたアクセスのある DB の長さ「0」が含まれています。

モーションコントロール用のデータブロックは、「ATTR_DB」命令では読み出すことができません。この場合、エラーコード 80B2 が出力されます。

パラメータ

以下の表に、「ATTR_DB」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ = 1: ブロック属性の読み出し要求
DB_NUMBER	Input	DB_ANY (UINT)	I、Q、M、D、L、または定数	テストする DB の番号
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報
DB_LENGTH	Output	UDINT	I、Q、M、D、L	選択された DB が含むデータのバイト数
ATTRIB	Output	BYTE	I、Q、M、D、L	DB プロパティ: • ビット 0*= 0: 属性[ロードメモリにのみ保存]が設定されていません。 • ビット 0*= 1: 属性[ロードメモリにのみ保存]が設定されています。 • ビット 1 = 0: 属性「デバイス内のデータブロックは書き込み禁止」が設定されていません。 • ビット 1 = 1: 属性「デバイス内のデータブロックは書き込み禁止」が設定されています。 ビット 0 = 1 の場合、ビット 2 は対象外であり、値 1 を取得します。 • ビット 2 = 0: 保持型 - 少なくとも 1 つの値が保持型として設定されている場合、DB は保持型とみなされます。 • ビット 2 = 1: 非保持型 - DB 全体が保持型でないもの。 • ビット 3*= 0: DB は、ロードメモリ内(ビット 0 = 1)、またはワークメモリ内(ビット 0 = 0)のいずれかにあります。 • ビット 3*= 1: DB はロードメモリとワークメモリの両方に生成されます

* ビット 0 とビット 3 の関係は、「CREATE_DB:データブロックの作成」命令のパラメータ内で説明しています。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
80A1	入力パラメータ DB_NUMBER のエラー: 選択した実際のパラメータが • 「0」です。 • 使用している CPU の最大許容 DB 番号を超えています。
80B1	CPU に指定された番号の DB が存在しません。
80B2	モーションコントロールテクノロジーオブジェクトのデータブロックは、「ATTR_DB」命令で読み出すことができません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

DELETE_DB:データブロック削除

説明

命令「DELETE_DB」を使用して、命令「CREATE_DB」を呼び出すことによってユーザープログラムから作成されたデータブロック(DB)を削除します。

データブロックが「CREATE_DB」によって作成されなかった場合は、エラーコード W#16#80B5 が RET_VAL パラメータに出力されます。

選択されたデータブロックはすぐには削除されませんが、サイクル OB の実行後にサイクルコントロールポイントで削除されます。

ファンクションの説明

「DELETE_DB」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。割り込みの転送は、REQ = 1 でこの命令を呼び出して開始します。

出力パラメータ BUSY および出力パラメータ RET_VAL のバイト 2 と 3 は、ジョブのステータスを示します。

データブロックの削除は、出力パラメータ BUSY が値 FALSE を持つときに完了します。

パラメータ

以下の表に、「DELETE_DB」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	REQ =1: パラメータに番号が含まれる DB を削除する要求 DB_NUMBER
DB_NUM-BER	Input	UINT	I、Q、M、D、L、または定数	削除する DB 番号
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報(「RET_VAL パラメータ」を参照)
BUSY	Output	BOOL	I、Q、M、D、L	BUSY= 1:プロセスが未完了です。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
7000	REQ=0 による最初の呼び出し:有効なデータ転送がありません。BUSY の値は「0」です。
7001	REQ=1 による最初の呼び出し:データ転送がトリガされました。BUSY の値は 1 です。

7002	中間呼び出し(REQ は対象外):データ転送が既に有効です。BUSY の値は「1」です。
80A1	入力パラメータ DB_NUMBER のエラー: • パラメータの値が「0」です。 • パラメータの値が、使用されている CPU の最大許容 DB 番号よりも大きくなっています。
80B1	CPU に指定された番号の DB が存在しません。
80B4	CPU のメモリカードが書き込み禁止になっているため、DB を削除できません。
80B5	DB が 「CREATE_DB」を使用して作成されませんでした。
80BB	ロードメモリが不足しています。
80C3	一時的なリソースのボトルネックのため、この時点で「DB を削除」ファンクションは実行できません。
一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

アドレス指定

この章には下記に関する情報が記載されています：

- アドレス変換の命令 (S7-1200, S7-1500)
- GEO2LOG: スロットのハードウェア識別子の識別 (S7-1200, S7-1500)
- LOG2GEO: ハードウェア識別子からのスロットの識別 (S7-1200, S7-1500)
- LOG2MOD: STEP 7 V5.5 SPx のアドレス指定からのハードウェア識別子の識別 (S7-1500)
- IO2MOD: IO アドレスからのハードウェア識別子の識別 (S7-1200, S7-1500)
- RD_ADDR: ハードウェア識別子からの IO アドレスの識別 (S7-1200, S7-1500)
- システムデータタイプ GEOADDR (S7-1200, S7-1500)
- レガシー (S7-1200, S7-1500)

アドレス変換の命令

説明

モジュールのアドレス指定にはさまざまなオプションがあります(IO アドレス、ハードウェア識別子、スロット)。

以下の命令に従って、アドレスデータを変換することができます。

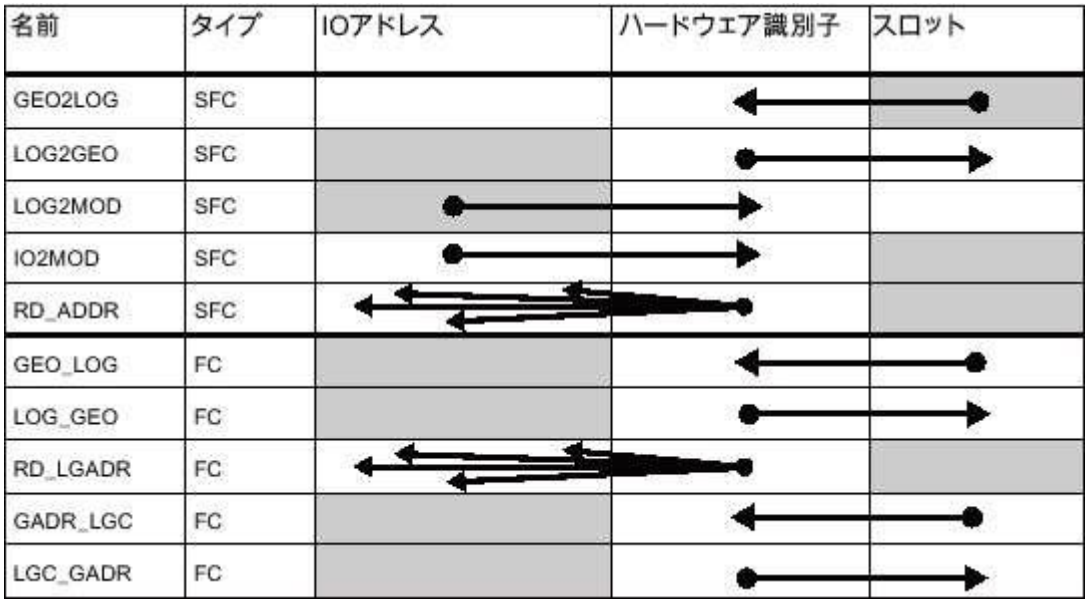
- GEO2LOG: スロットのハードウェア識別子の識別
- LOG2GEO: ハードウェア識別子からのスロットの識別
- LOG2MOD: STEP 7 V5.5 SPx のアドレス指定からのハードウェア識別子の識別
- IO2MOD: IO アドレスからのハードウェア識別子の識別
- RD_ADDR: ハードウェア識別子からの IO アドレスの識別

移行したプロジェクトの場合、さらに以下の命令がサポートされています。

- GEO_LOG: スロットのハードウェア識別子の識別
- LOG_GEO: ハードウェア識別子からのスロットの識別
- RD_LGADR: ハードウェア識別子からの IO アドレスの識別
- GADR_LGC: ユーザーデータアドレス領域のスロットおよびオフセットからのハードウェア識別子の識別
- LGC_GADR: ハードウェア識別子からのスロットの識別

アドレス変換のタイプ

以下の図は、アドレス変換を実行する命令を示しています。



GEO2LOG: スロットのハードウェア識別子の識別

説明

「GEO2LOG」命令を使用して、システムデータタイプ GEOADDR によって定義するスロット情報に基づいてハードウェア識別子を識別します。

HWTYPE パラメータで定義するハードウェアタイプに応じて、他のパラメータ GEOADDR の以下の情報が評価されます。

- HWTYPE = 1 の場合(IO システム):
  - IOSYSTEM のみが評価されます。 GEOADDR の他のパラメータは考慮されません。
  - IO システムのハードウェア識別子が出力されます。
- HWTYPE = 2 の場合(IO デバイス):
  - IOSYSTEM および STATION が評価されます。 GEOADDR の他のパラメータは考慮されません。
  - IO デバイスのハードウェア識別子が出力されます。
- HWTYPE = 4 (モジュール)の場合:
  - IOSYSTEM、STATION、および SLOT が評価されます。 GEOADDR の SUBSLOT パラメータは考慮されません。
  - モジュールのハードウェア識別子が出力されます。
- HWTYPE = 5 (サブモジュール)の場合:
  - GEOADDR の全パラメータが評価されます。
  - サブモジュールのハードウェア識別子が出力されます。

GEOADDR システムデータタイプの AREA パラメータは評価されません。

パラメータ

以下の表に、「GEO2LOG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
GEOADDR	Input	VARIANT	D、L	GEOADDR システムデータタイプの構造体へのポインタ。 このシステムデータタイプには、ハードウェア ID を識別するスロット情報が含まれています。 関連項目: システムデータタイプ GEOADDR
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報の出力。
LADDR	Output	HW_ANY	I、Q、M、D、L	アセンブリまたはモジュールのハードウェア識別子。 番号は自動的に割り当てられ、ハードウェアコンフィグレーションのプロパティに格納されます。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした。
8091	GEOADDR 内に、HWTYPE に対する無効な値が存在します。
8094	GEOADDR 内に、IOSYSTEM に対する無効な値が存在します。
8095	GEOADDR 内に、STATION に対する無効な値が存在します。
8096	GEOADDR 内に、SLOT に対する無効な値が存在します。
8097	GEOADDR 内に、SUBSLOT に対する無効な値が存在します。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

LOG2GEO: ハードウェア識別子からのスロットの識別

説明

「LOG2GEO」命令を使用し、ハードウェア識別子に属するモジュールスロットを判別します。

パラメータ

次の表に、「LOG2GEO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_ANY	I、Q、M、D、L、または定数	そのスロットを検索するモジュールのハードウェア識別子。 ハードウェア ID は自動的に割り当てられ、ハードウェアコンフィグレーションのモジュールのプロパティおよびシステム定数に格納されます。
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報の出力。
GEOADDR	InOut	VARIANT	D	GEOADDR システムデータタイプへのポインタ。 スロット情報は、システムデータタイプ GEOADDR に書き込まれます。 関連項目 システムデータタイプ GEOADDR

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした。
8090	LADDR パラメータで指定されたアドレスが無効です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

LOG2MOD: STEP 7 V5.5 SPx のアドレス指定からのハードウェア識別子の識別

説明

「LOG2MOD」命令を使用し、STEP 7 5.5 SPx (IO データアドレスまたは診断アドレス)のアドレス指定から IO (サブ)モジュールのハードウェア識別子を判別します。

ハードウェア識別子は LADDR 入力パラメータでさまざまな命令のアドレス指定に使用されます。先に「LOG2MOD」を呼び出すことで、STEP 7 5.5 SPx からアドレスパラメータを変換することができます。

パラメータ

以下の表に、「LOG2MOD」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IOID	Input	BYTE	I、Q、M、D、L、または定数	STEP 7 5.5 SPx でのアドレス領域の識別子: • B#16#00: ADDR のビット 15 で、入力(ビット 15=0)または出力アドレス(ビット 15=1)が存在するかどうかを指定します。 • B#16#54= 周辺機器入力(PI) • B#16#55= 周辺機器出力(PQ)
ADDR	Input	WORD	I、Q、M、D、L、または定数	オフセット(STEP 7 5.5 SPx での対応するアドレス指定)または診断アドレスとしてのモジュールの IO データの論理アドレス。
RET_VAL	Return	INT	I、Q、M、D、L	命令のエラーコード。
HWID	Output	HW_IO	I、Q、M、D、L	IO(サブ)モジュールの判別されたハードウェア識別子。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした。
8093	• 指定されたアドレスは、いずれのハードウェアコンポーネントにも使用されていません。 • IOID パラメータで指定された値が無効です。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

IO2MOD: IO アドレスからのハードウェア識別子の識別

説明

「IO2MOD」命令は、モジュールの IO アドレス(I、Q、PI、PQ)からモジュールのハードウェア識別子を判別します。

ADDR パラメータに IO アドレスを入力します。 このパラメータで複数の IO アドレスが使用されている場合、ハードウェア識別子の判別には最初のアドレスのみが評価されます。 最初のアドレスが正しく指定されている場合、ADDR でのアドレス指定の長さは問題とはなりません。複数のモジュールまたは使用されていないアドレスを含むアドレス領域が使用されている場合、最初のモジュールのハードウェア識別子も判別されます。

ADDR パラメータでモジュールの IO アドレスが指定されていない場合、パラメータ RET_VAL にエラーコード 8090 が出力されます。

注記

SCL での IO アドレスの入力

SCL では、IO アクセス ID「%QWx:P」を使用してプログラムできません。 この場合、プロセスイメージで、シンボルタグ名または絶対アドレスを使用します。

パラメータ

以下の表に、「IO2MOD」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
ADDR	Input	VARIANT	I、Q、M、D、L	モジュール内の IO アドレス(I、Q、PI、PQ)。 ADDR パラメータでスライスアクセスが使用されていないことを確認します。 スライスアクセスが使用された場合、不正な値が LADDR パラメータに出力されます。
RET_VAL	Return	INT	I、Q、M、D、L	命令のエラーコード。
LADDR	Output	HW_IO	I、Q、M、D、L	IO モジュールの判別されたハードウェア識別子(論理アドレス)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした。

8090	ADDR パラメータで指定された IO アドレスは、いずれのハードウェアコンポーネントにも使用されません。
8092	パラメータ ADDR で無効なデータタイプ(WCHAR または WSTRING など)。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

RD_ADDR: ハードウェア識別子からの IO アドレスの識別 

説明

「RD_ADDR」命令は、(サブ)モジュールのハードウェア識別子に基づいて入力または出力の長さおよび開始アドレスを判別します。

- LADDR パラメータを使用して、ハードウェア識別子に基づいて入力または出力モジュールを選択します。
- 入力モジュールか、または出力モジュールかに応じて、以下の出力パラメータが使用されます。
  - 入力モジュールの場合、判別された値は PIADDR および PICOUNT パラメータに出力されます。
  - 出力モジュールの場合、判別された値は PQADDR および PQCOUNT パラメータに出力されます。
- PIADDR および PQADDR パラメータは、それぞれモジュールの I/O アドレスの開始アドレスを含んでいます。
- PICOUNT および PQCOUNT パラメータは、それぞれ入力または出力のバイト数を含んでいます (入力/出力が 8 の場合は 1 バイト、入力/出力が 16 の場合は 2 バイト)。

注記

パックされたモジュールのアドレス指定

パック済みモジュール(パック済みモジュールグループの最初のモジュールではない)がアドレス指定され、表示データはハードウェアコンフィグレーションから外れます。 PIADDR または PQADDR および PICOUNT または PQCOUNT の場合、「0」が返されます。RET_VAL は、エラー(16#0000)を示しません。

パラメータ

次の表に、「RD_ADDR」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	(サブ)モジュールのハードウェア識別子。
RET_VAL	Return	INT	I、Q、M、D、L	命令のエラーコード。
PIADDR	Output	UDINT	I、Q、M、D、L	入力モジュールの開始アドレス。
PICOUNT	Output	UINT	I、Q、M、D、L	入力のバイト数。
PQADDR	Output	UDINT	I、Q、M、D、L	出力モジュールの開始アドレス。
PQCOUNT	Output	UINT	I、Q、M、D、L	出力のバイト数。

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ RET_VAL

エラーコード* (W#16#...)	説明
0	エラーは発生しませんでした。
8090	LADDR パラメータのモジュールのハードウェア識別子が不正です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

システムデータタイプ GEOADDR

領域アドレス

システムデータタイプ GEOADDR には、モジュールの領域アドレス、すなわち、スロット情報が収納されます。

- PROFINET IO の領域アドレス

  PROFINET IO の場合、領域アドレスは、PROFINET IO システムの ID、デバイス番号、スロット番号、およびサブモジュール(サブモジュールを使用する場合)から構成されました。

- PROFIBUS DP の領域アドレス

  PROFIBUS DP の場合、領域アドレスは、DP マスタシステムの ID、ステーション番号、およびスロット番号から構成されます。

モジュールのスロット情報は、各モジュールのハードウェアコンフィグレーションに存在します。

システムデータタイプ GEOADDR

データブロックでデータタイプとして「GEOADDR」を入力すると、構造体 GEOADDR が自動的に作成されます。

システムデータタイプ GEOADDR の構造を以下に示します。

パラメータ名	データタイプ	説明
GEOADDR	STRUCT	
HWTYPE	UINT	ハードウェアタイプ: • 1: IO システム(PROFINET/PROFIBUS) • 2: IO デバイス/DP スレーブ • 3: Rack • 4: モジュール • 5: サブモジュール ハードウェアタイプが命令によってサポートされていない場合、HWTYPE「0」が出力されます。
AREA	UINT	領域 ID: • 0 = CPU • 1 = PROFINET IO • 2 = PROFIBUS DP • 3 = AS-i
IOSYSTEM	UINT	PROFINET IO システム(0 = ラックの基本ユニット)
STATION	UINT	• 領域識別子 AREA= 0 (基本モジュール)の場合、ラック番号。 • 領域識別子 AREA > 0 の場合、ステーション番号。
SLOT	UINT	スロット番号
SUBSLOT	UINT	サブモジュール数。 サブモジュールが使用できないか、接続できない場合、このパラメータの値は「0」です。

レガシー

この章には下記に関する情報が記載されています：

- GEO_LOG: スロットのハードウェア識別子の識別 (S7-1500)
- LOG_GEO: ハードウェア識別子からのスロットの識別 (S7-1500)
- RD_LGADR: ハードウェア識別子からの IO アドレスの識別 (S7-1200, S7-1500)
- GADR_LGC: ユーザーデータアドレス領域のスロットおよびオフセットからのハードウェア識別子の識別 (S7-1500)
- LGC_GADR: ハードウェア識別子からのスロットの識別 (S7-1500)

GEO_LOG: スロットのハードウェア識別子の識別

説明

シグナルモジュールの対応モジュールスロットが既知です。「GEO_LOG」命令を使用し、シグナルモジュールのチャンネルからモジュールの対応するハードウェア識別子を判別します。

パラメータ

次の表に、「GEO_LOG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MASTER	Input	INT	I、Q、M、D、L、または定数	領域 ID: • 0、集中型構成内にスロットがある場合。 • 1～32: スロットが PROFIBUS 上のフィールドデバイスにある場合、関連するフィールドデバイスの DP マスタシステム ID • 100～115: スロットが PROFINET 上のフィールドデバイスにある場合、関連するフィールドデバイスの PROFINET IO マスタシステム ID
STATION	Input	INT	I、Q、M、D、L、または定数	• MASTER = 0 の場合: ラック番号 • MASTER > 0 の場合: フィールドデバイスのステーション番号
SLOT	Input	INT	I、Q、M、D、L、または定数	スロット番号
SUBSLOT	Input	INT	I、Q、M、D、L、または定数	SUBSLOT この命令はパラメータを評価しません。
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
LADDR	Output	HW_IO	I、Q、M、D、L	モジュールのハードウェア識別子

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8094	指定された SUBNETID で構成されたサブネットがありません。
8095	STATION パラメータの無効な値
8096	SLOT パラメータの無効な値

一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

LOG_GEO: ハードウェア識別子からのスロットの識別

説明

「LOG_GEO」命令を使用し、ハードウェア識別子に属するモジュールスロットを判別します。

パラメータ

次の表に、「LOG_GEO」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
LADDR	Input	HW_IO	I、Q、M、D、L、または定数	スロットが識別されるモジュールのハードウェア識別子。
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
AREA	Output	INT	I、Q、M、D、L	領域 ID: 残りの出力パラメータの解釈方法を示します。 • 0: 基本デバイス • 2: PROFIBUS DP / PROFINET IO
MASTER	Output	INT	I、Q、M、D、L	AREA = 0 の場合: • 0: ラックのいずれかにスロットがある場合(基本デバイス)。 AREA = 2 の場合: • 1 ~ 32: スロットが PROFIBUS 上のフィールドデバイスにある場合、関連するフィールドデバイスの DP マスタシステム ID • 100 ~ 115: スロットが PROFINET 上のフィールドデバイスにある場合、関連するフィールドデバイスの PROFINET IO マスタシステム ID
STATION	Output	INT	I、Q、M、D、L	• MASTER = 0 の場合: ラック番号 • MASTER > 0 の場合: フィールドデバイスのステーション番号
SLOT	Output	INT	I、Q、M、D、L	スロット番号
SUBSLOT	Output	INT	I、Q、M、D、L	この命令は SUBSLOT パラメータを出力しません(常時「0」)。
OFFSET	Output	INT	I、Q、M、D、L	この命令は OFFSET パラメータを出力しません(常時「0」)。

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	指定された論理アドレスが無効
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

RD_LGADR: ハードウェア識別子からの IO アドレスの識別

説明

「RD_LGADR」命令を使用して、ハードウェア識別子に基づいて、モジュール、中央サブモジュールまたは PNIO 用サブモジュールの論理アドレスを確定できます。

- LADDR パラメータでサブモジュールのハードウェア ID を指定できます。
- アドレスは PEADDR および PAADDR パラメータに昇順で書き込まれます。
  - 入力モジュールの場合、PEADDR パラメータにのみ書き込まれます。 出力モジュールの場合、PAADDR パラメータに書き込まれます。
  - いずれの場合にも、アドレスを格納するために Array of WORD が使用されます。
- アドレスの数は、PECOUNT パラメータ(入力モジュールの場合)および PACOUNT パラメータ(出力モジュールの場合)で出力されます。

パラメータ

次の表に、「RD_LGADR」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IOID	Input	BYTE	I、Q、M、D、L、または定数	アドレス領域識別子: • B#16#54 = 周辺機器入力(PI) • B#16#55 = 周辺機器出力(PQ)
LADDR	Input	HW_ANY	I、Q、M、D、L、または定数	モジュールまたはサブモジュールのハードウェア識別子。
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
PEADDR	Output	ANY	I、Q、M、D、L	Array of WORD データタイプの PI アドレス用のフィールド
PECOUNT	Output	INT	I、Q、M、D、L	返される PI アドレスの数
PAADDR	Output	ANY	I、Q、M、D、L	Array of WORD データタイプの PQ アドレス用のフィールド
PACOUNT	Output	INT	I、Q、M、D、L	返される PQ アドレスの数

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	指定された論理アドレスが無効であるか、IOID パラメータの無効な値です。

80A0	出力パラメータ PEADDR のエラー: 配列エレメントのデータタイプが WORD ではありません。
80A1	出力パラメータ PAADDR のエラー: 配列エレメントのデータタイプが WORD ではありません。
80A2	出力パラメータ PEADDR のエラー: 指定された配列にすべての論理アドレスを収めることができませんでした。
80A3	出力パラメータ PAADDR のエラー: 指定された配列にすべての論理アドレスを収めることができませんでした。
一般エラー情報	関連項目 GET_ERR_ID: ローカルでエラー ID を取得

GADR_LGC: ユーザーデータアドレス領域のスロットおよびオフセットからのハードウェア識別子の識別

説明

「GADR_LGC」命令を使用して、シグナルモジュールのハードウェア識別子を判別します。 ハードウェア識別子は、モジュールスロットおよびモジュールのユーザーデータアドレス領域のオフセットから識別されます。

注記

診断アドレスの出力

「GADR_LGC」命令を電源モジュールまたは PACK アドレスを持つモジュールに対して使用すると、診断アドレスが返されます。

注記

使用の制限

「GADR_LGC」命令はゲートウェイの後にあるモジュールには使用できません(IE/PB リンクなど)。この命令の代わりに「GEO2LOG」命令を使用してください。

パラメータ

以下の表に、「GADR_LGC」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
SUBNETID	Input	BYTE	I、Q、M、D、L、または定数	領域 ID: • 0: 基本モジュール内にスロットがある場合。 • 1～32: 対応するリモート I/O システム DP マスタシステム ID、スロットがリモート I/O デバイスにある場合。 • 100～115:スロットが PROFINET 上のフィールドデバイスにある場合、関連するフィールドデバイスの PROFINET IO マスタシステム ID
RACK	Input	WORD	I、Q、M、D、L、または定数	• ラック番号、領域識別子が 0 の場合。 • リモート I/O デバイスのデバイス番号、領域識別子> 0 の場合
SLOT	Input	WORD	I、Q、M、D、L、または定数	スロット番号
SUBSLOT	Input	BYTE	I、Q、M、D、L、または定数	サブモジュールスロット(サブモジュールを挿入できない場合、ここに 0 を入力する必要があります)
SUBADDR	Input	WORD	I、Q、M、D、L、または定数	モジュールのユーザーデータアドレス領域のオフセット

RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
IOID	Output	BYTE	I、Q、M、D、L	IOID 出力パラメータは書き込まれません(常時「0」)。
LADDR	Output	HW_MOD-ULE	I、Q、M、D、L	モジュールのハードウェア識別子

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8093	SUBNETID パラメータの無効な値
8094	指定された SUBNETID で構成されたサブネットがありません。
8095	RACK パラメータの無効な値。
8096	SLOT パラメータの無効な値。
8097	SUBSLOT パラメータの無効な値。
8098	SUBADDR パラメータの無効な値。
8099	スロットが未設定です。
809A	選択したスロットのサブアドレスが設定されていません(CPU および IM の集中 I/O でのみ可能です)。
一般エラー情報	関連項目:GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

LGC_GADR: ハードウェア識別子からのスロットの識別

説明

「LGC_GADR」命令を使用し、ハードウェア識別子に属するモジュールスロットを判別します。

注記
「LGC_GADR」命令は、パックされたアドレスを持つモジュール(ET 200S)では使用できません。

注記

使用の制限

「LGC_GADR」命令はゲートウェイの後にあるモジュールには使用できません(IE/PB リンクなど)。この命令の代わりに「LOG2GEO」命令を使用してください。

パラメータ

以下の表に、「LGC_GADR」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
IOID	Input	BYTE	I、Q、M、D、L、または定数	評価されません。
LADDR	Input	HW_MOD-ULE	I、Q、M、D、L、または定数	モジュールのハードウェア識別子
RET_VAL	Return	INT	I、Q、M、D、L	エラー情報
AREA	Output	BYTE	I、Q、M、D、L	領域 ID: 残りの出力パラメータの解釈方法を示します。 • 0: セントラルモジュール • 2: PROFIBUS DP
RACK	Output	WORD	I、Q、M、D、L	ラック番号: • 基本モジュール(AREA= 0)の場合: ○ ラック番号 • PROFIBUS DP (AREA=2)の場合: ○ 下位バイト: ステーション番号 ○ 上位バイト: DP マスタシステム ID
SLOT	Output	WORD	I、Q、M、D、L	スロット番号: • 基本モジュール(AREA= 0)の場合: ○ スロット番号 • PROFIBUS DP (AREA=2)の場合:

				○ ステーションのスロット番号
SUBADDR	Output	WORD	I、Q、M、D、L	出力されません(常時「0」)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード* (W#16#...)	説明
0000	エラーは発生しませんでした。
8090	指定された論理アドレスが無効であるか、IOID パラメータの無効な値です。
8093	この命令は、パラメータ IOID および LADDR で選択されたモジュールでは許可されません。
一般エラー情報	関連項目:GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

テクノロジー

この章には下記に関する情報が記載されています：

- モーションコントロール (S7-1200, S7-1500)
- 高速カウンタ (S7-1200)
- カウントと測定 (S7-1500)
- PID 制御 (S7-1200)
- (S7-1500)

モーションコントロール

この章には下記に関する情報が記載されています：

- S7-1200 モーションコントロール (S7-1200)
- S7-1500 モーションコントロール (S7-1500)

S7-1200 モーションコントロール

この章には下記に関する情報が記載されています：

- V4 以降の S7-1200 モーションコントロール (S7-1200)
- S7-1200 モーションコントロール V1...3 (S7-1200)

V4 以降の S7-1200 モーションコントロール

この章には下記に関する情報が記載されています：

- MC_Power (S7-1200)
- MC_Reset (S7-1200)
- MC_Home (S7-1200)
- MC_Halt (S7-1200)
- MC_MoveAbsolute (S7-1200)
- MC_MoveRelative (S7-1200)
- MC_MoveVelocity (S7-1200)
- MC_MoveJog (S7-1200)
- MC_CommandTable (S7-1200)
- MC_ChangeDynamic (S7-1200)
- MC_ReadParam (S7-1200)
- MC_WriteParam (S7-1200)

MC_Power

この章には下記に関する情報が記載されています：

- MC_Power: V4 以降の軸の有効化、無効化 (S7-1200)
- MC_Power: V4 以降のファンクションチャート (S7-1200)

MC_Power: V4 以降の軸の有効化、無効化

説明

モーションコントロール命令「MC_Power」は、軸を有効/無効にします。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 未解決のイネーブル禁止エラーがないこと。

応答の無効化

「MC_Power」の実行は、モーションコントロールコマンドによって中止できません。

軸の無効化(入力パラメータ「Enable」 = FALSE)は、選択した「StopMode」に従って、関連するテクノロジーオブジェクトに対するすべてのモーションコントロールコマンドを中止します。

パラメータ

パラメータ	宣言	データタイプ	既定値	説明	
Axis	INPUT	TO_Axis	-	軸テクノロジーオブジェクト	
Enable	INPUT	BOOL	FALSE	TRUE	軸が有効になります。
				FALSE	設定された「StopMode」に従ってすべての現在のジョブが中止されます。 軸は停止し、無効になっています。
Stop-Mode	INPUT	INT	0	0	緊急停止 軸を無効にする要求が保留中の場合、設定された緊急減速で、軸が制動します。軸は停止後に無効になります。
				1	直ちに停止 軸を無効にする要求が保留中の場合、このセットポイントゼロが出力され、軸は無効になります。軸はドライブでの設定に応じて制動し、停止します。 PTO (Pulse Train Output)経由のドライブ接続: 軸を無効にすると、周波数依存の減速度でパルス出力が停止します。 • 出力周波数 ≥ 100 Hz 減速: 最大 30 ミリ秒 • 出力周波数< 100 Hz 減速: 30 ミリ秒 ~ 最大 1.5 秒(2 Hz)

				2	ジャーク制御による緊急停止 軸を無効にする要求が保留中の場合、設定された緊急減速で、軸が制動します。 ジャーク制御が有効な場合、設定されたジャークが考慮されます。 軸は停止後に無効になります。
Status	OUTPUT	BOOL	FALSE	軸のイネーブルのステータス	
				FALSE	軸が無効になります。 軸はモーションコントロールコマンドを実行せず、すべての新しいコマンドの承認も行いません(例外: MC_Reset コマンド)。 PTO (Pulse Train Output)経由のドライブ接続: 軸が原点復帰していません。 無効時、軸が静止するまでは、ステータスは FALSE に変わりません。
				TRUE	軸が有効になります。 軸は、モーションコントロールコマンドを実行できる状態です。 軸が有効なときは、信号「ドライブ準備完了」が保留中になるまで、ステータスは TRUE に変わりません。 軸の構成で「ドライブ準備完了」ドライブインターフェースが設定されなかった場合、ステータスは直ちに TRUE に切り替わります。
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" が動作中。
Error	OUTPUT	BOOL	FALSE	TRUE	モーションコントロール命令「MC_Power」、または関連するテクノロジーオブジェクトでエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の[エラー ID]	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の[エラー情報 ID]	

注記
軸がエラーのためにスイッチがオフになった場合、エラーを除去し、確認した後は、軸が自動的に有効になります。 この場合、このプロセス時に、入力パラメータ「Enable」が値 TRUE を保持していることが必要です。

ドライブインターフェースが設定された軸の有効化

この軸を有効にするには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 入力パラメータ「StopMode」を必要な値で初期化します。 入力パラメータ「Enable」を TRUE に設定します。

「ドライブ有効化済み」のための許可出力が TRUE に変わり、ドライブへの電源を有効にします。 CPU は、ドライブの「ドライブ準備完了」信号を待機します。

CPU の設定された準備完了入力で「ドライブ準備完了」信号が使用可能になると、軸が有効になります。 出力パラメータ「Status」およびテクノロジーオブジェクト<軸名>.StatusBits.Enable のタグは、真の値を示します。

ドライブインターフェースが設定されていない軸の有効化

この軸を有効にするには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 入力パラメータ「StopMode」を必要な値で初期化します。 入力パラメータ「Enable」を TRUE に設定します。 軸が有効になります。 出力パラメータ「Status」およびテクノロジーオブジェクト<軸名>.StatusBits.Enable のタグは、真の値を示します。

軸の無効化

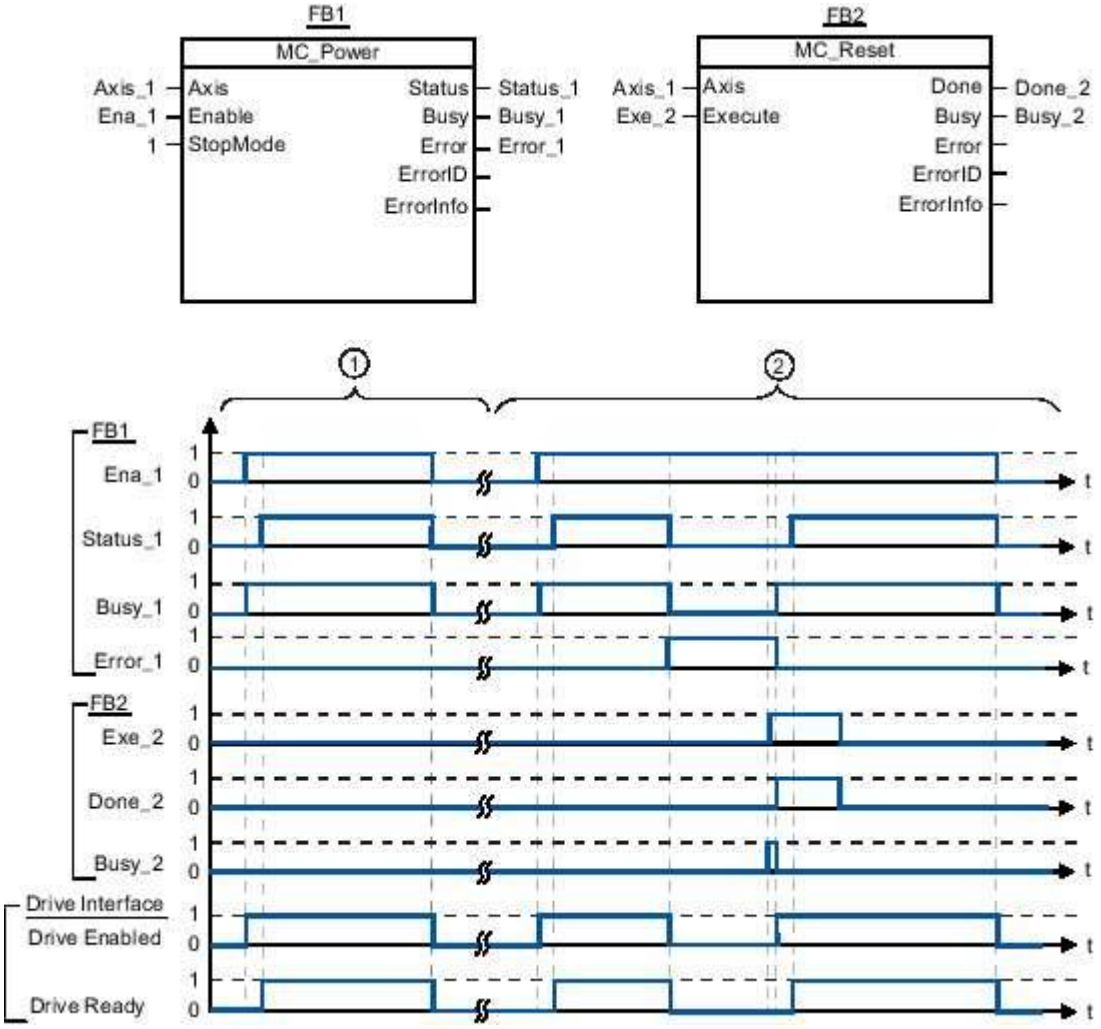
軸を無効にするには、以下の手順を実行します。

1. 軸を停止します。

   軸の停止は、テクノロジーオブジェクト<軸名>.StatusBits.StandStill のタグで識別できます。
2. 停止した後、入力パラメータ「Enable」を FALSE に設定します。
3. 出力パラメータ「Busy」および「Status」、およびテクノロジーオブジェクト<軸名>.StatusBits.Enable のタグは偽の値を示し、軸の無効化が完了します。

MC_Power: V4 以降のファンクションチャート

ファンクションチャート



①	軸が有効になった後、再び無効になります。 ドライブが「ドライブ準備完了」信号を CPU に戻すと、成功したイネーブルを「Status_1」経由で読み出すことができます。
②	軸イネーブルに続いて、軸の無効化の原因となる エラーが発生しました。 エラーが除去され、「MC_Reset」で確認されます。 この場合、軸は再び有効になります。

MC_Reset

この章には下記に関する情報が記載されています：

- MC_Reset: V4 以降の故障確認応答 (S7-1200)

MC_Reset: V4 以降の故障確認応答

説明

モーションコントロール命令「MC_Reset」を使用して、「軸停止ありの動作エラー」および「構成エラー」を確認することができます。 確認が必要なエラーは、「対策」の「ErrorID および ErrorInfo のリスト」に記載されています。

RUN モードでのダウンロード後に、軸設定をワークメモリにダウンロードできます。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 確認が必要な保留中の構成エラーの原因が除去済みであること(たとえば、位置決め軸テクノロジーオブジェクトの加速が有効な値に変更済みであること)。

応答の無効化

MC_Reset コマンドは、他のモーションコントロールコマンドで中止することはできません。

新しい MC_Reset コマンドは、他のすべての動作中のモーションコントロールコマンドを中止しません。

パラメータ

パラメータ	宣言	データタイプ	既定値	説明	
Axis	INPUT	TO_Axis	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
Restart	INPUT	BOOL	FALSE	TRUE	軸設定をロードメモリからワークメモリにダウンロードします。 このコマンドは、軸が無効になっているときのみ実行できます。 CPU へのダウンロードに関する注記を参照してください。
				FALSE	保留中のエラーを確認します
Done	OUTPUT	BOOL	FALSE	TRUE	エラーが確認応答済みです。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

エラーの確認には、MC_Reset による確認が必要です。

エラーを確認するには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 入力パラメータ「Execute」での立ち上がりエッジで、エラーの確認応答を開始します。
3. 出力パラメータ「Done」が真の値を示した場合やテクノロジーオブジェクト<軸名>.StatusBits.Error のタグが偽の値を示した場合、エラーが確認応答済みです。

MC_Home

この章には下記に関する情報が記載されています：

- MC_Home: V4 以降の原点復帰、基準点の設定 (S7-1200)

MC_Home: V4 以降の原点復帰、基準点の設定

説明

モーションコントロール命令「MC_Home」を使用して、軸座標を物理的な実際のドライブ位置に適合させます。 軸の絶対位置決めには、原点復帰が必要です。 次のタイプの原点復帰を実行できます。

- アクティブ原点復帰(Mode = 3)

  原点復帰手順が自動的に実行されます。
- パッシブ原点復帰(Mode = 2)

  パッシブ原点復帰時には、モーションコントロール命令「MC_Home」は、原点復帰モーションを実行しません。 このステップに必要なトラベルは、他のモーションコントロール命令で、ユーザーによって実装される必要があります。 原点復帰スイッチが検出されると、軸が原点復帰します。
- 絶対直接原点復帰(Mode = 0)

  現在の軸位置が、パラメータ「Position」の値に設定されます。
- 相対直接原点復帰(Mode = 1)

  現在の軸位置が、パラメータ「Position」の値だけオフセットされます。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 軸が有効であること。
- Mode = 0、1、または 2 での開始時に、MC_CommandTable コマンドが動作中でないこと。

応答の無効化

応答の無効化は、選択したモードに応じて異なります。

Mode = 0, 1

MC_Home コマンドは、他のモーションコントロールコマンドで中止することはできません。

MC_Home コマンドは、他のすべての動作中のモーションコントロールコマンドを中止しません。 位置関連のモーションコマンドは、新しい原点復帰位置(入力パラメータ: 「Position」での値)に従って原点復帰した後、再開されます。 "Position").

Mode = 2

MC_Home コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 2, 3

新しい MC_Home コマンドは、次の有効なモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 2

位置関連のモーションコマンドは、新しい原点復帰位置(入力パラメータ: 「Position」での値)に従って原点復帰した後、再開されます。 "Position").

Mode = 3

MC_Home コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3

- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_Home コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 2, 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

パラメータ

パラメータ	宣言	データタイプ	既定値	説明	
Axis	INPUT	TO_Axis	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Position	INPUT	REAL	0.0	• Mode = 0、2、および 3 原点復帰動作を完了した後の絶対位置 • Mode = 1 現在の軸位置用補正値 制限値: $-1.0e^{12} \leq Position \leq 1.0e^{12}$	
Mode	INPUT	INT	0	原点復帰モード	
				0	直接原点復帰(絶対値) 新しい軸位置は、パラメータ「Position」の値です。
				1	直接原点復帰(相対値) 新しい軸位置は、現在の軸位置 + パラメータ「Position」の値です。
				2	パッシブ原点復帰 軸の構成に従った原点復帰。 原点復帰した後、パラメータ「Position」の値が新しい軸位置として設定されます。
				3	アクティブ原点復帰

					軸の設定に従った原点復帰手順です。 原点復帰した後、パラメータ「Position」の値が新しい軸位置として設定されます。
Done	OUTPUT	BOOL	FALSE	TRUE	コマンドが完了しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

「原点復帰済み」ステータスのリセット

テクノロジーオブジェクトの「原点復帰済み」ステータス(<AxisName>.StatusBits.HomingDone)は、以下の条件でリセットされます。

- **PTO (Pulse Train Output) 経由のドライブ接続:**
  - アクティブ原点復帰の[MC_Home]コマンドを開始します。

    (原点復帰動作の正常終了後、「原点復帰済み」ステータスはリセットされます。)
  - 「MC_Power」モーションコントロール命令による軸の無効化
  - 自動モードと手動コントロールの切り替え
  - CPU の電源オフ -> 電源オンの後
  - CPU 再起動(RUN-STOP -> STOP-RUN)後
- **現在の増分値を持つテクノロジーオブジェクト:**
  - アクティブ原点復帰の[MC_Home]コマンドを開始します。

    (原点復帰動作の正常終了後、「原点復帰済み」ステータスはリセットされます。)
  - エンコーダシステムのエラー、またはエンコーダ障害
  - テクノロジーオブジェクトの再起動
  - CPU の電源オフ → 電源オンの後
  - メモリリセット
  - エンコーダ設定の変更
- **絶対現在値を持つテクノロジーオブジェクト:**
  - CPU 工場設定の復元
  - エンコーダ設定の変更
  - CPU の交換

軸の原点復帰

軸を原点復帰するには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 必要な入力パラメータと値を提供し、「Execute」入力パラメータでの立ち上がりエッジで原点復帰動作を開始します。
3. 「Done」出力パラメータおよびテクノロジーオブジェクトタグ<軸名>.StatusBits.HomingDone が真の値を示す場合、原点復帰が完了しています。

MC_Halt

この章には下記に関する情報が記載されています：

- MC_Halt: V4 以降の軸の停止 (S7-1200)
- MC_Halt: V4 以降のファンクションチャート (S7-1200)

MC_Halt: V4 以降の軸の停止

説明

「MC_Halt」モーションコントロール命令は、すべての動きを停止し、設定された減速で軸を停止状態にします。 停止位置は定義されていません。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 軸が有効であること。

応答の無効化

MC_Halt コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_Halt コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

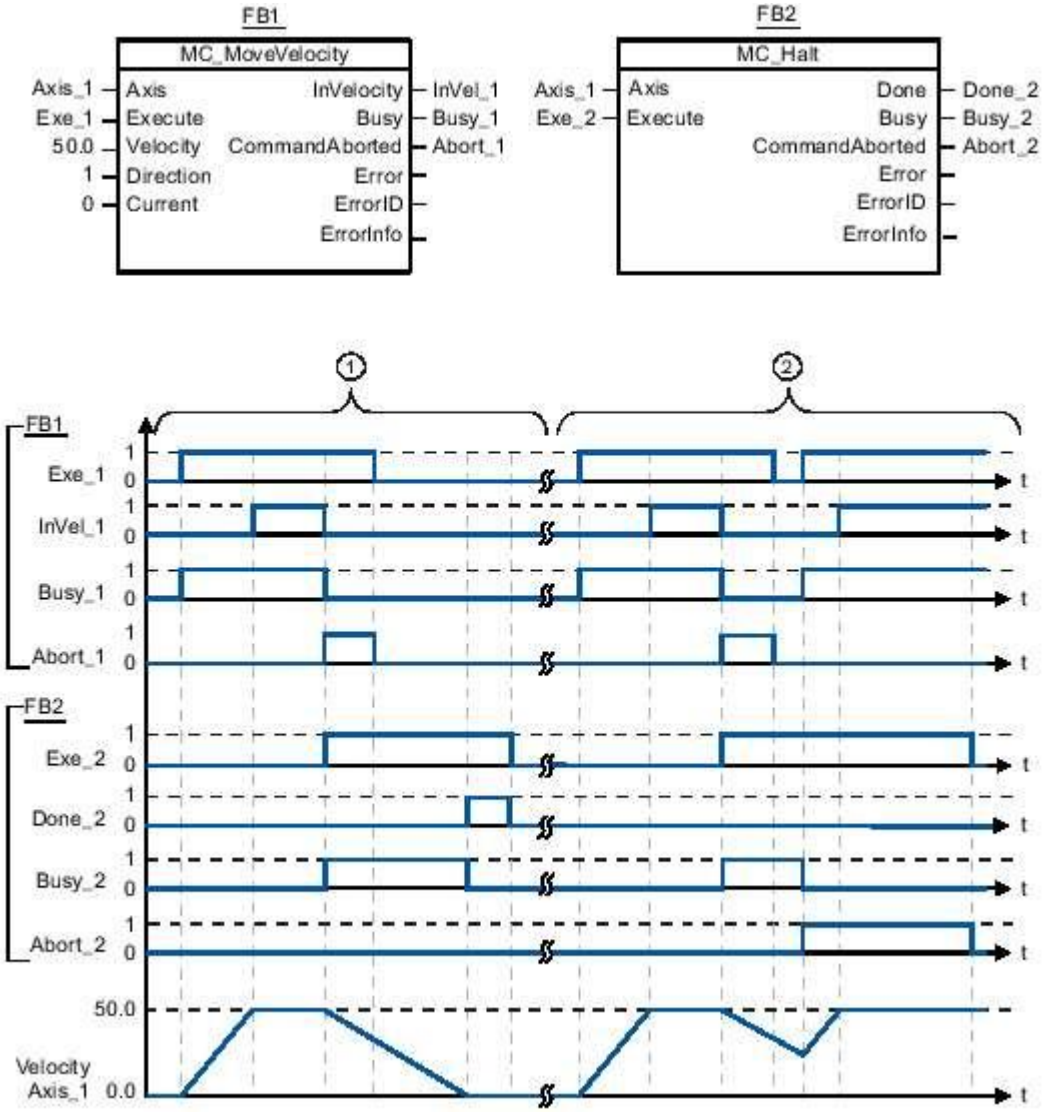
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_Spee-dAxis	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
Done	OUTPUT	BOOL	FALSE	TRUE	速度 0 に達しました。

Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

MC_Halt: V4 以降のファンクションチャート

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 5.0

①	軸は、停止状態になるまで、MC_Halt コマンドによって制動します。 軸の停止は、「Done_2」によって通知されます。
②	MC_Halt コマンドが軸を制動している途中でも、このコマンドは別のモーションコマンドによって中止されます。 この中止は、「Abort_2」によって通知されます。

MC_MoveAbsolute

この章には下記に関する情報が記載されています：

- MC_MoveAbsolute: V4 以降の軸の絶対位置決め (S7-1200)
- MC_MoveAbsolute: V4 以降のファンクションチャート (S7-1200)

MC_MoveAbsolute: V4 以降の軸の絶対位置決め

説明

「MC_MoveAbsolute」モーションコントロール命令は、軸位置決めモーションを開始し、軸を絶対位置まで移動します。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 軸が有効であること。
- 軸が原点復帰していること。

応答の無効化

MC_MoveAbsolute コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_MoveAbsolute コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

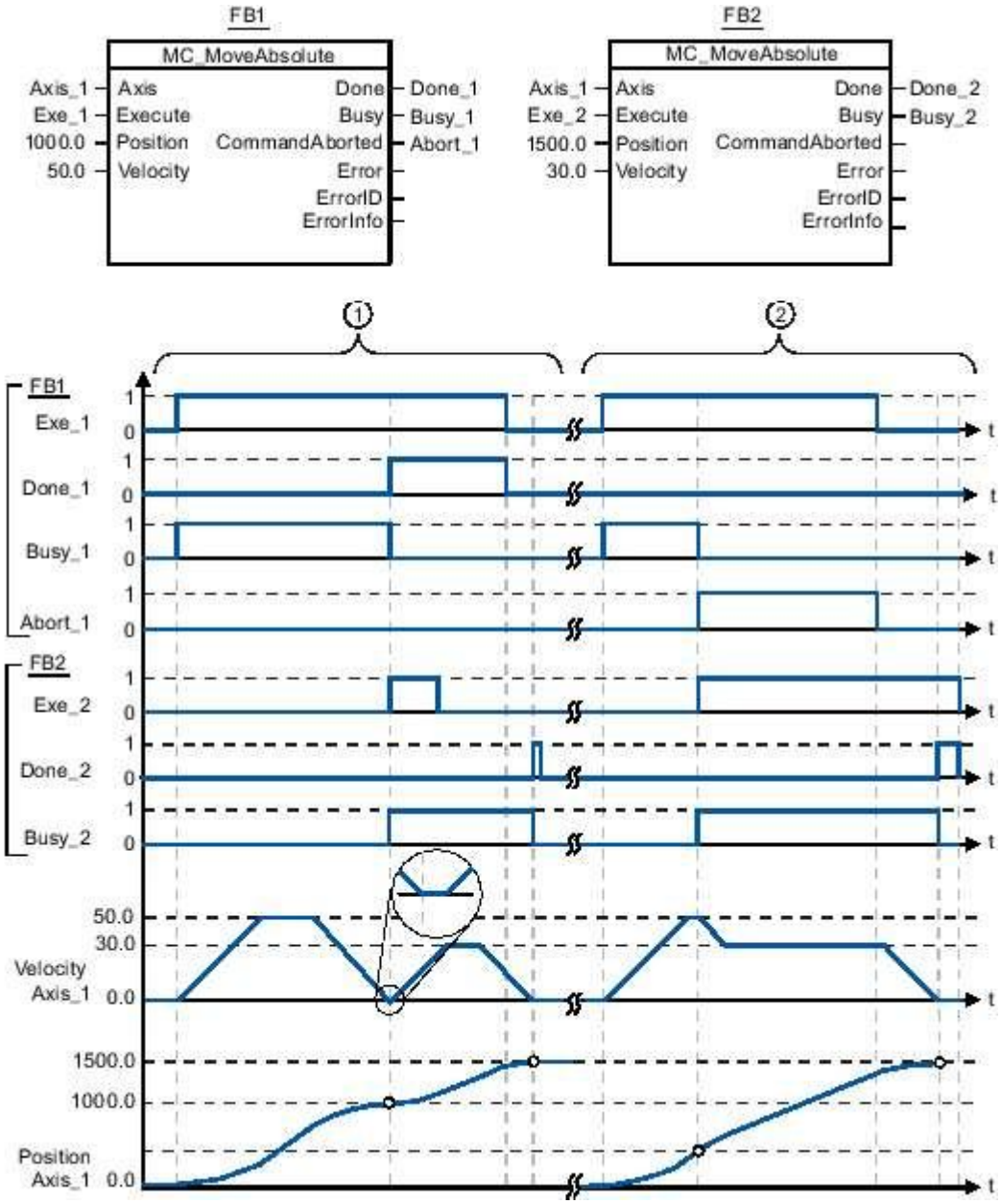
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_Position-ingAxis	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始

Position	INPUT	REAL	0.0	絶対ターゲット位置 制限値: $-1.0e^{12} \leq Position \leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	軸の速度 設定された加速および減速、およびアプローチするターゲット位置のために、常にこの速度に達するとは限りません。 制限値: 開始/停止速度 ≤ Velocity ≤ 最大速度	
Done	OUTPUT	BOOL	FALSE	TRUE	絶対ターゲット位置に到達しました
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

MC_MoveAbsolute: V4 以降のファンクションチャート

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 10.0

①	軸は、MC_MoveAbsolute コマンドによって絶対位置 1000.0 まで移動します。 軸がターゲット位置に到達すると、「Done_1」によって通知されます。 「Done_1」 = TRUE の場合、ターゲット位置が 1500.0 の別の MC_MoveAbsolute コマンドが開始されます。応答時間(たとえば、ユーザープログラムのサイクルタイムなど)のために、軸はすぐに停止状態になります(拡大された詳細図を参照)。 軸が新しいターゲット位置に到達すると、「Done_2」によって通知されます。
②	動作中の MC_MoveAbsolute コマンドは、別の MC_MoveAbsolute コマンドによって中止されます。 この中止は、「Abort_1」によって通知されます。 この後、この軸は、新しい速度で新しいターゲット位置 1500.0 まで移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveRelative

この章には下記に関する情報が記載されています：

- MC_MoveRelative: V4 以降の軸の相対位置決め (S7-1200)
- MC_MoveRelative: V4 以降のファンクションチャート (S7-1200)

MC_MoveRelative: V4 以降の軸の相対位置決め

説明

「MC_MoveRelative」モーションコントロール命令は、開始位置を基準にした位置決めモーションを開始します。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 軸が有効であること。

応答の無効化

MC_MoveRelative コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_MoveRelative コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

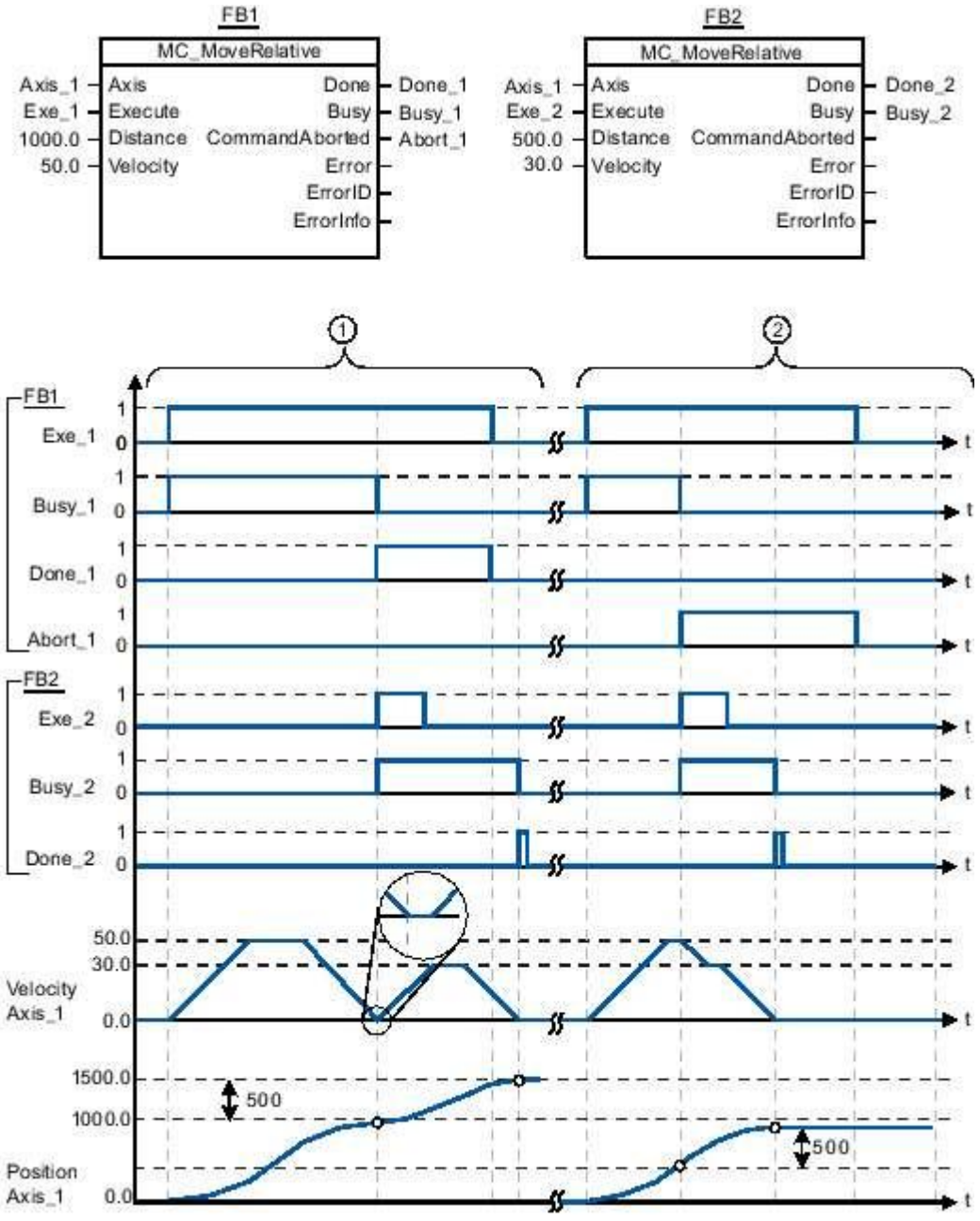
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_Position-ingAxis	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始
Distance	INPUT	REAL	0.0	位置決め動作での移動距離

				制限値: -1.0e^{12} ≤ Distance ≤ 1.0e^{12}	
Velocity	INPUT	REAL	10.0	軸の速度 設定された加速および減速、移動する距離のために、常にこの速度に達するとは限りません。 制限値: 開始/停止速度 ≤ Velocity ≤ 最大速度	
Done	OUTPUT	BOOL	FALSE	TRUE	ターゲット位置に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorIn-fo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

MC_MoveRelative: V4 以降のファンクションチャート

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 10.0

①	軸は、MC_MoveRelative コマンドによって、距離(「Distance」)1000.0 だけ移動します。 軸がターゲット位置に到達すると、「Done_1」によって通知されます。 「Done_1」 = TRUE の場合、移動距離が 500.0 の別の MC_MoveRelative コマンドが開始されます。応答時間(たとえば、ユーザープログラムのサイクルタイムなど)のために、軸はすぐに停止状態になります(拡大された詳細図を参照)。 軸が新しいターゲット位置に到達すると、「Done_2」によって通知されます。
②	動作中の MC_MoveRelative コマンドは、別の MC_MoveRelative コマンドによって中止されます。 この中止は、「Abort_1」によって通知されます。 この後、軸は、新しい速度で新しい距離(「Distance」) 500.0 だけ移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveVelocity

この章には下記に関する情報が記載されています：

- MC_MoveVelocity: V4 以降の軸をあらかじめ設定された回転速度で移動 (S7-1200)
- MC_MoveVelocity: V4 以降のファンクションチャート (S7-1200)

MC_MoveVelocity: V4 以降の軸をあらかじめ設定された回転速度で移動

説明

モーションコントロール命令「MC_MoveVelocity」は、軸を常に指定された速度で移動します。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 軸が有効であること。

応答の無効化

MC_MoveVelocity は、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

この新しい MC_MoveVelocity コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_Speed-Axis	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始
Velocity	INPUT	REAL	10.0	軸モーションの速度指定 制限値:

				開始/停止速度 ≤ |Velocity ≤ 最大速度 (Velocity = 0.0 は許可されます)	
Direction	INPUT	INT	0	方向指定	
				0	回転方向は、パラメータ「Velocity」の値の符号に対応します。
				1	正転の回転方向 (パラメータ「Velocity」の値の符号は無視されます)
				2	逆転の回転方向 (パラメータ「Velocity」の値の符号は無視されます)
Current	INPUT	BOOL	FALSE	現在速度の維持	
				FALSE	「現在速度の維持」が無効になります。 パラメータ「Velocity」および「Direction」が使用されます。
				TRUE	「現在速度の維持」が有効になります。 パラメータ「Velocity」および「Direction」は考慮されません。 軸が現在の速度でモーションを再開すると、「InVelocity」パラメータは値 TRUE *を返します。*
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: パラメータ「Velocity」で指定された速度に達しました。 • "Current" = TRUE: 軸は、開始時の現在速度で移動します。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

注記

PLCopen バージョン 2.0

モーションコントロール命令「MC_MoveVelocity」は、V4 以降の PLCopen バージョン 2.0 と互換性があります。

「InVelocity」および「Busy」パラメータは、「Execute」パラメータに関係なく、コマンドがオーバーライドされるかエラーで停止されるまでステータスを表示します。 詳細は、セクション「**有効なコマンドのトラッキング**」を参照してください。

ゼロセットポイント速度の場合の動作(Velocity = 0.0)

「Velocity」 = 0.0 の MC_MoveVelocity コマンド(MC_Halt コマンドと同様に)は、動作中のモーションコマンドを中止、設定された減速で軸を停止します。

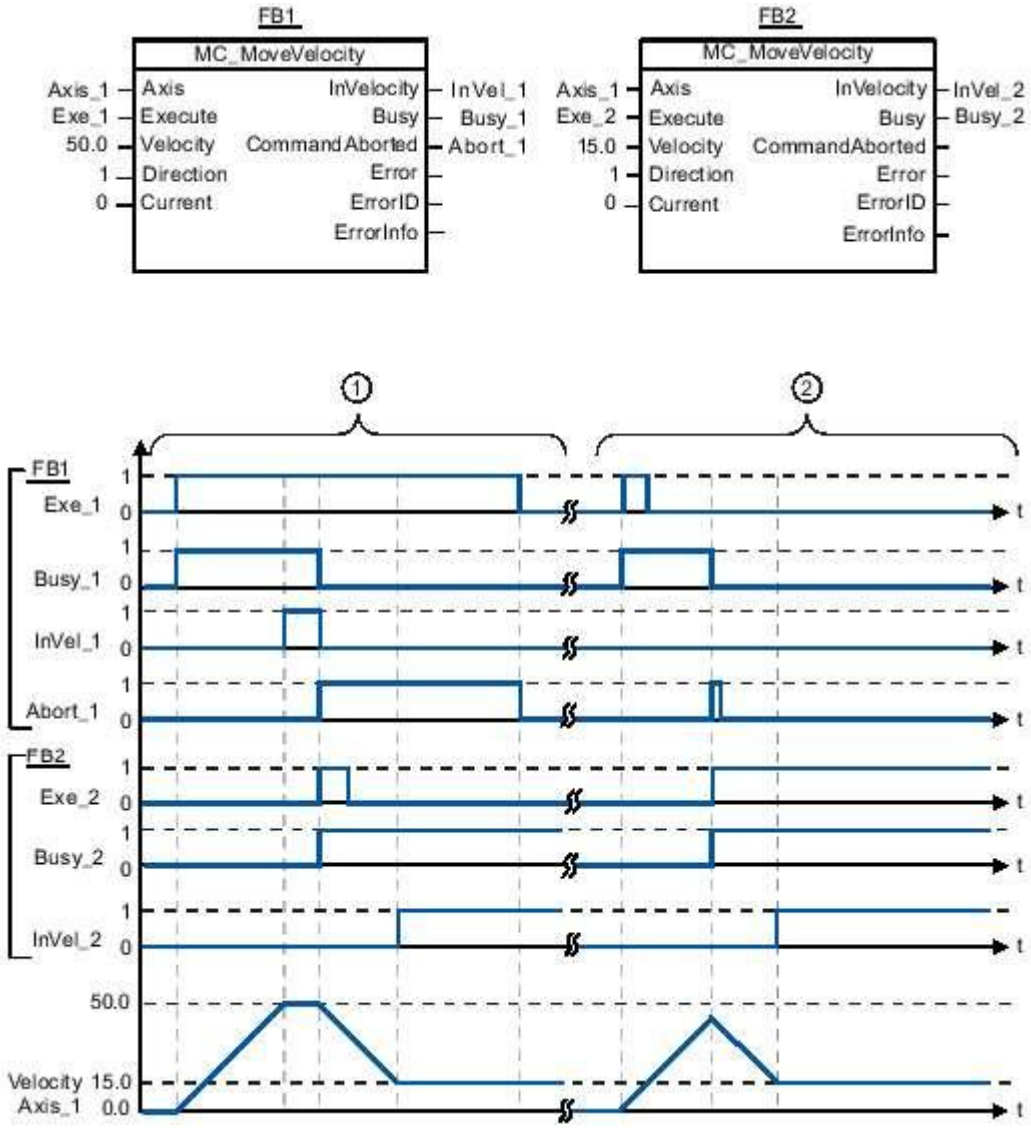
軸が停止状態になると、出力パラメータ「InVelocity」が少なくとも 1 つのプログラムサイクルで TRUE を示します。

"Busy」は減速プロセス中に値 TRUE を示し、「InVelocity」とともに、FALSE に切り替わります。 パラメータ「Execute」 = TRUE を設定すると、「InVelocity」および「Busy」がラッチされます。

「MC_MoveVelocity」コマンドを開始すると、テクノロジーオブジェクトで、ステータスビット「SpeedCommand」がセットされます。 ステータスビット「ConstantVelocity」は、軸停止時にセットされます。 両方のビットは、新しいモーションコマンドが開始されるという新しい状況に合わせて調整されます。

MC_MoveVelocity: V4 以降のファンクションチャート

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速:10.0

①	動作中の MC_MoveVelocity コマンドは、「InVel_1」を介して、コマンドのターゲット速度に達したことを通知します。この後、このコマンドは、別の MC_MoveVelocity コマンドによって中止されます。この中止は、「Abort_1」によって通知されます。新しいターゲット速度 15.0 に達すると、「InVel_2」によって通知されます。この後、軸は、新しい一定の速度で動作し続けます。

②	動作中の MC_MoveVelocity コマンドは、そのターゲット速度に達する前に、別の MC_MoveVelocity コマンドによって中止されます。この中止は、「Abort_1」によって通知されます。新しいターゲット速度 15.0 に達すると、「InVel_2」によって通知されます。この後、軸は、新しい一定の速度で動作し続けます。

MC_MoveJog

この章には下記に関する情報が記載されています：

- MC_MoveJog: V4 以降のジョグモード (S7-1200)
- MC_MoveJog: V4 以降のファンクションチャート (S7-1200)

MC_MoveJog: V4 以降のジョグモード

説明

モーションコントロール命令「MC_MoveJog」は、ジョグモードで、指定された一定の速度で軸を移動します。 このモーションコントロール命令は、テストやコミッショニングなどのために使用します。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- 軸が有効であること。

応答の無効化

MC_MoveJog コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

この新しい MC_MoveJog コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

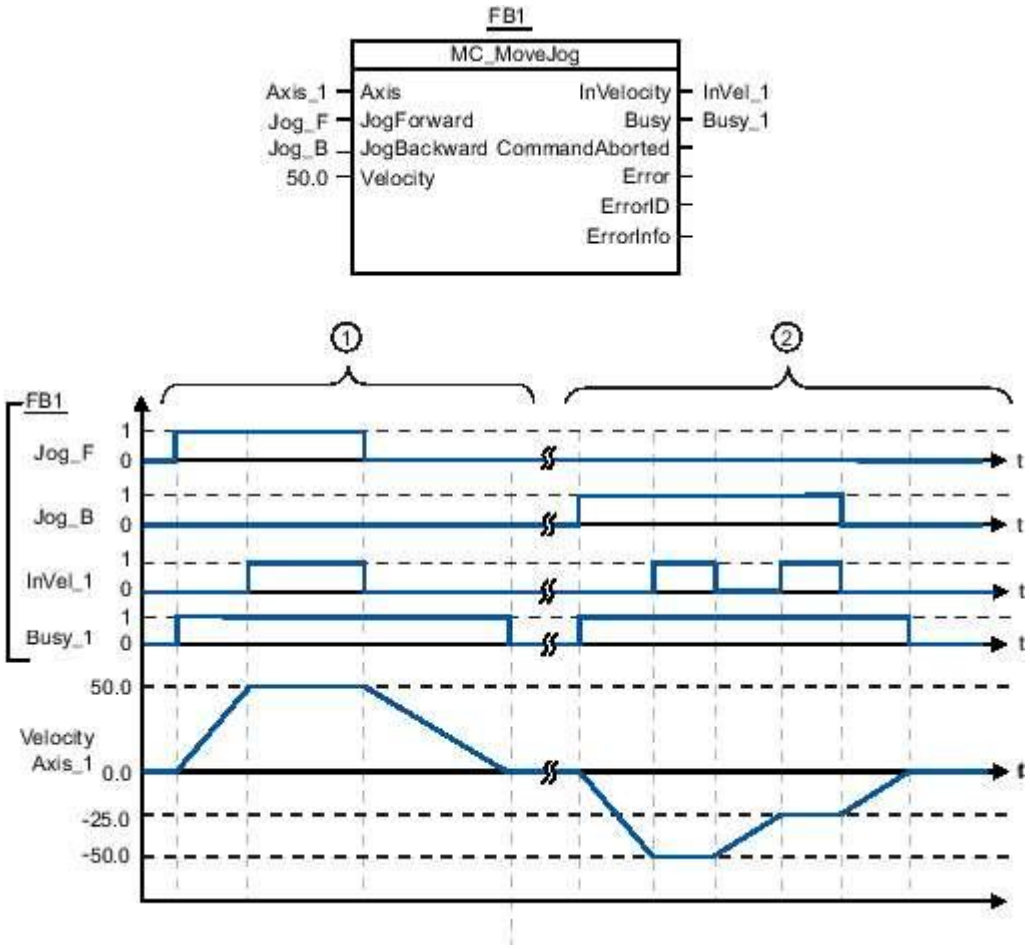
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_Speed-Axis	-	軸テクノロジーオブジェクト

JogForward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸は、パラメータ「Velocity」で指定された速度で、正転方向に移動します。	
JogBackward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸は、パラメータ「Velocity」で指定された速度で、逆転方向に移動します。	
両方のパラメータが同時に TRUE の場合、軸は、設定された減速で停止します。 エラーは、パラメータ「Error」、「ErrorID」、および「ErrorInfo」に示されます。					
Velocity	INPUT	REAL	10.0	あらかじめ設定されたジョグモードの速度 制限値: 開始/停止速度 ≤ 速度 ≤ 最大速度	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	パラメータ「Velocity」で指定された速度に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

MC_MoveJog: V4 以降のファンクションチャート

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 5.0

①	軸は、「Jog_F」によって、ジョグモードで正転方向に移動します。ターゲット速度 50.0 に達すると、「InVel_1」によって通知されます。 軸は、「Jog_F」がリセットされた後、停止状態まで減速します。
②	軸は、「Jog_B」によって、ジョグモードで逆転方向に移動します。ターゲット速度-50.0 に達すると、「InVel_1」によって通知されます。 「Jog_B」が設定されると、パラメータ「Velocity」の値が 25.0 に変更されます。 "InVel_1" がリセットされ、軸を制動します。新しいターゲット速度-25.0 に達すると、「InVel_1」によって通知されます。 軸は、「Jog_B」がリセットされた後、停止状態まで減速します。

MC_CommandTable

この章には下記に関する情報が記載されています：

- MC_CommandTable: V4 以降の軸コマンドをモーションシーケンスとして実行 (S7-1200)

MC_CommandTable: V4 以降の軸コマンドをモーションシーケンスとして実行

説明

モーションコントロール命令「MC_CommandTable」は、複数の個別の軸コントロールコマンドを 1 つの移動シーケンスに結合します。 "MC_CommandTable」は、PTO (Pulse Train Output)経由のドライブ接続で使用できます。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に挿入され、設定されていること。
- ドライブは PTO (Pulse Train Output)を経由して接続されます。
- コマンドテーブルテクノロジーオブジェクトが挿入され、正しく設定済みであること。
- 軸が有効であること。

応答の無効化

MC_CommandTable コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

この新しい MC_CommandTable コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

有効なモーションコントロールコマンドは、最初の[Positioning Relative]、[Positioning Absolute]、[Velocity set point]または[Halt]コマンドの開始によってキャンセルされます。

パラメータ

パラメータ	宣言	データタイプ	既定値	説明	
Axis	INPUT	TO_SpeedAxis	-	軸テクノロジーオブジェクト	
CommandTable	INPUT	TO_CommandTable	-	コマンドテーブルテクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドテーブルの開始	
StartStep	INPUT	INT	1	コマンドテーブルの実行を開始するステップを定義します 制限値: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	コマンドテーブルを実行する最後のステップを定義します。 制限値: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルが正常に実行されました
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルが別のコマンドによってキャンセルされました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	
CurrentStep	OUTPUT	INT	0	コマンドテーブルのステップが現在実行中	
StepCode	OUTPUT	WORD	16#0000	現在実行中のステップのユーザー定義の数値/ビットパターン	

MC_ChangeDynamic

この章には下記に関する情報が記載されています：

- MC_ChangeDynamic: V4 以降の軸の動的設定の変更 (S7-1200)

MC_ChangeDynamic: V4 以降の軸の動的設定の変更

説明

モーションコントロール命令「MC_ChangeDynamic」を使用して、軸の以下の設定を変更できます。

- ランプアップタイム(加速)値の変更
- ランプダウンタイム(減速)値の変更
- 緊急停止ランプダウンタイム(緊急停止減速)値の変更
- スムージングタイム(ジャーク)値の変更

変更の有効性については、タグの説明を参照してください。

必要条件

位置決め軸テクノロジーオブジェクトが適切に設定されていること。

応答の無効化

MC_ChangeDynamic コマンドは、他のモーションコントロールコマンドで中止することはできません。

新しい MC_ChangeDynamic コマンドは、動作中のモーションコントロールコマンドを中止することはしません。

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_SpeedAxis	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	入力パラメータ「RampUpTime」に従ったランプアップタイムの変更
RampUpTime	INPUT	REAL	5.00	軸を停止から設定された最大速度まで、ジャークリミットなしで加速する時間(秒単位)。 この変更は、タグ<軸名>. Config.DynamicDefaults.Acceleration 変更の有効性については、このタグの説明を参照してください。	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	ランプダウンタイムが入力パラメータ「RampDownTime」に一致するよう変更
RampDownTime	INPUT	REAL	5.00	軸を設定された最大速度から停止まで、ジャークリミッタなしで減速する時間(秒単位)。 この変更は、タグ<軸名>. Config.DynamicDefaults.Deceleration 変更の有効性については、このタグの説明を参照してください。	

Change-Emergen-cy	INPUT	BOOL	FALSE	TRUE	入力パラメータ「EmergencyRampTime」に従った緊急停止ランプダウンタイムの変更
Emer-gencyR-ampTime	INPUT	REAL	2.00	軸を設定された最大速度から停止まで、緊急停止モードで、ジャークリミッタなしで減速する時間(秒)。 この変更は、タグ<軸名>. Config.DynamicDe-faults.EmergencyDeceleration 変更の有効性については、このタグの説明を参照してください。	
Change-JerkTime	INPUT	BOOL	FALSE	TRUE	入力パラメータ「JerkTime」に従ったスムージングタイムの変更
JerkTime	INPUT	REAL	0.25	軸の加速および減速ランプで使用されるスムージングタイム(秒単位) この変更は、タグ<軸名>. Config.DynamicDe-faults.Jerk 変更の有効性については、このタグの説明を参照してください。	
Done	OUTPUT	BOOL	FALSE	TRUE	変更された値は、テクノロジーデータブロックに書き込み済みです。タグの説明には、変更が有効になる時点が表示されます。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。エラーの原因は、パラメータ「Error-ID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

注記

入力パラメータ「RampUpTime」、「RampDownTime」、「EmergencyRampTime」、および「Jerk-Time」では、結果パラメータ:「加速」、「減速」、「緊急停止」、および「ジャーク」の許容制限値を超える値を入力できます。

「動的」セクションの式および制限値を考慮して、入力が確実に有効範囲内になるように注意してください。

MC_ReadParam

この章には下記に関する情報が記載されています：

- MC_ReadParam: V4 以降の位置決め軸のモーションデータの連続的な読み取り (S7-1200)

MC_ReadParam: V4 以降の位置決め軸のモーションデータの連続的な読み取り

説明

モーションコントロール命令[MC_ReadParam]によって、軸のモーションデータおよびステータスメッセージの連続読み出しが可能になります。 対応するタグの現在値は、コマンドの開始時に決定されます。

以下のモーションデータおよびステータスメッセージを読み出すことができます。

- テクノロジーのバージョン V4 以降:
  - 軸のセットポイント位置
  - セットポイントおよび軸の現在速度
  - 軸のターゲット位置からの現在の距離
  - 軸のターゲット位置
- テクノロジーバージョン V5 以降(追加):
  - 軸の現在の位置
  - 軸の現在速度
  - 現在の追従誤差
  - ドライブステータス
  - エンコーダステータス
  - ステータスビット
  - エラービット

必要条件

位置決め軸テクノロジーオブジェクトが適切に設定されていること。

応答の無効化

MC_ReadParam コマンドは、他のモーションコントロールコマンドで中止することはできません。

新しい MC_ReadParam コマンドは、動作中のモーションコントロールコマンドを中止することはしません。

パラメータ

パラメータ	宣言	データタイプ	既定値	説明	
Enable	INPUT	BOOL	FALSE	TRUE	「Parameter」で指定されたタグを読み取り、値を「Value」で指定された宛先アドレスに格納します。
				FALSE	割り当て済みのモーションデータを更新しません。
Parameter	INPUT	VARIANT (REAL)	-	読み取る値への VARIANT ポインタ。 次のタグが許可されています。 • <軸名>.Position • <軸名>.Velocity • <軸名>.ActualPosition	

				• <軸名>.ActualVelocity • <軸名>.StatusPositioning.<タグ名> • <軸名>.StatusDrive.<タグ名> • <軸名>.StatusSensor.<タグ名> • <軸名>.StatusBits.<タグ名> • <軸名>.ErrorBits.<タグ名> タグ名およびタグ構造の説明については、付録「位置決め軸テクノロジーオブジェクト V4 以降のタグ」を参照してください。	
Value	INOUT	VARIANT (REAL)	-	ターゲットタグへの VARIANT ポインタまたは読み取られた値が書き込まれる宛先アドレス。	
Valid	OUTPUT	BOOL	FALSE	TRUE	読み取った値が有効です。
				FALSE	読み取った値が無効です。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

MC_WriteParam

この章には下記に関する情報が記載されています：

- MC_WriteParam: V4 以降の位置決め軸のタグの書き込み (S7-1200)

MC_WriteParam: V4 以降の位置決め軸のタグの書き込み 

説明

モーションコントロール命令「MC_WriteParam」は、ユーザープログラムへの位置決め軸テクノロジーオブジェクトのタグの書き込みを可能にします。 ユーザープログラムでのタグへの値の割り当てとは異なり、「MC_WriteParam」は読み取り専用タグの値も変更できます。

タグ、タグを書き込むことができる条件、それが有効になるタイミングについては、テクノロジーオブジェクトタグの説明を参照してください。

PROFIdrive/アナログ出力経由のドライブ接続の場合、パラメータによっては、[MC_WriteParam]による書き込みの後に、テクノロジーオブジェクトの再起動が必要な場合があります。 再起動が必要な場合、これはテクノロジーオブジェクト<軸名>.StatusBits.RestartRequired のタグで示されます。 パラメータ値の変更は、テクノロジーオブジェクトの有効化(MC_Power.Status = TRUE)による再起動後にこれらのパラメータで有効になります。

必要条件

- 位置決め軸テクノロジーオブジェクトが適切に設定されていること。
- ユーザープログラムで読み取り専用のタグを書き込むために、軸が無効になっていること。

応答の無効化

MC_WriteParam コマンドは、他のモーションコントロールコマンドで中止することはできません。

新しい MC_WriteParam コマンドは、動作中のモーションコントロールコマンドを中止することはしません。

パラメータ

パラメータ	宣言	データタイプ	既定値	説明	
Parameter	INPUT	VARIANT (BOOL、INT、DINT、REAL)	-	書き込むテクノロジーオブジェクトタグの位置決め軸への VARIANT ポインタ(宛先アドレス)	
Value	INPUT	VARIANT (BOOL、INT、DINT、REAL)	-	書き込む値への VARIANT ポインタ(ソースアドレス)	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
Done	OUTPUT	BOOL	FALSE	TRUE	値が書き込まれました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID

S7-1200 モーションコントロール V1...3

この章には下記に関する情報が記載されています：

- MC_Power (S7-1200)
- MC_Reset (S7-1200)
- MC_Home (S7-1200)
- MC_Halt (S7-1200)
- MC_MoveAbsolute (S7-1200)
- MC_MoveRelative (S7-1200)
- MC_MoveVelocity (S7-1200)
- MC_MoveJog (S7-1200)
- MC_CommandTable (S7-1200)
- MC_ChangeDynamic (S7-1200)

MC_Power

この章には下記に関する情報が記載されています：

- MC_Power: 軸の有効化、無効化 V1...3 (S7-1200)
- MC_Power: ファンクションチャート V1...3 (S7-1200)

MC_Power: 軸の有効化、無効化 V1...3

説明

モーションコントロール命令「MC_Power」は、軸を有効/無効にします。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 未解決のイネーブル禁止エラーがないこと。

応答の無効化

「MC_Power」の実行は、モーションコントロールコマンドによって中止できません。

軸の無効化(入力パラメータ「Enable」 = FALSE)は、選択した「StopMode」に従って、関連するテクノロジーオブジェクトに対するすべてのモーションコントロールコマンドを中止します。

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト	
Enable	INPUT	BOOL	FALSE	TRUE	モーションコントロールが、軸を有効にしようとしています。
				FALSE	設定された「StopMode」に従ってすべての現在のジョブが中止されます。 軸は停止し、無効になっています。
Stop-Mode	INPUT	INT	0	0	緊急停止 軸を無効にする要求が保留中の場合、設定された緊急停止減速で、軸が制動します。 軸は停止後に無効になります。
				1	直ちに停止 軸を無効にする要求が保留中の場合、減速なしで、軸が無効になります。 パルス出力が直ちに停止します。
				2	ジャーク制御による緊急停止 軸を無効にする要求が保留中の場合、設定された緊急停止減速で、軸が制動します。 ジャーク制御が有効な場合、設定されたジャークが考慮されます。 軸は停止後に無効になります。
Status	OUTPUT	BOOL	FALSE	軸イネーブルのステータス	
				FALSE	軸が無効になります。 軸はモーションコントロールコマンドを実行せず、すべての新しいコマンドの承

					認も行いません(例外: MC_Reset コマンド)。 軸が原点復帰していません。 無効時、軸が静止するまでは、ステータスは FALSE に変わりません。
				TRUE	軸が有効になります。 軸は、モーションコントロールコマンドを実行できる状態です。 軸が有効なときは、信号「ドライブ準備完了」が保留中になるまで、ステータスは TRUE に変わりません。 軸の構成で「ドライブ準備完了」ドライブインターフェースが設定されなかった場合、ステータスは直ちに TRUE に切り替わります。
Busy	OUTPUT	BOOL	FALSE	TRUE	"MC_Power" が動作中。
Error	OUTPUT	BOOL	FALSE	TRUE	モーションコントロール命令「MC_Power」、または関連するテクノロジーオブジェクトでエラーが発生しました。エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の[エラー ID]	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の[エラー情報 ID]	

注記
軸がエラーのためにスイッチがオフになった場合、エラーを除去し、確認した後は、軸が自動的に有効になります。 この場合、このプロセス時に、入力パラメータ「Enable」が値 TRUE を保持していることが必要です。

ドライブインターフェースが設定された軸の有効化

この軸を有効にするには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 入力パラメータ「StopMode」を必要な値で初期化します。 入力パラメータ「Enable」を TRUE に設定します。

   「ドライブ有効化済み」のためのイネーブル出力が TRUE に変わり、ドライブへの電源を有効にします。 CPU は、ドライブの「ドライブ準備完了」信号を待機します。

   CPU の設定された準備完了入力で「ドライブ準備完了」信号が使用可能になると、軸が有効になります。 出力パラメータ「Status」と、テクノロジーオブジェクトのタグ<軸名>.StatusBits.Enable が、値 TRUE を示します。

ドライブインターフェースが設定されていない軸の有効化

この軸を有効にするには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 入力パラメータ「StopMode」を必要な値で初期化します。 入力パラメータ「Enable」を TRUE に設定します。軸が有効になります。 出力パラメータ「Status」と、テクノロジーオブジェクトのタグ<軸名>.StatusBits.Enable が、値 TRUE を示します。

軸の無効化

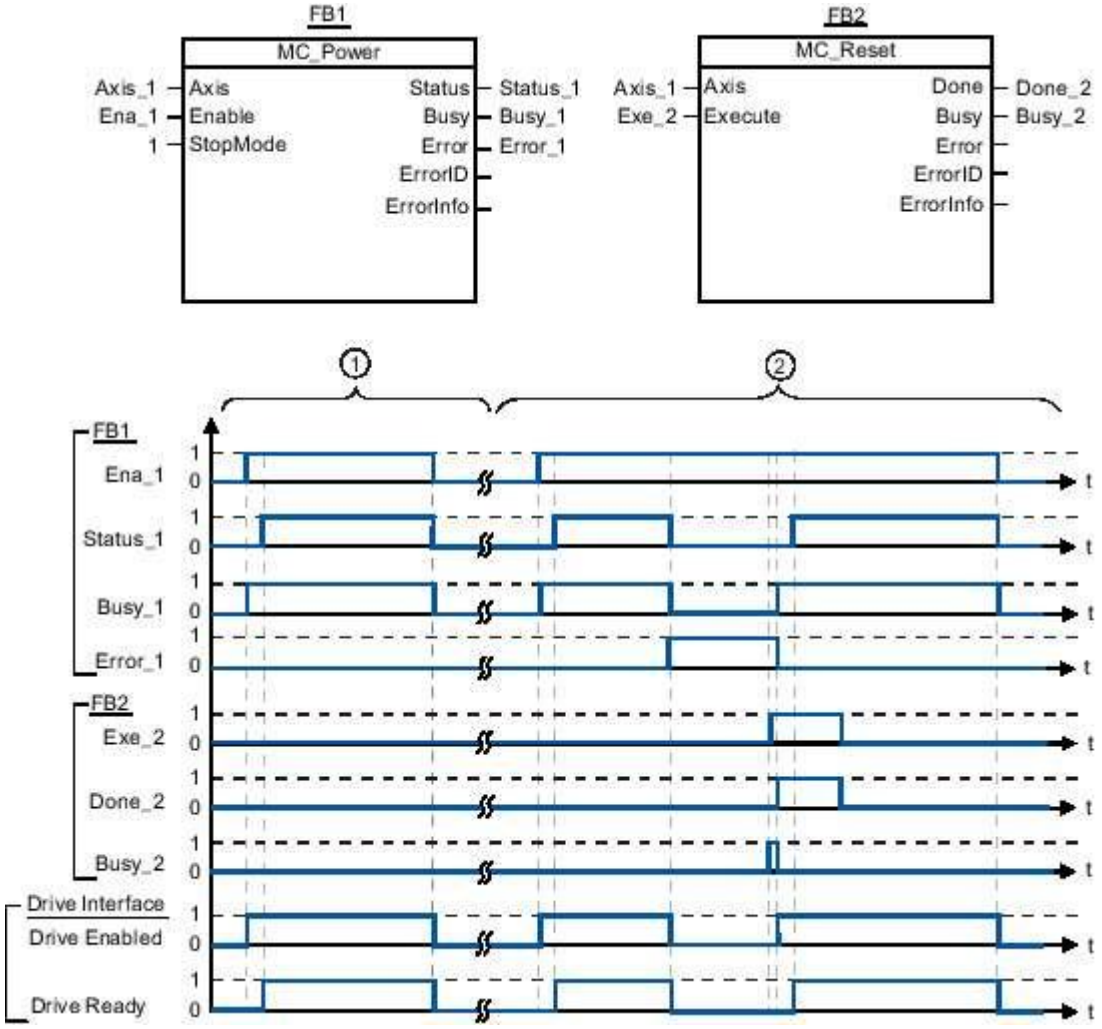
軸を無効にするには、以下の手順を実行します。

1. 軸を停止します。

   テクノロジーオブジェクトのタグ<軸名>.StatusBits.StandStill で、軸をいつ停止させるかを指定できます。
2. 停止した後、入力パラメータ「Enable」を FALSE に設定します。
3. 出力パラメータ「Busy」および「Status」と、テクノロジーオブジェクトのタグ<軸名>.StatusBits.Enable が値 FALSE を示すと、軸の無効化が完了です。

MC_Power: ファンクションチャート V1...3

ファンクションチャート



①	軸が有効になった後、再び無効になります。 ドライブが「ドライブ準備完了」信号を CPU に戻すと、成功したイネーブルを「Status_1」経由で読み出すことができます。
②	軸イネーブルに続いて、軸の無効化の原因となる エラーが発生しました。 エラーが除去され、「MC_Reset」で確認されます。 この場合、軸は再び有効になります。

MC_Reset

この章には下記に関する情報が記載されています：

- MC_Reset: エラーの確認 V1...3 (S7-1200)

MC_Reset: エラーの確認 V1...3

説明

モーションコントロール命令「MC_Reset」を使用して、「軸停止ありの動作エラー」および「コンフィグレーションエラー」を確認することができます。 確認が必要なエラーについては、「ErrorID および ErrorInfo の一覧」の「対策」を参照してください。

バージョン V3.0 から、RUN 動作モードで、軸設定をワークメモリにダウンロードできます。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 確認が必要な保留中のコンフィグレーションエラーの原因が除去済みであること(たとえば、位置決め軸テクノロジーオブジェクトの加速が有効な値に変更済みであること)。

応答の無効化

MC_Reset コマンドは、他のモーションコントロールコマンドで中止することはできません。

新しい MC_Reset コマンドは、他のすべての動作中のモーションコントロールコマンドを中止しません。

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
Restart	INPUT	BOOL	FALSE	(バージョン V3.0 から)	
				TRUE	軸設定をロードメモリからワークメモリにダウンロードします。 このコマンドは、軸が無効になっているときのみ実行できます。 CPU へのダウンロードに関する注記を参照してください。
				FALSE	保留中のエラーを確認します
Done	OUTPUT	BOOL	FALSE	TRUE	エラーが確認応答済みです。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**

MC_Reset による確認を必要とするエラーの確認

エラーを確認するには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 入力パラメータ「Execute」での立ち上がりエッジで、エラーの確認を開始します。
3. 出力パラメータ「Done」が値 TRUE を示し、テクノロジーオブジェクトのタグ<軸名>.StatusBits.Error が値 FALSE を示す場合、エラーが確認応答済みです。

MC_Home

この章には下記に関する情報が記載されています：

- MC_Home: 原点軸、基準点の設定 V1...3 (S7-1200)

MC_Home: 原点軸、基準点の設定 V1...3

説明

モーションコントロール命令「MC_Home」を使用して、軸座標を物理的な実際のドライブ位置に適合させます。 軸の絶対位置決めには、原点復帰が必要です。 次のタイプの原点復帰を実行できます。

- アクティブ原点復帰(Mode = 3)

  原点復帰手順が自動的に実行されます。

- パッシブ原点復帰(Mode = 2)

  パッシブ原点復帰時には、モーションコントロール命令「MC_Home」は、原点復帰モーションを実行しません。 これに必要な移動モーションは、ユーザーが他のモーションコントロール命令を使用して実行する必要があります。 原点復帰スイッチが検出されると、軸が原点復帰します。

- 絶対直接原点復帰(Mode = 0)

  現在の軸位置が、パラメータ「Position」の値に設定されます。

- 相対直接原点復帰(Mode = 1)

  現在の軸位置が、パラメータ「Position」の値だけオフセットされます。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 軸が有効であること。
- Mode = 0、1、または 2 での開始時に、MC_CommandTable コマンドが動作中でないこと。

応答の無効化

応答の無効化は、選択したモードに応じて異なります。

Mode = 0, 1

MC_Home コマンドは、他のモーションコントロールコマンドで中止することはできません。

MC_Home コマンドは、他のすべての動作中のモーションコントロールコマンドを中止しません。 位置関連のモーションコマンドは、新しい原点復帰位置(入力パラメータ: 「Position」での値)に従って原点復帰した後、再開されます。 "Position").

Mode = 2

MC_Home コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 2, 3

新しい MC_Home コマンドは、次の有効なモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 2

位置関連のモーションコマンドは、新しい原点復帰位置(入力パラメータ: 「Position」での値)に従って原点復帰した後、再開されます。 "Position").

Mode = 3

MC_Home コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3

- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_Home コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 2, 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
Position	INPUT	REAL	0.0	• Mode = 0、2、および 3 原点復帰動作を完了した後の絶対位置 • Mode = 1 現在の軸位置用補正値 制限値: $-1.0e^{12} \leq Position \leq 1.0e^{12}$	
Mode	INPUT	INT	0	原点復帰モード	
				0	絶対直接原点復帰 新しい軸位置は、パラメータ「Position」の値です。
				1	相対直接原点復帰 新しい軸位置は、現在の軸位置 + パラメータ「Position」の値です。
				2	パッシブ原点復帰 軸の構成に従った原点復帰。 原点復帰した後、パラメータ「Position」の値が新しい軸位置として設定されます。
				3	アクティブ原点復帰

					軸の設定に従った原点復帰手順です。 原点復帰した後、パラメータ「Position」の値が新しい軸位置として設定されます。
Done	OUTPUT	BOOL	FALSE	TRUE	コマンドが完了しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

注記

以下の条件の場合、軸の原点復帰は失われます。

- 「MC_Power」モーションコントロール命令による軸の無効化
- 自動モードと手動コントロールの切り替え
- アクティブ原点復帰の開始時。 原点復帰動作が正常終了した後、軸原点復帰が再び使用できます。
- CPU の電源オフ -> 電源オンの後
- CPU 再起動(RUN-STOP -> STOP-RUN)後

軸の原点復帰

軸を原点復帰するには、以下の手順を実行します。

1. 上記の必要条件をチェックします。
2. 必要な入力パラメータを値で初期化し、入力パラメータ「Execute」での立ち上がりエッジで原点復帰動作を開始します。
3. 出力パラメータ「Done」とテクノロジーオブジェクトタグ<軸名>.StatusBits.HomingDone が値 TRUE を示す場合、原点復帰が完了しています。

MC_Halt

この章には下記に関する情報が記載されています：

- MC_Halt: 軸の一時停止 V1...3 (S7-1200)
- MC_Halt: ファンクションチャート V1...3 (S7-1200)

MC_Halt: 軸の一時停止 V1...3

説明

「MC_Halt」モーションコントロール命令は、すべての動きを停止し、設定された減速で軸を停止状態にします。 停止位置は定義されていません。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 軸が有効であること。

応答の無効化

MC_Halt コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_Halt コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

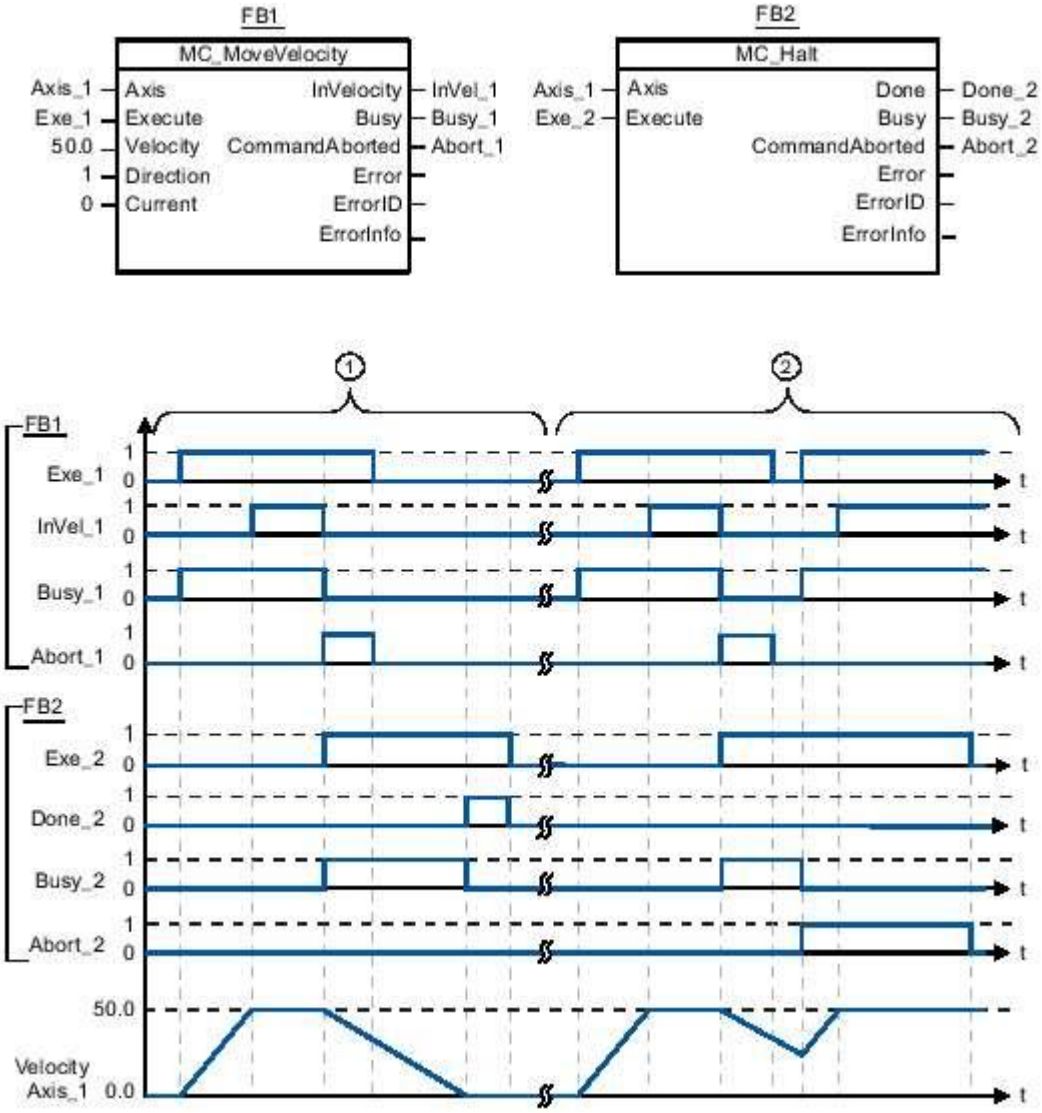
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
Done	OUTPUT	BOOL	FALSE	TRUE	速度 0 に達しました。

Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

MC_Halt: ファンクションチャート V1...3

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 5.0

①	軸は、停止状態になるまで、MC_Halt コマンドによって制動します。 軸の停止は、「Done_2」によって通知されます。
②	MC_Halt コマンドが軸を制動している途中でも、このコマンドは別のモーションコマンドによって中止されます。 この中止は、「Abort_2」によって通知されます。

MC_MoveAbsolute

この章には下記に関する情報が記載されています：

- MC_MoveAbsolute: 軸の絶対位置決め V1...3 (S7-1200)
- MC_MoveAbsolute: ファンクションチャート V1...3 (S7-1200)

MC_MoveAbsolute: 軸の絶対位置決め V1...3

説明

「MC_MoveAbsolute」モーションコントロール命令は、軸位置決めモーションを開始し、軸を絶対位置まで移動します。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 軸が有効であること。
- 軸が原点復帰していること。

応答の無効化

MC_MoveAbsolute コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_MoveAbsolute コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

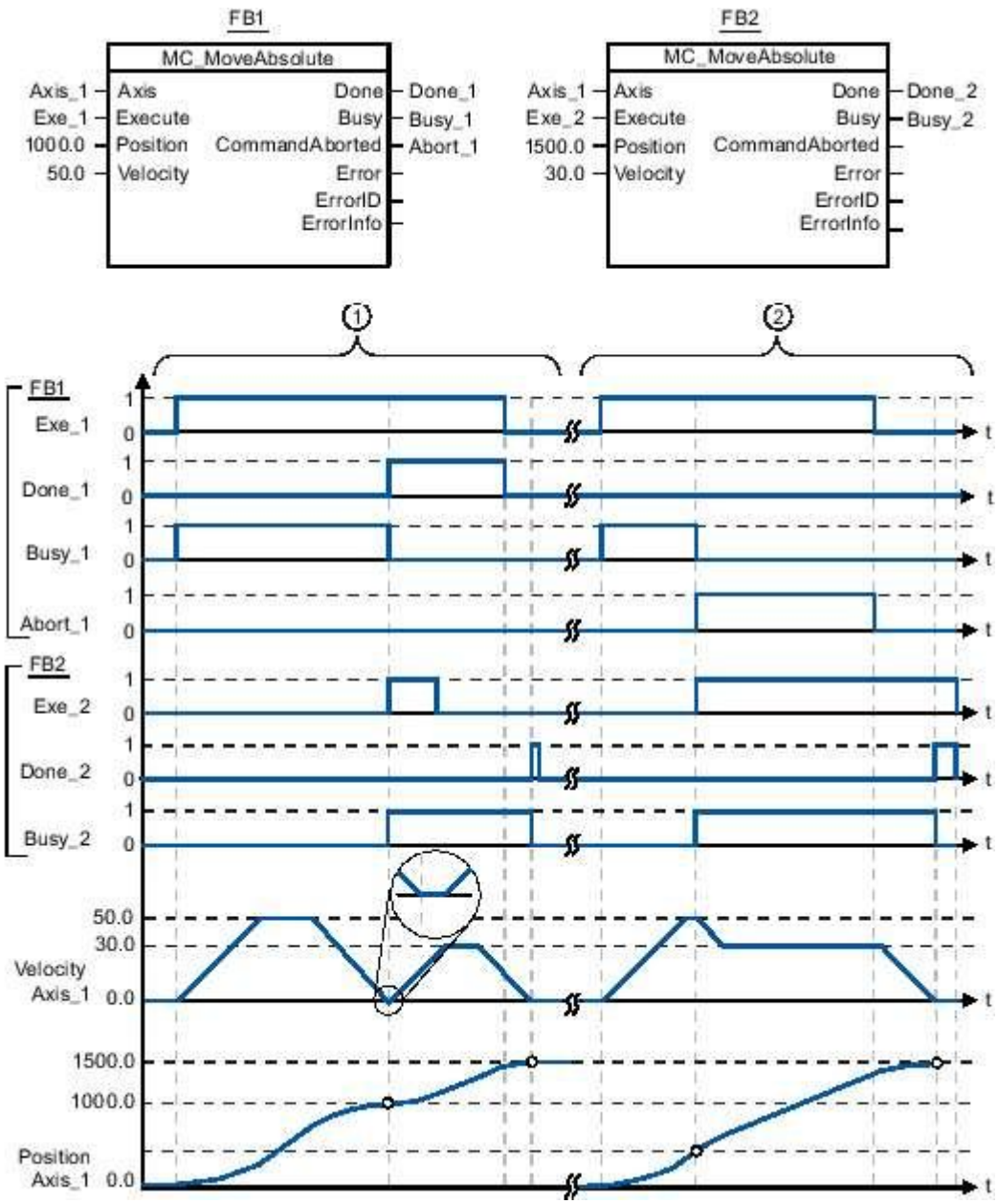
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始
Position	INPUT	REAL	0.0	絶対ターゲット位置

				制限値: -1.0e^{12} ≤ Position ≤ 1.0e^{12}	
Velocity	INPUT	REAL	10.0	軸の速度 設定された加速および減速、およびアプローチするターゲット位置のために、常にこの速度に達するとは限りません。 制限値: 開始/停止速度 ≤ Velocity ≤ 最大速度	
Done	OUTPUT	BOOL	FALSE	TRUE	絶対ターゲット位置に到達しました
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

MC_MoveAbsolute: ファンクションチャート V1...3

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 10.0

①	軸は、MC_MoveAbsolute コマンドによって絶対位置 1000.0 まで移動します。 軸がターゲット位置に到達すると、「Done_1」によって通知されます。 「Done_1」 = TRUE の場合、ターゲット位置が 1500.0 の別の MC_MoveAbsolute コマンドが開始されます。応答時間(たとえば、ユーザープログラムのサイクルタイムなど)のために、軸はすぐに停止状態になります(拡大された詳細図を参照)。 軸が新しいターゲット位置に到達すると、「Done_2」によって通知されます。
②	動作中の MC_MoveAbsolute コマンドは、別の MC_MoveAbsolute コマンドによって中止されます。 この中止は、「Abort_1」によって通知されます。 この後、この軸は、新しい速度で新しいターゲット位置 1500.0 まで移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveRelative

この章には下記に関する情報が記載されています：

- MC_MoveRelative: 軸の相対位置決め V1...3 (S7-1200)
- MC_MoveRelative: ファンクションチャート V1...3 (S7-1200)

MC_MoveRelative: 軸の相対位置決め V1...3

説明

「MC_MoveRelative」モーションコントロール命令は、開始位置を基準にした位置決めモーションを開始します。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 軸が有効であること。

応答の無効化

MC_MoveRelative コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

新しい MC_MoveRelative コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

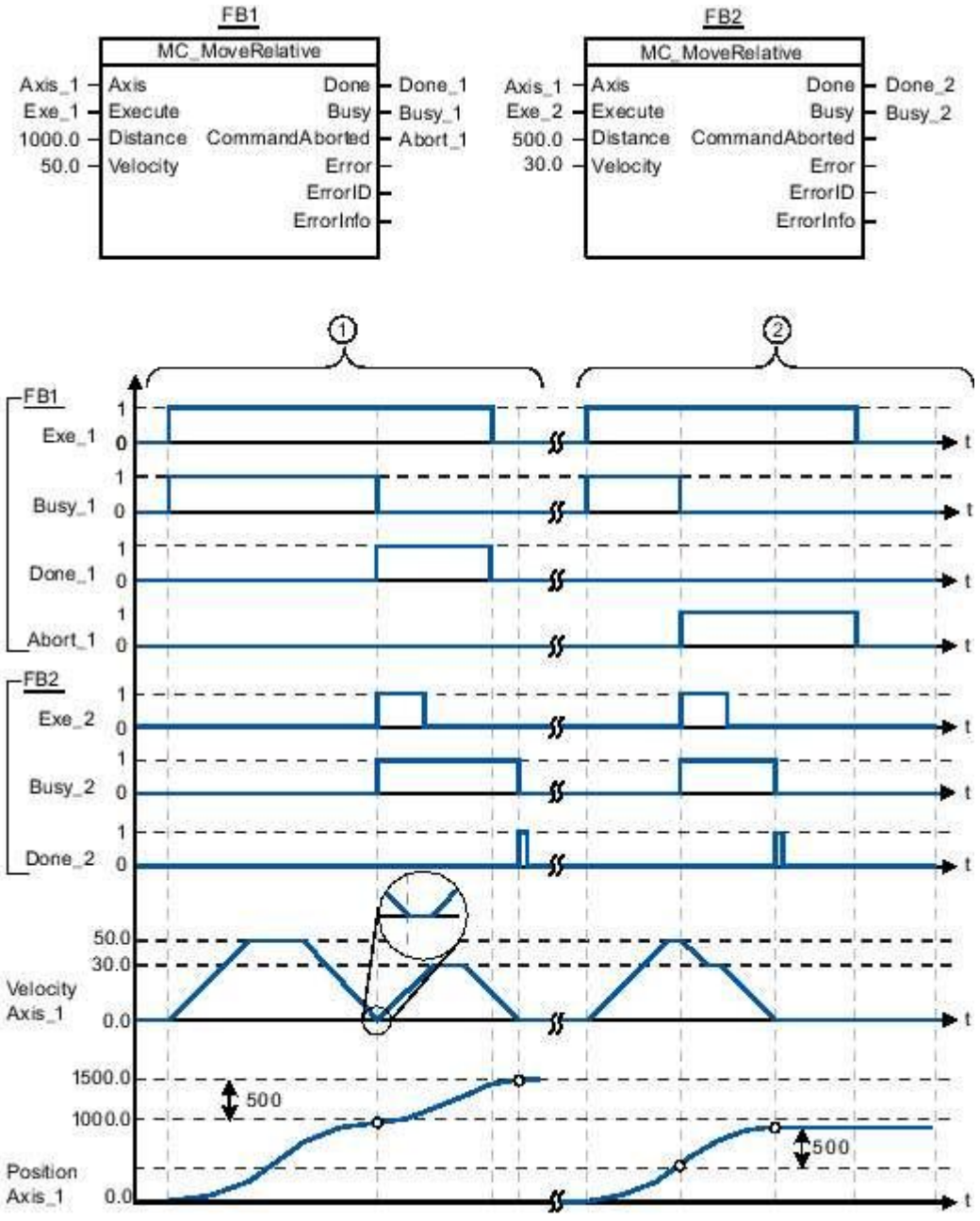
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始
Distance	INPUT	REAL	0.0	位置決め動作での移動距離 制限値:

				$-1.0e^{12} \leq$ Distance $\leq 1.0e^{12}$	
Velocity	INPUT	REAL	10.0	軸の速度 設定された加速および減速、移動する距離のために、常にこの速度に達するとは限りません。 制限値: 開始/停止速度 ≤ Velocity ≤ 最大速度	
Done	OUTPUT	BOOL	FALSE	TRUE	ターゲット位置に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorIn-fo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

MC_MoveRelative: ファンクションチャート V1...3

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速: 10.0

①	軸は、MC_MoveRelative コマンドによって、距離(「Distance」)1000.0 だけ移動します。 軸がターゲット位置に到達すると、「Done_1」によって通知されます。 「Done_1」 = TRUE の場合、移動距離が 500.0 の別の MC_MoveRelative コマンドが開始されます。応答時間(たとえば、ユーザープログラムのサイクルタイムなど)のために、軸はすぐに停止状態になります(拡大された詳細図を参照)。 軸が新しいターゲット位置に到達すると、「Done_2」によって通知されます。
②	動作中の MC_MoveRelative コマンドは、別の MC_MoveRelative コマンドによって中止されます。 この中止は、「Abort_1」によって通知されます。 この後、軸は、新しい速度で新しい距離(「Distance」) 500.0 だけ移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveVelocity

この章には下記に関する情報が記載されています：

- MC_MoveVelocity: 軸をあらかじめ設定された回転速度で移動 V1...3 (S7-1200)
- MC_MoveVelocity: ファンクションチャート V1...3 (S7-1200)

MC_MoveVelocity: 軸をあらかじめ設定された回転速度で移動 V1...3

説明

モーションコントロール命令「MC_MoveVelocity」は、軸を常に指定された速度で移動します。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 軸が有効であること。

応答の無効化

MC_MoveVelocity は、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

この新しい MC_MoveVelocity コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始
Velocity	INPUT	REAL	10.0	軸モーションの速度指定 制限値:

				開始/停止速度 ≤ |Velocity ≤ 最大速度 (Velocity = 0.0 は許可されます)	
Direction	INPUT	INT	0	方向指定	
				0	回転方向は、パラメータ「Velocity」の値の符号に対応します。
				1	正転の回転方向 (パラメータ「Velocity」の値の符号は無視されます)
				2	逆転の回転方向 (パラメータ「Velocity」の値の符号は無視されます)
Current	INPUT	BOOL	FALSE	現在速度の維持	
				FALSE	「現在速度の維持」が無効になります。 パラメータ「Velocity」および「Direction」が使用されます。
				TRUE	「現在速度の維持」が有効になります。 パラメータ「Velocity」および「Direction」は考慮されません。 軸が現在の速度でモーションを再開すると、「InVelocity」パラメータは値 TRUE *を返します。*
InVelocity	OUTPUT	BOOL	FALSE	TRUE	• "Current" = FALSE: パラメータ「Velocity」で指定された速度に達しました。 • "Current" = TRUE: 軸は、開始時の現在速度で移動します。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の**エラー ID**	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」の**エラー情報 ID**	

ゼロセットポイント速度の場合の動作(Velocity = 0.0)

「Velocity」 = 0.0 の MC_MoveVelocity コマンド(MC_Halt コマンドと同様に)は、動作中のモーションコマンドを中止、設定された減速で軸を停止します。

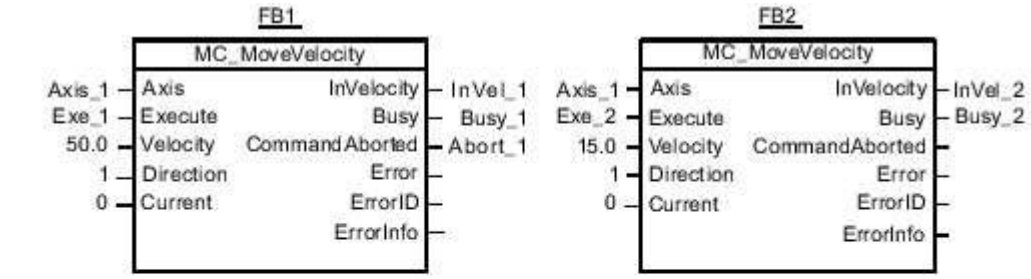
軸が停止状態になると、出力パラメータ「InVelocity」が少なくとも 1 つのプログラムサイクルで TRUE を示します。

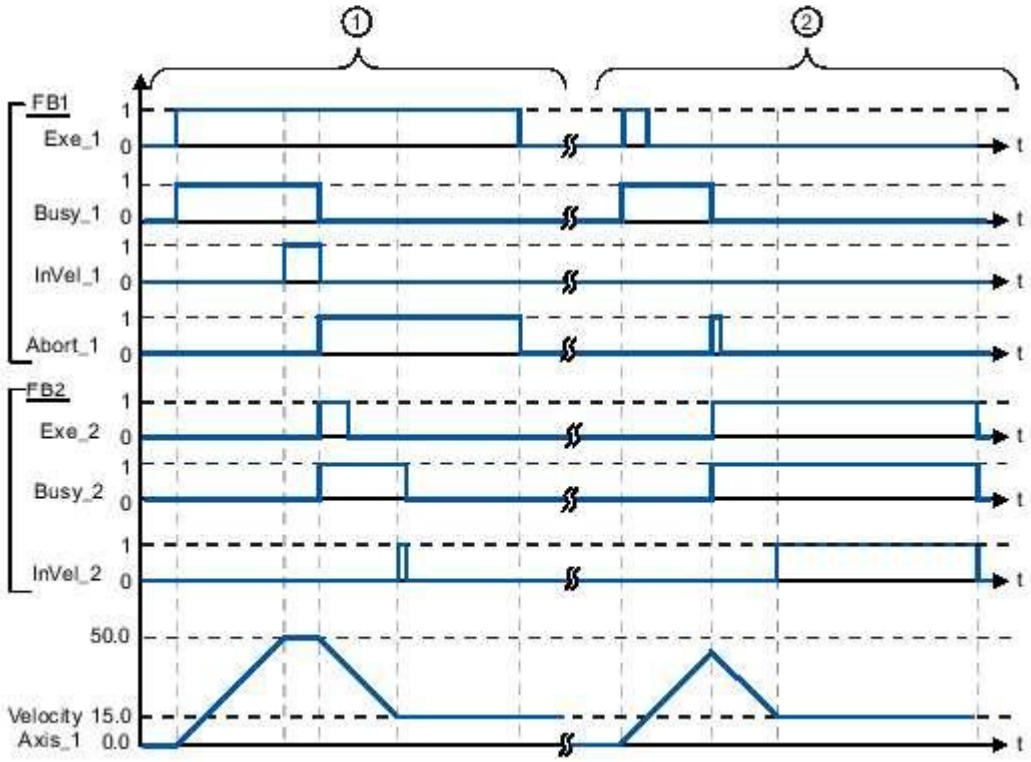
"Busy」は減速プロセス中に値 TRUE を示し、「InVelocity」とともに、FALSE に切り替わります。 パラメータ「Execute」 = TRUE を設定すると、「InVelocity」および「Busy」がラッチされます。

「MC_MoveVelocity」コマンドを開始すると、テクノロジーオブジェクトで、ステータスビット「SpeedCommand」がセットされます。 ステータスビット「ConstantVelocity」は、軸停止時にセットされます。 両方のビットは、新しいモーションコマンドが開始されるという新しい状況に合わせて調整されます。

MC_MoveVelocity: ファンクションチャート V1...3

ファンクションチャート





設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速:10.0
- 減速:10.0

①	動作中の MC_MoveVelocity コマンドは、「InVel_1」を介して、コマンドのターゲット速度に達したことを通知します。この後、このコマンドは、別の MC_MoveVelocity コマンドによって中止されます。この中止は、「Abort_1」によって通知されます。新しいターゲット速度 15.0 に達すると、「InVel_2」によって通知されます。この後、軸は、新しい一定の速度で動作し続けます。

②	動作中の MC_MoveVelocity コマンドは、そのターゲット速度に達する前に、別の MC_MoveVelocity コマンドによって中止されます。この中止は、「Abort_1」によって通知されます。新しいターゲット速度 15.0 に達すると、「InVel_2」によって通知されます。この後、軸は、新しい一定の速度で動作し続けます。

MC_MoveJog

この章には下記に関する情報が記載されています：

- MC_MoveJog: ジョグモードでの軸の移動 V1...3 (S7-1200)
- MC_MoveJog: ファンクションチャート V1...3 (S7-1200)

MC_MoveJog: ジョグモードでの軸の移動 V1...3

説明

モーションコントロール命令「MC_MoveJog」は、ジョグモードで、指定された一定の速度で軸を移動します。 このモーションコントロール命令は、テストやコミッショニングなどのために使用します。

必要条件

- 軸テクノロジーオブジェクトが正しく設定済みであること。
- 軸が有効であること。

応答の無効化

MC_MoveJog コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

この新しい MC_MoveJog コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

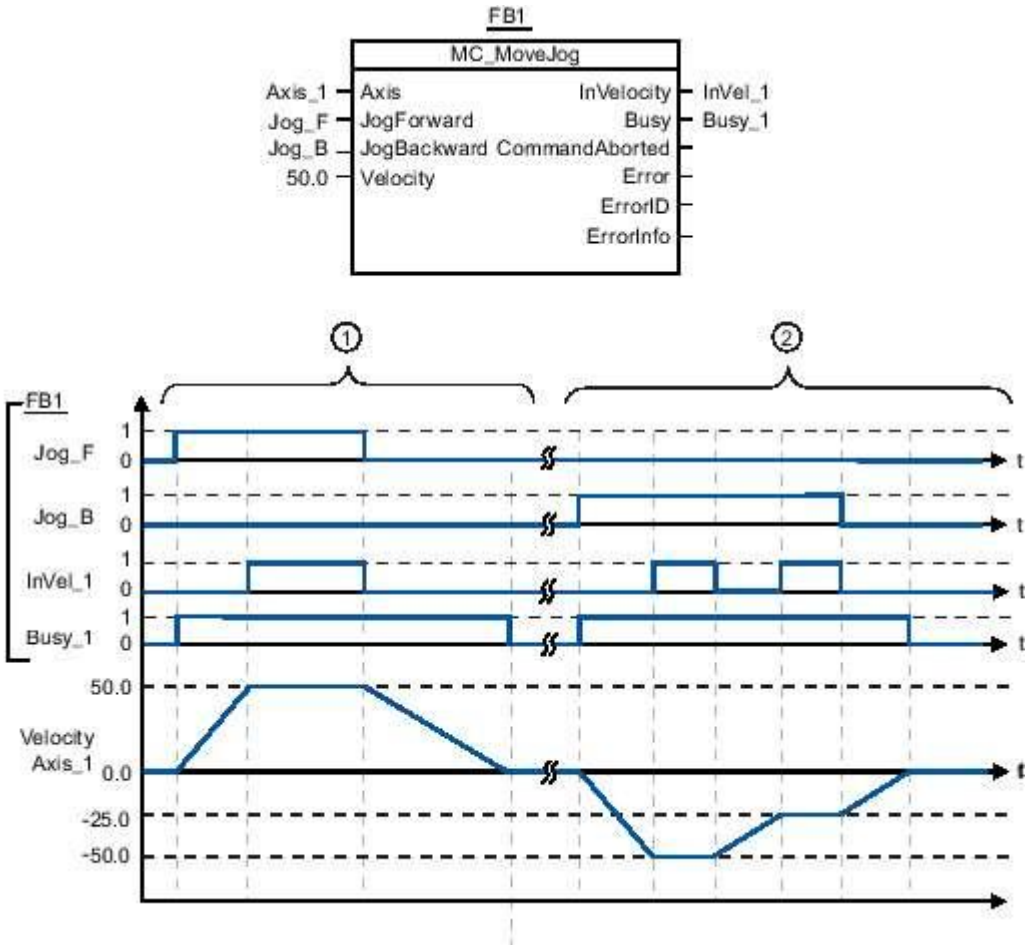
パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト
JogForward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸は、パラメータ「Velocity」で指定された速度で、正転方向に移動します。

JogBackward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸は、パラメータ「Velocity」で指定された速度で、逆転方向に移動します。	
両方のパラメータが同時に TRUE の場合、軸は、設定された減速で停止します。 エラーは、パラメータ「Error」、「ErrorID」、および「ErrorInfo」に示されます。					
Velocity	INPUT	REAL	10.0	あらかじめ設定されたジョグモードの速度 制限値、命令バージョン V1.0: 開始/停止速度 ≤ |Velocity ≤ 最大速度 制限範囲、命令バージョン V2.0: 開始/停止速度 ≤ 速度 ≤ 最大速度	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	パラメータ「Velocity」で指定された速度に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドを実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行時に、コマンドが別のコマンドによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

MC_MoveJog: ファンクションチャート V1...3

ファンクションチャート



設定ウィンドウ[**動的|全般**]で、以下の値が設定されました。

- 加速: 10.0
- 減速: 5.0

①	軸は、「Jog_F」によって、ジョグモードで正転方向に移動します。 ターゲット速度 50.0 に達すると、「InVelo_1」によって通知されます。 軸は「 Jog_F」がリセットされた後、再び停止状態まで減速します。
②	軸は、「Jog_B」によって、ジョグモードで逆転方向に移動します。 ターゲット速度 50.0 に達すると、「InVelo_1」によって通知されます。 軸は「 Jog_B」がリセットされた後、再び停止状態まで減速します。

MC_CommandTable

この章には下記に関する情報が記載されています：

- MC_CommandTable: 軸コマンドをモーションシーケンスとして実行 V2...3 (S7-1200)

MC_CommandTable: 軸コマンドをモーションシーケンスとして実行 V2...3

説明

モーションコントロール命令「MC_CommandTable」は、複数の個別の軸コントロールコマンドを 1 つの移動シーケンスに結合します。

必要条件

- 軸テクノロジーオブジェクトが V2 に挿入され、正しく設定済みであること。
- コマンドテーブルテクノロジーオブジェクトが挿入され、正しく設定済みであること。
- 軸が有効であること。

応答の無効化

MC_CommandTable コマンドは、以下のモーションコントロールコマンドによって中止できます。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

この新しい MC_CommandTable コマンドは、以下の動作中のモーションコントロールコマンドを中止します。

- MC_Home コマンド Mode = 3
- MC_Halt コマンド
- MC_MoveAbsolute コマンド
- MC_MoveRelative コマンド
- MC_MoveVelocity コマンド
- MC_MoveJog コマンド
- MC_CommandTable コマンド

有効なモーションコントロールコマンドは、最初の「Positioning Relative」、「Positioning Absolute」、「Velocity set point」、または「Halt」コマンドの開始によってキャンセルされます。

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト

Com-mandTable	INPUT	TO_Com-mandTa-ble_1	-	コマンドテーブルテクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドテーブルの開始	
StartStep	INPUT	INT	1	コマンドテーブルの実行を開始するステップを定義します 制限値: 1 ≤ StartStep ≤ EndStep	
EndStep	INPUT	INT	32	コマンドテーブルを実行する最後のステップを定義します。 制限値: StartStep ≤ EndStep ≤ 32	
Done	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルが正常に実行されました
Busy	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルが実行中です。
Comman-dAborted	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルが別のコマンドによってキャンセルされました。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドテーブルの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	
Current-Step	OUTPUT	INT	0	コマンドテーブルのステップが現在実行中	
StepCode	OUTPUT	WORD	16#0000	現在実行中のステップのユーザー定義の数値/ビットパターン	

MC_ChangeDynamic

この章には下記に関する情報が記載されています：

- MC_ChangeDynamic: 軸の動的設定の変更 V2...3 (S7-1200)

MC_ChangeDynamic: 軸の動的設定の変更 V2...3

説明

モーションコントロール命令「MC_ChangeDynamic」を使用して、軸の以下の設定を変更できます。

- ランプアップタイム(加速)値の変更
- ランプダウンタイム(減速)値の変更
- 緊急停止ランプダウンタイム(緊急停止減速)値の変更
- スムージングタイム(ジャーク)値の変更

変更の有効性については、*タグ*の説明を参照してください。

必要条件

- 軸テクノロジーオブジェクトが V2 に挿入されていること。
- 軸テクノロジーオブジェクトが正しく設定済みであること。

応答の無効化

MC_ChangeDynamic コマンドは、他のモーションコントロールコマンドで中止することはできません。

新しい MC_ChangeDynamic コマンドは、動作中のモーションコントロールコマンドを中止することはしません。

パラメータ

パラメータ	宣言	データタイプ	デフォルト値	説明	
Axis	INPUT	TO_AXIS_1	-	軸テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのコマンドの開始	
ChangeRampUp	INPUT	BOOL	FALSE	TRUE	入力パラメータ「RampUpTime」に従ったランプアップタイムの変更
RampUpTime	INPUT	REAL	5.00	軸を停止から設定された最大速度まで、ジャークリミットなしで加速する時間(秒単位)。 この変更は、タグ<軸名>. Config.DynamicDefaults.Acceleration 変更の有効性については、このタグの説明を参照してください。	
ChangeRampDown	INPUT	BOOL	FALSE	TRUE	入力パラメータ「RampDownTime」に従ったランプダウンタイムの変更
RampDownTime	INPUT	REAL	5.00	軸を設定された最大速度から停止まで、ジャークリミッタなしで減速する時間(秒単位)。	

				この変更は、タグ<軸名>. Config.DynamicDefaults.Deceleration 変更の有効性については、このタグの説明を参照してください。	
ChangeEmergency	INPUT	BOOL	FALSE	TRUE	入力パラメータ「EmergencyRampTime」に従った緊急停止ランプダウンタイムの変更
EmergencyRampTime	INPUT	REAL	2.00	軸を設定された最大速度から停止まで、緊急停止モードで、ジャークリミッタなしで減速する時間(秒)。 この変更は、タグ<軸名>. Config.DynamicDefaults.EmergencyDeceleration 変更の有効性については、このタグの説明を参照してください。	
ChangeJerkTime	INPUT	BOOL	FALSE	TRUE	入力パラメータ「JerkTime」に従ったスムージングタイムの変更
JerkTime	INPUT	REAL	0.25	軸の加速および減速ランプで使用されるスムージングタイム(秒単位) この変更は、タグ<軸名>. Config.DynamicDefaults.Jerk 変更の有効性については、このタグの説明を参照してください。	
Done	OUTPUT	BOOL	FALSE	TRUE	変更された値は、テクノロジーデータブロックに書き込み済みです。 タグの説明には、変更が有効になる時点が表示されます。
Error	OUTPUT	BOOL	FALSE	TRUE	コマンドの実行時にエラーが発生しました。 エラーの原因は、パラメータ「ErrorID」および「ErrorInfo」で検出することができます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	
ErrorInfo	OUTPUT	WORD	16#0000	パラメータ「ErrorID」のエラー情報 ID	

注記

入力パラメータ「RampUpTime」、「RampDownTime」、「EmergencyRampTime」、および「JerkTime」では、結果パラメータ: 「加速」、「減速」、「緊急停止」、および「ジャーク」の許容制限値を超える値を入力できます。

[軸テクノロジーオブジェクト|テクノロジーオブジェクトの設定|ダイナミクス]の等式と制限値に注意して、入力する値が有効な範囲内であることを確認してください。

S7-1500 モーションコントロール

この章には下記に関する情報が記載されています：

- S7-1500 モーションコントロール V2 (S7-1500)
- S7-1500 モーションコントロール V1 (S7-1500)

S7-1500 モーションコントロール V2

この章には下記に関する情報が記載されています：

- MC_Power (S7-1500)
- MC_Home (S7-1500)
- MC_MoveJog (S7-1500)
- MC_MoveVelocity (S7-1500)
- MC_MoveRelative (S7-1500)
- MC_MoveAbsolute (S7-1500)
- MC_MoveSuperimposed (S7-1500)
- MC_GearIn (S7-1500)
- MC_Halt (S7-1500)
- MC_Reset (S7-1500)

MC_Power

この章には下記に関する情報が記載されています：

- MC_Power: テクノロジーオブジェクトの有効化/無効化 V2 (S7-1500)
- MC_Power: ファンクションチャート V2 (S7-1500)

MC_Power: テクノロジーオブジェクトの有効化/無効化 V2

説明

モーションコントロール命令「MC_Power」は、テクノロジーオブジェクトを有効/無効にするために使用されます。

適用対象

- 同期軸
- 位置決め軸
- 速度軸
- 外部エンコーダ

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。

応答の無効化

- MC_Power ジョブは、他のすべてのモーションコントロールジョブで中止することはできません。
- パラメータが Enable = TRUE の MC_Power ジョブは、テクノロジーオブジェクトを有効にしますが、それによって、他のすべてのモーションコントロール命令を中止することはしません。
- テクノロジーオブジェクトの無効化(入力パラメータ「Enable」 = FALSE)は、選択した「Stop-Mode」に従って、関連するテクノロジーオブジェクトに対するすべてのモーションコントロールジョブを中止します。 ユーザーは、このプロセスを中止できません。

パラメータ

次の表に、「MC_POWER」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Axis	-	テクノロジーオブジェクト	
Enable	INPUT	BOOL	FALSE	TRUE	このテクノロジーオブジェクトが有効になります。
				FALSE	このテクノロジーオブジェクトは無効です。 このテクノロジーオブジェクトに対するすべての現在のジョブが、設定された「StopMode」に従って中止されます。
Stop-Mode	INPUT	INT	0	外部エンコーダテクノロジーオブジェクトには適用できません。 Enable パラメータの立ち下がりエッジでテクノロジーオブジェクトを無効にすると、軸は選択した StopMode に従って減速します。	

				0	緊急停止 このテクノロジーオブジェクトを無効にすると、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|緊急停止ランプ]で設定された緊急停止減速を使用して、ジャーク制限なしで停止状態まで制動します。 この場合、有効化は解除されます。 (<TO>.DynamicDefaults.EmergencyDeceleration)
				1	直ちに停止 テクノロジーオブジェクトを無効にすると、セットポイント 0 が出力され、有効化が解除されます。 軸はドライブでの設定に応じて制動し、停止します。
				2	最大動的値による停止 このテクノロジーオブジェクトを無効にすると、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|動的な制限値]で設定された最大減速を使用して、停止状態まで制動します。 同時に、設定された最大ジャークも考慮されます。 この場合、有効化は解除されます。 (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)
Status	OUTPUT	BOOL	FALSE	テクノロジーオブジェクトの有効ステータス	
				FALSE	無効 - 位置決め軸または速度軸では、モーションコントロールジョブはすべて承認されません。 - 速度コントロールおよび位置コントロールが動作中ではありません。 - テクノロジーオブジェクトの実際の値の有効性チェックは行われません。
				TRUE	有効 - 有効な位置決め軸または速度軸では、モーションコントロールジョブが承認されます。 - 速度コントロールおよび位置コントロールが動作中です。 - テクノロジーオブジェクトの実際の値が有効です。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
Error	OUTPUT	BOOL	FALSE	TRUE	モーションコントロール命令 MC_Power で、エラーが発生しました。 このエラーの原因は、「ErrorID」パラメータで検出できます。

ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID

テクノロジーオブジェクトの有効化

テクノロジーオブジェクトを有効にするには、その Enable パラメータを TRUE に設定します。

この Status パラメータが値 TRUE を示す場合、このテクノロジーオブジェクトは有効になります。

軸が移動中(現在速度が存在)にこのテクノロジーオブジェクトを有効にすると、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|動的な制限値](<TO>.DynamicLimits.MaxDeceleration)で設定された最大減速を使用して、セットポイント 0 まで制動します。 この制動ランプは、モーションコントロールジョブによって無効にできます。

注記

テクノロジーアラームの確認応答後の自動有効化

テクノロジーオブジェクトがテクノロジーアラームによって無効になると、アラームの原因が除去されてアラームが確認応答された後に、このオブジェクトは自動的に再び有効になります。 このためには、このプロセスの間、Enable パラメータが値 TRUE を保持していることが必要です。

テクノロジーオブジェクトの無効化

テクノロジーオブジェクトを無効にするには、その Enable パラメータを FALSE に設定します。

軸が移動中は、選択した「StopMode」に従って、軸が停止状態まで制動します。

「Busy」および「Status」パラメータが値 FALSE を示す場合は、テクノロジーオブジェクトの無効化が完了済みです。

PROFIdrive によるドライブ接続

ドライブを PROFIdrive と接続すると、セットポイント、有効化、およびドライブステータスが PROFIdrive フレームを使用して転送されます。

- **テクノロジーオブジェクトおよびドライブの有効化**

  テクノロジーオブジェクトの有効化には、パラメータ「"Enable" = TRUE」が使用されます。 ドライブは、PROFIdrive 標準に従って有効になります。

  タグ<TO>.StatusDrive.InOperation が値 TRUE を示す場合、ドライブがセットポイントを実行する準備が完了しています。 "Status"パラメータが値 TRUE に設定されます。

- **テクノロジーオブジェクトおよびドライブの無効化**

  パラメータ"Enable" = FALSE の場合、"Status"パラメータが値 FALSE に設定され、軸は選択した「StopMode」に従って制動します。 ドライブは、PROFIdrive 標準に従って無効になります。

アナログドライブ接続

セットポイントは、アナログ出力によって出力されます。 オプションでは、デジタル出力(<TO>.Actor.Interface.EnableDriveOutput)によって、許可信号とデジタル入力(<TO>.Actor.Interface.DriveReadyInput で準備完了信号を設定できます。

- **テクノロジーオブジェクトおよびドライブの有効化**

  パラメータ"Enable" = TRUE の場合、許可出力(「Enable drive output」)が設定されます。

  ドライブが、準備完了入力(「Drive ready input」)によって準備完了信号を返すと、"Status"パラメータと、テクノロジーオブジェクトの<TO>.StatusDrive.InOperation タグが TRUE に設定され、セットポイントがアナログ出力で切り替えられます。

- **テクノロジーオブジェクトおよびドライブの無効化**

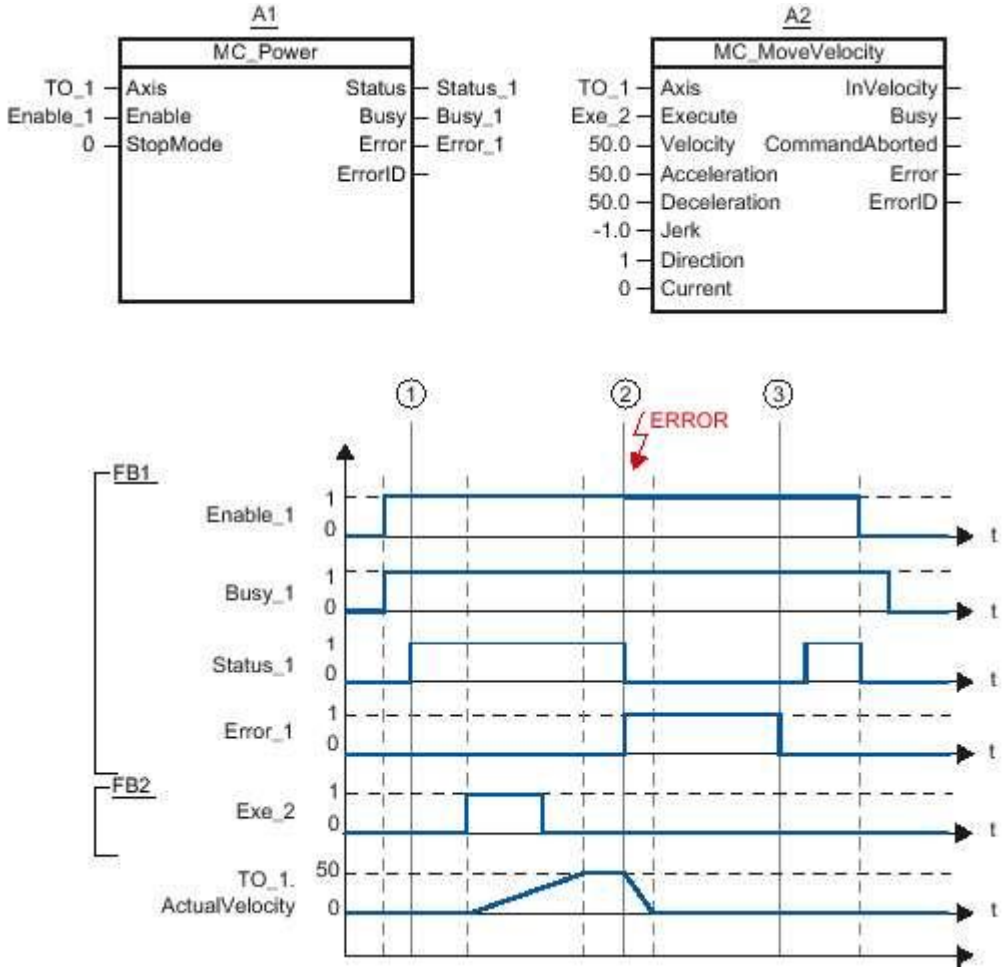
パラメータ"Enable" = FALSE の場合、"Status"パラメータが値 FALSE に設定され、軸は選択した「StopMode」に従って制動します。 セットポイント 0 に達すると、許可出力が FALSE に設定されます。

追加情報

テクノロジーオブジェクトおよびドライブの有効化および無効化に関する追加情報は、付録「MC_Power ファンクションチャート」を参照してください。

MC_Power: ファンクションチャート V2

ファンションチャート: テクノロジーオブジェクトの有効化とアラーム応答の例



テクノロジーオブジェクトは、「Enable_1 = TRUE」で有効になります。 有効化の成功は、時刻①に、「Status_1」から読み取ることができます。 この後、軸は「MC_MoveVelocity」ジョブによって移動します(A2)。 軸の速度プロファイルは、「Velocity Axis_1」から読み取ることができます。

時刻②に、テクノロジーオブジェクトのエラーが発生し、その結果、テクノロジーオブジェクトが無効になります(アラーム応答: 有効化の解除)。 軸はドライブでの設定に応じて制動し、停止します。 テクノロジーオブジェクトが無効になると、Status_1 がリセットされます。 軸は「Enable_1」 = FALSE を使用して無効にならなかったため、選択した「StopMode」は適用されません。 エラーの原因を訂正し、時刻③にアラームに確認応答します。

「Enable_1」はまだ設定されているため、テクノロジーオブジェクトが再び有効になります。 有効化の成功は、「Status_1」から読み取ることができます。 次に、テクノロジーオブジェクトは、「Enable_1」 = FALSE で無効になります。

MC_Home

この章には下記に関する情報が記載されています：

- MC_Home: Home テクノロジーオブジェクト、原点復帰位置の設定 V2 (S7-1500)

MC_Home: Home テクノロジーオブジェクト、原点復帰位置の設定 V2

説明

モーションコントロール命令「MC_Home」を使用して、テクノロジーオブジェクトの位置と機械の位置の間の関係を作成します。 同時に、テクノロジーオブジェクトの位置の値が、原点復帰マークに割り当てられます。 この原点復帰マークは、既知の機械の位置を表します。

原点復帰プロセスは、「Mode」パラメータで選択したモードと、[テクノロジーオブジェクト|設定|拡張パラメータ|原点復帰]での設定に従って、実行されます。

S7-1200 モーションコントロールおよび S7-1500 モーションコントロールの「MC_Home.Mode」パラメータは、テクノロジーバージョン V2.0 のフレームワークの中で標準化されました。この結果、「MC_Home.Mode」パラメータのパラメータ値が新たに割り当てられます。 テクノロジーバージョン V1.0 と V2.0 の「MC_Home.Mode」パラメータの比較については、「**バージョン概要**」セクションを参照してください。

[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定値]で事前に設定された値は、動的な値(加速、減速、ジャーク)で使用されます。

適用対象

- 同期軸
- 位置決め軸
- 外部エンコーダ

次の表は、テクノロジーオブジェクトごとに、可能であるモードを示します。

動作モード	インクリメンタルエンコーダ付き位置決め軸/同期軸	アブソリュートエンコーダ付き位置決め軸/同期軸	外部インクリメンタルエンコーダ	外部アブソリュートエンコーダ
アクティブ原点復帰 ("Mode" = 3, 5)	X	-	-	-
パッシブ原点復帰 ("Mode" = 2, 8, 10)	X	-	X	-
現在位置の設定 ("Mode" = 0)	X	X	X	X
現在位置の相対シフト ("Mode" = 1)	X	X	X	X
アブソリュートエンコーダの原点復帰 ("Mode" = 6, 7)	-	X	-	X

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。

- "Mode」 = 2、3、5、8、10
  テクノロジーオブジェクトが有効であること。
- "Mode」 = 0、1、6、7、8
  エンコーダの値が有効であること。 (<TO>.StatusSensor[n].State = 2)

応答の無効化

- パッシブ原点復帰用の MC_Home ジョブは、以下によって中止されます。
  - 「MC_Power.Enable」 = FALSE によるテクノロジーオブジェクトの無効化
  - "MC_Home」ジョブ(パラメータ「Mode」 = 3、5、9)
- パッシブ原点復帰用の「MC_Home」ジョブは、他のすべてのモーションコントロールジョブを中止しません。
- アクティブ原点復帰用の「MC_Home」ジョブは、以下によって中止されます。
  - 「MC_Power.Enable」 = FALSE のテクノロジーオブジェクトの無効化
  - "MC_Home」ジョブ 「Mode」 = 3、5
  - "MC_Halt」ジョブ
  - "MC_MoveAbsolute」ジョブ
  - "MC_MoveRelative」ジョブ
  - "MC_MoveVelocity」ジョブ
  - "MC_MoveJog」ジョブ
  - "MC_GearIn」ジョブ
- アクティブ原点復帰用の「MC_Home」ジョブは、以下の動作中のモーションコントロールジョブを中止します。
  - "MC_Home」ジョブ(パラメータ「Mode」 = 3、5、8、10
  - "MC_Halt」ジョブ
  - "MC_MoveAbsolute」ジョブ
  - "MC_MoveRelative」ジョブ
  - "MC_MoveVelocity」ジョブ
  - "MC_MoveJog」ジョブ
  - "MC_GearIn」ジョブ
  - "MC_MoveSuperimposed」ジョブ

パラメータ

次の表に、「MC_Home」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Position	INPUT	LREAL	0.0	選択した「Mode」モードに従って、指定した値が使用されます。	
Mode	INPUT	INT	0	動作モード	
				0	直接原点復帰(絶対値) テクノロジーオブジェクトの現在の位置として、パラメータ「Position」の値が設定されます。
				1	直接原点復帰(相対値)

	テクノロジーオブジェクトの現在の位置が、パラメータ「Position」の値だけシフトされます。
2	パッシブ原点復帰 (リセットなし) ファンクションが有効になったときに[原点復帰済み]ステータスがリセット**されない**[Mode] 8 などのファンクション。
3	アクティブ原点復帰 TO 位置決め軸/同期軸は、設定に従って、原点復帰移動を実行します。この移動が完了した後、軸は「Position」パラメータの値に位置決めされます。
4	予備
5	アクティブ原点復帰 (「Position」パラメータは影響しません) TO 位置決め軸/同期軸は、設定に従って、原点復帰移動を実行します。この移動が完了した後、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|原点復帰|アクティブ原点復帰]で設定された原点復帰位置に位置決めされます。 (<TO>.Homing.HomePosition)
6	アブソリュートエンコーダの調整 (相対値) 現在の位置が、パラメータ「Position」の値だけシフトされます。 計算された絶対値オフセットが、CPU に保持されて格納されます。 (<TO>.StatusSensor[n].AbsEncoderOffset)
7	アブソリュートエンコーダの調整 (絶対値) 現在の位置が、パラメータ「Position」の値に設定されます。 計算された絶対値オフセットが、CPU に保持されて格納されます。 (<TO>.StatusSensor[n].AbsEncoderOffset)
8	パッシブ原点復帰 原点復帰マークが検出されると、現在値が「Position」パラメータの値に設定されます。

				9	パッシブ原点復帰のキャンセル パッシブ原点復帰用の実行中のジョブが中止されます。
				10	パッシブ原点復帰 (「Position」パラメータは影響しません) 原点復帰マークが検出されると、現在値として、[テクノロジーオブジェクト|設定|拡張パラメータ|原点復帰|パッシブ原点復帰]で設定された原点復帰位置が設定されます。 (<TO>.Homing.HomePosition)
Done	OUTPUT	BOOL	FALSE	TRUE	ジョブ完了
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

「原点復帰済み」ステータスのリセット

テクノロジーオブジェクトの「原点復帰済み」ステータスは以下の条件でリセットされます(<TO>.StatusWord.X5 (HomingDone))。

- **現在の増分値を持つテクノロジーオブジェクト:**
  - 「MC_Home」ジョブの「Mode」= 3、5、8、10 での開始

    (原点復帰動作の正常終了後、「原点復帰済み」ステータスはリセットされます。)
  - エンコーダシステムのエラー、またはエンコーダ障害
  - テクノロジーオブジェクトの再起動
  - CPU の電源オフ -> 電源オンの後
  - メモリリセット
  - エンコーダ設定の変更
- **絶対現在値を持つテクノロジーオブジェクト:**
  - CPU 工場設定の復元
  - エンコーダ設定の変更
  - CPU の交換

「Mode」 = 1～8、10 でのテクノロジーオブジェクトの原点復帰

テクノロジーオブジェクトを原点復帰するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 「Mode」パラメータで、必要な原点復帰ファンクションを指定します。
3. 必要なパラメータを値で初期化し、「Execute」パラメータでの立ち上がりエッジで原点復帰動作を開始します。

「Done」パラメータが値 TRUE を示す場合、「MC_Home」ジョブが、選択した「Mode」に従って完了済みです。テクノロジーオブジェクトの「原点復帰済み」ステータスは、[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス|原点復帰済み]で示されます(<TO>.StatusWord.X5 (HomingDone))。

「Mode」 = 9 によるパッシブ原点復帰プロセスの終了

「Mode」 = 9 の場合、テクノロジーオブジェクトは原点復帰されません。実行中のパッシブ原点復帰用「MC_Home」ジョブ(「Mode」 = 2、8、10)が別の「MC_Home」ジョブ(「Mode」 = 9)によってオーバーライドされると、この実行中のジョブはパラメータ「CommandAborted」 = TRUE で終了します。 オーバーライドジョブ(「Mode」 = 9)は、パラメータ「Done」 = TRUE で実行が成功したことを通知します。

追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_MoveJog

この章には下記に関する情報が記載されています：

- MC_MoveJog: ジョグモードでの軸の移動 V2 (S7-1500)
- MC_MoveJog: ファンクションチャート V2 (S7-1500)

MC_MoveJog: ジョグモードでの軸の移動 V2

説明

モーションコントロール命令「MC_MoveJog」では、ジョグモードで軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

- 位置決め軸/同期軸:

  速度は、「Velocity」パラメータで指定されます。
- 速度軸:

  速度は、「Velocity」パラメータで指定されます。

適用対象

- 同期軸
- 位置決め軸
- 速度軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveJog」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ

「MC_MoveJog」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ

- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

パラメータ

次の表に、「MC_MoveJog」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Axis	InOut	TO_Speed-Axis	-	テクノロジーオブジェクト
JogForward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸はパラメータ「Velocity」で指定された速度で、正転方向に移動します。
JogBackward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸はパラメータ「Velocity」で指定された速度で、逆転方向に移動します。
Velocity	INPUT	LREAL	100.0	セットポイント速度/モーションプロセス用のセットポイント速度 値 > 0.0: 指定された値が使用されます。 値 < 0.0: 指定された値が使用されます。 ("Velocity」 = 0.0 は許可されます)
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0: 不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます。 値 = 0.0:台形速度プロファイル

				値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	セットポイント速度/セットポイント速さに達しました。今後この速度が保持されます。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「Error-ID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

セットポイント速度/セットポイント速さが 0 の場合の動作(「Velocity」 = 0.0)

「Velocity」 = 0.0 の「MC_MoveJog」ジョブは、設定された減速で軸を停止します。 セットポイント速度/セットポイント速さ 0 に達すると、パラメータ「InVelocity」が値 TRUE を示します。

[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス]で、「一定速度」と「停止」が表示されます(<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Stand-still))。

ジョグモードでの軸の移動

ジョグモードで軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 軸を「JogForward」で正転方向に移動するか、または「JogBackward」で逆転方向に移動します。

現在のモーション状態は、「Busy」、「InVelocity」、および「Error」に示されます。

注記

オーバーライドを変更するときの動作

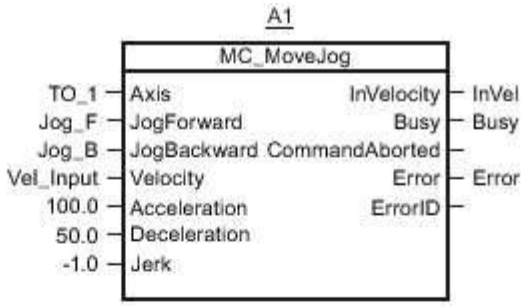
速度/速さが、一定の移動中にオーバーライド(<TO>.Override.Velocity)の変更によって影響を受けると、「InVelocity」パラメータが加速または減速中にリセットされます。 新しく計算された速度に達すると(「Velocity」 ×「Override」 %)、「InVelocity」が再び設定されます。

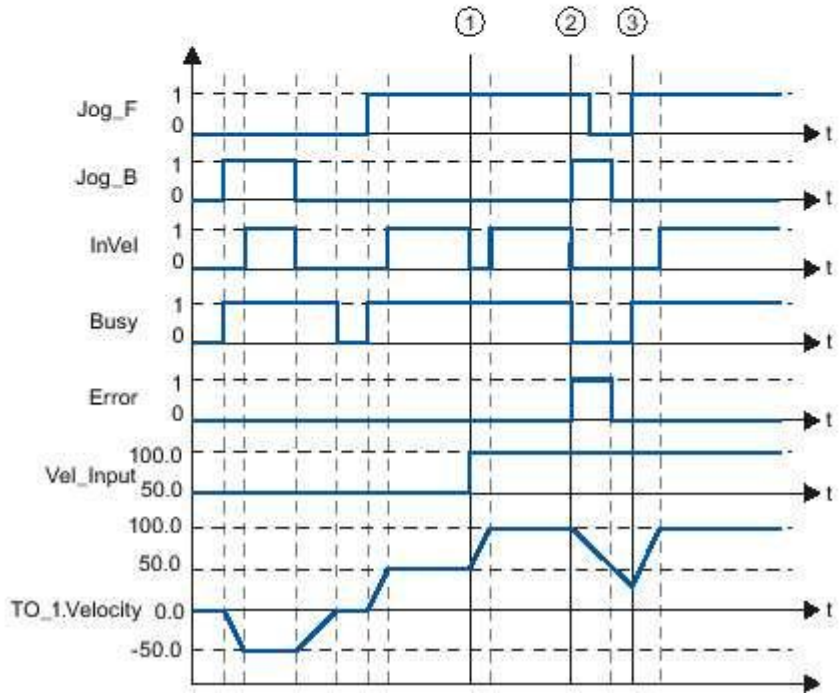
追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_MoveJog: ファンクションチャート V2

ファンションチャート: ジョグモードでの軸の移動





軸は、「Jog_B」によって、ジョグモードで逆転方向に移動します。 セットポイント速度-50.0 に達すると、このことが「InVel」 = TRUE を介して通知されます。 「Jog_B」がリセットされた後、軸は制動し、停止します。 この後、「Jog_F」によって、正転方向に移動します。 セットポイント速度 50.0 に達すると、このことが「InVel」 = TRUE を介して通知されます。

時刻①に「Jog_F」が設定されると、「Vel_Input」によって、セットポイント速度が 100.0 に変更されます。 その代わりに、速度オーバーライドによって、セットポイント速度を変更することもできます。 "InVel」がリセットされます。 軸が加速中です。 新しいセットポイント速度 100.0 に達すると、このことが「InVel」 = TRUE を介して通知されます。

「Jog_F」が設定されると、「Jog_B」が時刻②に同様に設定されます。 「Jog_F」と「Jog_B」の両方が設定されると、軸は適用可能な最後の減速で制動します。 エラーは「Error」によって示され、エラー 16#8007 の「ErrorID」(不正な方向指定)が出力されます。

このエラーは、2 つの入力「Jog_F」および「Jog_B」をリセットすることによって解決します。

制動ランプ中の時刻③に、「Jog_F」が設定されます。 軸は、最後に設定された速度まで加速されます。 セットポイント速度 100.0 に達すると、このことが「InVel」 = TRUE を介して通知されます。

MC_MoveVelocity

この章には下記に関する情報が記載されています：

- MC_MoveVelocity: 速度制御 V2 (S7-1500)
- MC_MoveVelocity: ファンクションチャート V2 (S7-1500)

MC_MoveVelocity: 速度制御 V2

説明

モーションコントロール命令「MC_MoveVelocity」では、一定の速度で軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

- 位置決め軸/同期軸:

  速度は、「Velocity」パラメータで指定されます。
- 速度軸:

  速度は、「Velocity」パラメータで指定されます。

適用対象

- 同期軸
- 位置決め軸
- 速度軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveVelocity」は以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ

「MC_MoveVelocity」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ

- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

パラメータ

次の表に、「MC_MoveVelocity」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Axis	InOut	TO_SpeedAxis	-	テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始
Velocity	INPUT	LREAL	100.0	セットポイント速度/モーションプロセス用のセットポイント速度 ("Velocity」 = 0.0 は許可されます)
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます。 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	0	軸の回転方向 0: 「Velocity」パラメータで指定された速度の符号が、回転方向を定義します。 1: 正転の回転方向

					「Velocity」の値が使用されます。
				2	逆転の回転方向 「Velocity」の値が使用されます。
Current	INPUT	BOOL	FALSE	現在の速度の維持	
				FALSE	無効 パラメータ「Velocity」および「Direction」の値が考慮されます。
				TRUE	有効 パラメータ「Velocity」および「Direction」の値は考慮されません。ファンクション開始時の現在の速度と方向が保持されます。 軸がファンクション開始時に現在速度だった速度でモーションを再開すると、「InVelocity"」パラメータが値 TRUE.を返します。
InVelocity	OUTPUT	BOOL	FALSE	TRUE	セットポイント速度/セットポイント速さに達しました。今後この速度が保持されます。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

セットポイント速度/セットポイント速さが 0 の場合の動作(「Velocity」 = 0.0)

「Velocity」 = 0.0 の「MC_MoveVelocity」ジョブは、設定された減速で軸を停止します。 セットポイント速度/セットポイント速さ 0 に達すると、パラメータ「InVelocity」が値 TRUE を示します。

[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス]で、「一定速度」と「停止」が表示されます(<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill))。

パラメータ「InVelocity」および「Busy」は、「MC_MoveVelocity」ジョブが別のモーションコントロールジョブによってオーバーライドされるまで、値 TRUE を示します。

一定の速度/速さでの軸の移動

一定の速度/速さで軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Velocity」で、軸を移動する速度/速さを指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveVelocity」ジョブを開始します。

現在のモーション状態は、「Busy」、「InVelocity」、および「Error」に示されます。

「InVelocity」パラメータが値 TRUE を示す場合、セットポイント速度/セットポイント速さに達しています。 軸は、この一定の速度で移動し続けます。 パラメータ「InVelocity」および「Busy」は、「MC_MoveVelocity」ジョブが別のモーションコントロールジョブによってオーバーライドされるまで、値 TRUE を示します。

注記

オーバーライドを変更するときの動作

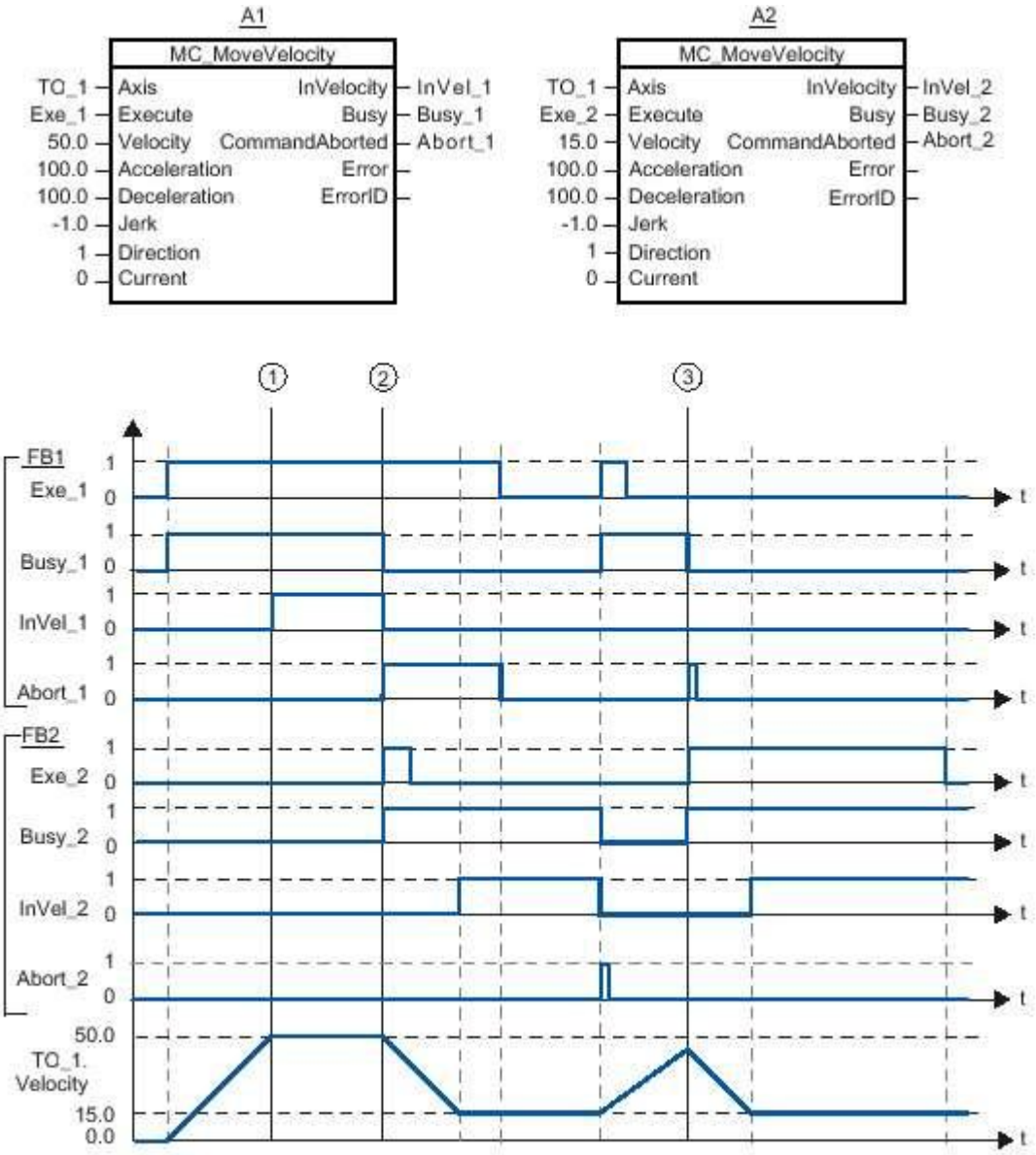
速度/速さが、一定の移動中にオーバーライド(<TO>.Override.Velocity)の変更によって影響を受けると、「InVelocity」パラメータが加速または減速中にリセットされます。 新しく計算された速度/速さに達すると(「Velocity」 ×「Override」 %)、「InVelocity」がリセットされます。

追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_MoveVelocity: ファンクションチャート V2

ファンションチャート: 速度の指定による軸の移動とオーバーライドジョブに対する応答



「Exe_1」によって開始された「MC_MoveVelocity」ジョブ(A1)は、軸を加速し、時刻①に、「InVel_1」を介して、セットポイント速度 50.0 に達したことを通知します。

時刻②に、このタスクは、別の「MC_MoveVelocity」ジョブ(A2)によって無効にされます。 この中止は、「Abort_1」によって通知されます。 新しいセットポイント速度 15.0 に達すると、「InVel_2」によって通知されます。 この後、軸は、一定の速度 15.0 で動作し続けます。

この実行中の「MC_MoveVelocity」ジョブ(A2)は、別の「MC_MoveVelocity」ジョブ(A1)によってオーバーライドされます。 この中止は、「Abort_2」によって通知されます。 軸は、新しいセットポイント速度 50.0 まで加速されます。 このセットポイント速度に達する前に、現在の「MC_MoveVelocity」ジョブ(A1)は、時刻③に、別の「MC_MoveVelocity」ジョブ(A2)によって無効にされます。 この中止

は、「Abort_1」によって通知されます。 新しいセットポイント速度 15.0 に達すると、「InVel_2」によって通知されます。 この後、軸は、一定の速度 15.0 で動作し続けます。

MC_MoveRelative

この章には下記に関する情報が記載されています：

- MC_MoveRelative: 軸の相対位置決め V2 (S7-1500)
- MC_MoveRelative: ファンクションチャート V2 (S7-1500)

MC_MoveRelative: 軸の相対位置決め V2

説明

モーションコントロール命令「MC_MoveRelative」では、ジョブ処理の開始時の位置を基準にして、相対的に軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

適用対象

- 同期軸
- 位置決め軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveRelative」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ

「MC_MoveRelative」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

パラメータ

次の表に、「MC_MoveRelative」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_PositioningAxis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Distance	INPUT	LREAL	0.0	位置決めプロセス用の距離 (逆転または正転)	
Velocity	INPUT	LREAL	-1.0	位置決め用のセットポイント速度 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された速度が使用されます。 (<TO>.DynamicDefaults.Velocity)	
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	ターゲット位置に達しました。

Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

開始位置を基準にして相対的に軸を移動

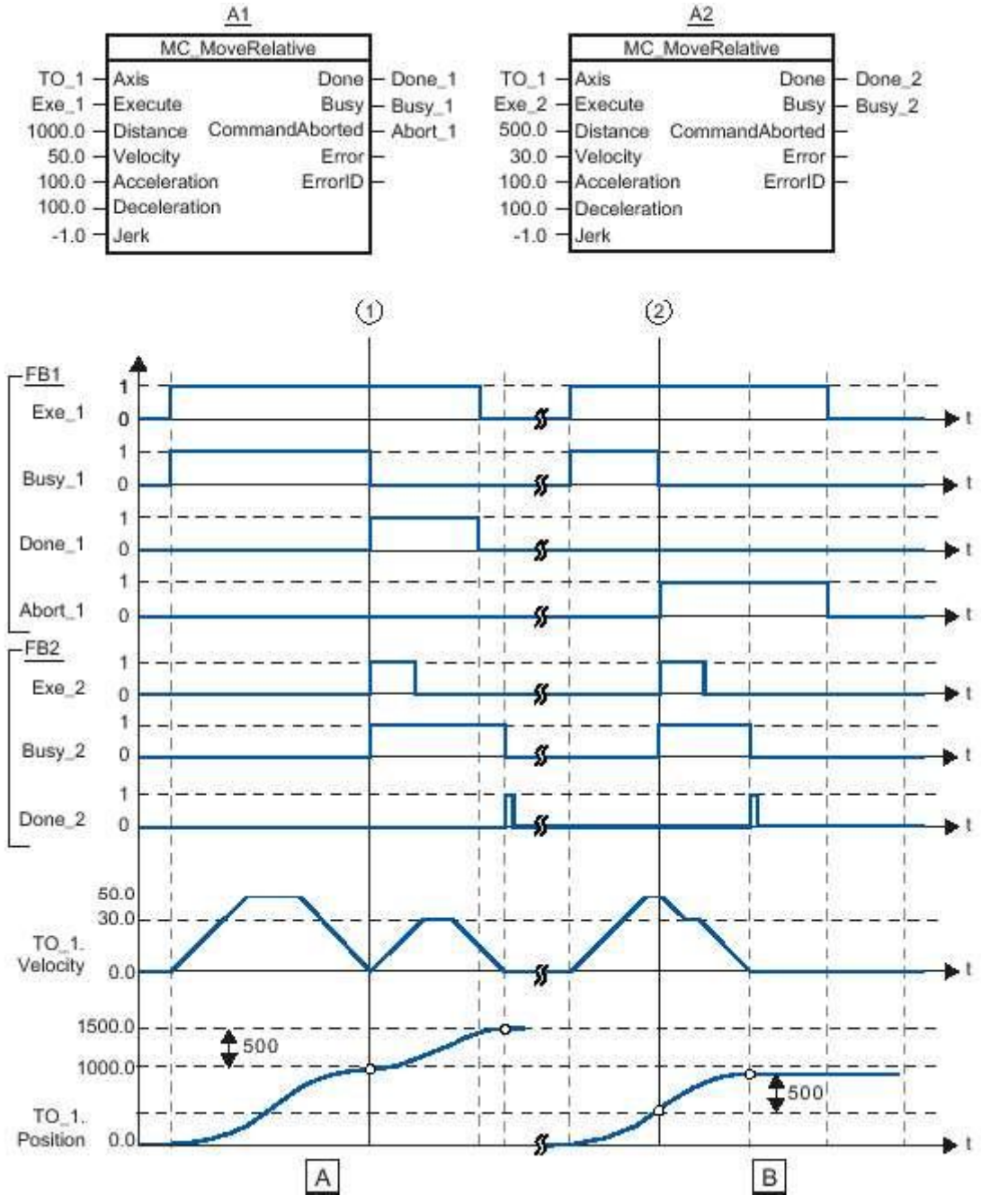
開始位置を基準にして相対的に軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 「Distance」パラメータで、移動する距離を指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveRelative」ジョブを開始します。

現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。

MC_MoveRelative: ファンクションチャート V2

ファンションチャート: 軸の相対位置決めと、オーバーライドジョブへの応答



セクション	軸は、「MC_MoveRelative」ジョブ(A1)によって、距離(「Distance」) 1000.0 だけ移動します (この場合、開始位置は 0.0 です)。 軸がターゲット位置に到達すると、このことが時刻① に「Done_1」によって通知されます。 この時刻①に、距離が 500.0 の別の「MC_MoveRelative」ジョブ(A2)が開始されます。 軸が新しいターゲット位置に到達すると、「Done_2」

A	によって通知されます。「Exe_2」が以前にリセットされたため、「Done_2」は 1 つのサイクルのみに適用されます。
セクション B	実行中の「MC_MoveRelative」ジョブ(A1)が、別の「MC_MoveRelative」ジョブ(A2)によってオーバーライドされます。 この中止は、時刻②に、「Abort_1」によって通知されます。 この後、軸は、新しい速度で距離(「Distance」) 500.0 だけ移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveAbsolute

この章には下記に関する情報が記載されています：

- MC_MoveAbsolute: 軸の絶対位置決め V2 (S7-1500)
- MC_MoveAbsolute: ファンクションチャート V2 (S7-1500)

MC_MoveAbsolute: 軸の絶対位置決め V2

説明

モーションコントロール命令「MC_MoveAbsolute」では、絶対位置まで軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

適用対象

- 同期軸
- 位置決め軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。
- テクノロジーオブジェクトが原点復帰済みであること。

応答の無効化

「MC_MoveAbsolute」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ

「MC_MoveAbsolute」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

パラメータ

次の表に、「MC_MoveAbsolute」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Axis	InOut	TO_Positioning-Axis	-	テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始
Position	INPUT	REAL	0.0	絶対ターゲット位置
Velocity	INPUT	LREAL	-1.0	位置決め用のセットポイント速度 値 > 0.0: 指定された値が使用されます。 値 = 0.0: 不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された速度が使用されます。 (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0: 固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0: 台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	3	軸の回転方向 「モジュロ」が有効な場合のみ評価されます。

				[テクノロジーオブジェクト|設定|基本パラメータ|モジュロの有効化]	
				1	正転の回転方向
				2	逆転の回転方向
				3	最短パス
Done	OUTPUT	BOOL	FALSE	TRUE	ターゲット位置に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「Error-ID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

絶対位置まで軸を移動

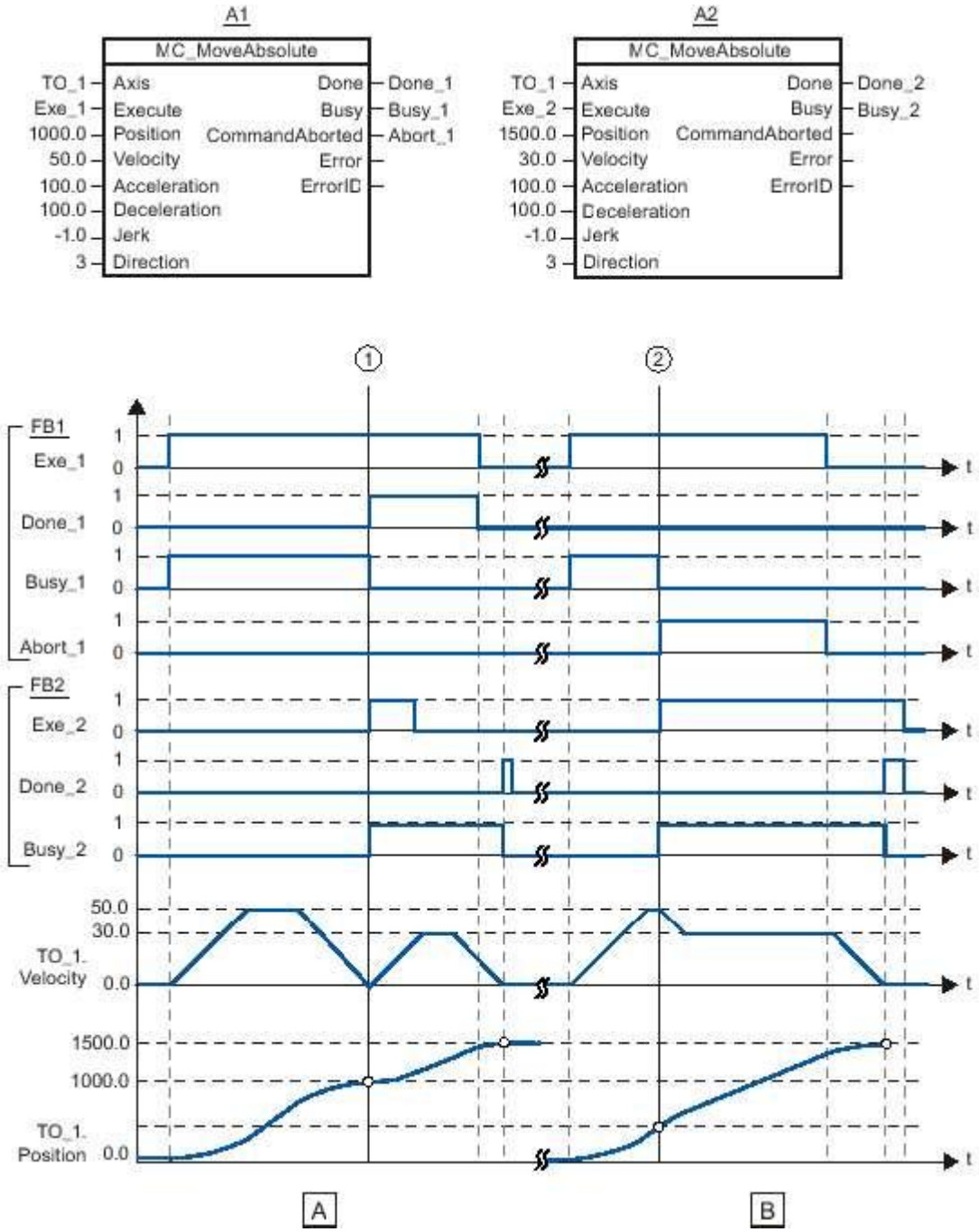
絶対位置まで軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 「Position」パラメータで、必要なターゲット位置を指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveAbsolute」ジョブを開始します。

現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。

MC_MoveAbsolute: ファンクションチャート V2

ファンションチャート: 軸の絶対位置決めと、オーバーライドジョブへの応答



セクション	軸が、「MC_MoveAbsolute」ジョブ(A1)によって、絶対位置 1000.0 まで移動します。 軸がターゲット位置に到達すると、このことが時刻①に「Done_1」によって通知されます。 この時刻①に、ターゲット位置が 1500.0 の別の「MC_MoveAbsolute」ジョブ(A2)が開始されます。 軸がターゲット位置 1500.0 に到達すると、このことが「Done_2」によって通知

A	されます。 「Exe_2」が以前にリセットされたため、「Done_2」は 1 つのサイクルのみに適用されます。
セクション B	実行中の「MC_MoveAbsolute」ジョブ(A1)が、時刻②に、別の「MC_MoveAbsolute」ジョブ(A2)によって無効にされます。 この中止は、「Abort_1」によって通知されます。 軸は変更された速度まで制動し、新しいターゲット位置 1500.0 まで移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveSuperimposed

この章には下記に関する情報が記載されています：

- MC_MoveSuperimposed: 軸の重ね合わされた位置決め V2 (S7-1500)
- MC_MoveSuperimposed: ファンクションチャート V2 (S7-1500)

MC_MoveSuperimposed: 軸の重ね合わされた位置決め V2 

説明

モーションコントロール命令[MC_MoveSuperimposed]を使用して、実行中の基本モーションに重ね合わせた相対位置決めモーションを開始できます。

移動中の動的動作は、パラメータ「VelocityDiff」、「Jerk」、「Acceleration」および「Deceleration」で定義されます。 基本モーションの値に動的値が加算されます。 基本モーションの持続時間は、重ね合わせたモーションで延長されません。

合計軸モーションのダイナミクスは、基本モーションと重ね合わせたモーションの動的値の合計です。

合計モーションの動作は、基本モーションのタイプによって異なります。

- 基本モーションは、単軸モーションです。
  - 重ね合わされたモーションの最大ダイナミクスは、基本モーションの現在の動的値と動的限界の差です。
  - 全体のモーションは、設定された動的限界に制限されます。
- 基本モーションは、同期モーションです。
  - 重ね合わされたモーションの最大ダイナミクスは、基本モーションの現在の動的値と動的限界の差です。
  - 従動軸の同期モーションは、従動軸の動的限界に制限されません。
  - 同期動作中の誘導軸の「MC_MoveSuperimposed」ジョブは、誘導軸と従動軸に影響します。
  - 同期動作中の従動軸の「MC_MoveSuperimposed」ジョブは、従動軸のみに影響します。

テクノロジーデータブロックおよび TIA ポータルに表示されるのは、常に合計モーションのダイナミクスです。

適用対象

- 同期軸
- 位置決め軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveSuperimposed」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

「MC_MoveSuperimposed」ジョブを開始すると、有効な「MC_MoveSuperimposed」ジョブが中止されます。

パラメータ

次の表に、[MC_MoveSuperimposed]モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Axis	InOut	TO_Positioning-Axis	-	軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始
Distance	INPUT	LREAL	0.0	重ね合わされた位置決め動作の追加の移動距離 (正または負)
VelocityDiff	INPUT	LREAL	-1.0	実行中のモーションと比較した最大速度偏差 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された速度が使用されます。 (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0:台形速度プロファイル

				値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
Done	OUT-PUT	BOOL	FALSE	TRUE	重ね合わされた位置決めが完了
Busy	OUT-PUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAbor-ted	OUT-PUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUT-PUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。このジョブは拒否されます。このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUT-PUT	WORD	0	パラメータ「Error」のエラー ID	

重ね合わされた位置決めモーションの開始

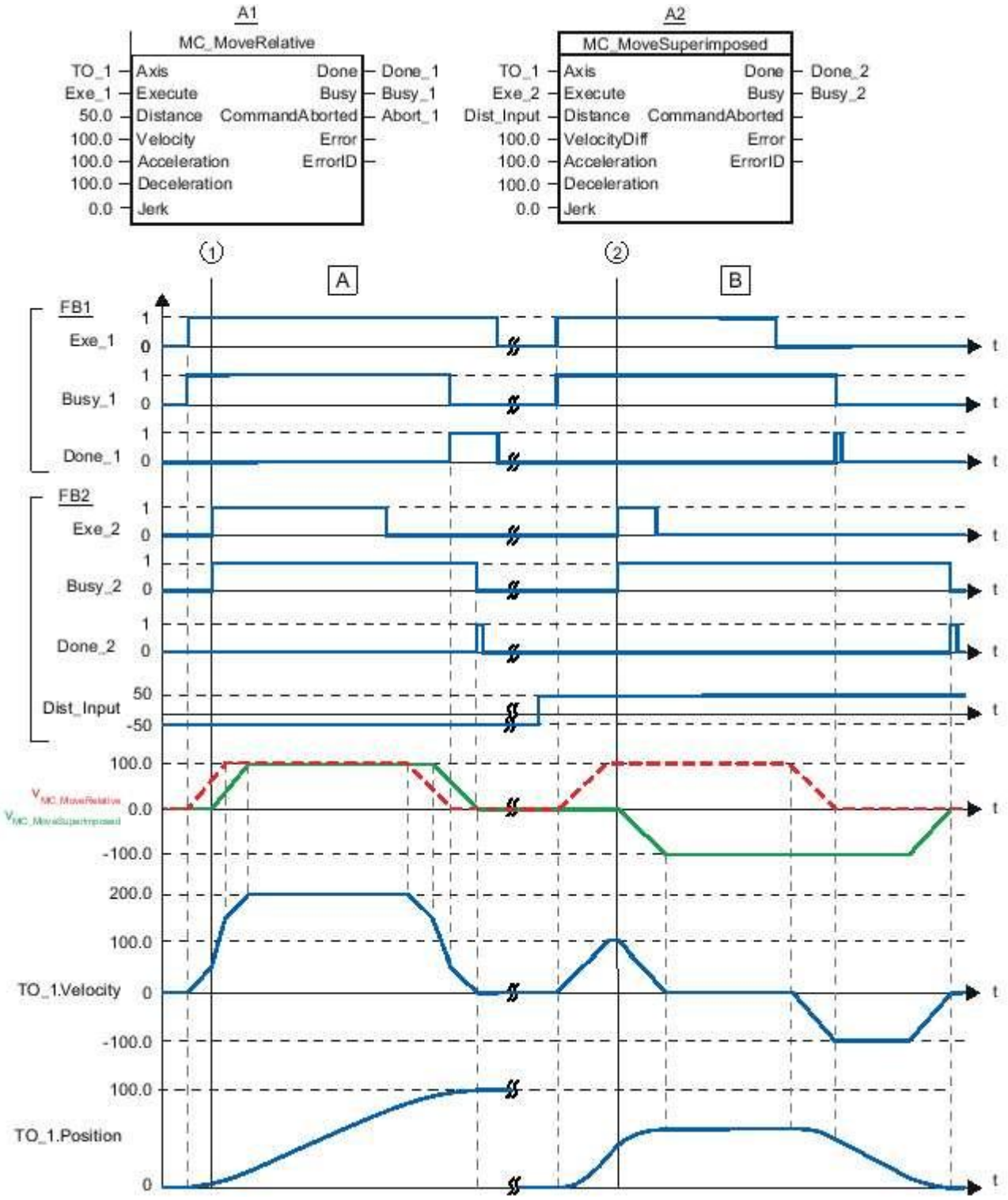
[MC_MoveSuperimposed]モーションコントロール命令で重ね合わせた位置決めモーションを開始するには、次の手順を実行します。

1. 上記の必要条件をチェックします。
2. 「Distance」パラメータで、移動する追加距離を指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveSuperimposed」ジョブを開始します。

現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。

MC_MoveSuperimposed: ファンクションチャート V2

ファンションチャート: 軸の重ね合わされた位置決め



セクション A	「Exe_1」を使用して、距離 50.0 の「MC_MoveRelative」ジョブを開始します。 時点①で、「Exe_2」を使用して、距離 50.0 の MC_MoveSuperimposed ジョブを開始します。 軸は距離 50 + 50 = 100.0 の両方のジョブの加算された動的値で移動されます。 軸がターゲット位置に到達すると、「Done_2」が通知されます。
セクション B	「Exe_1」を使用して、距離 50.0 の「MC_MoveRelative」ジョブを開始します。 時点②で、「Exe_2」を使用して、距離-50.0 の MC_MoveSuperimposed ジョブを開始します。 軸は距離 50.0 - 50.0 = 0.0 の両方のジョブの加算された動的値で移動されます。 軸がターゲット位置に到達すると、「Done_2」が通知されます。

MC_GearIn

この章には下記に関する情報が記載されています：

- MC_GearIn: 伝動 V2 の開始 (S7-1500)
- MC_GearIn: ファンクションチャート V2 (S7-1500)

MC_GearIn: 伝動 V2 の開始

説明

モーションコントロール命令[MC_GearIn]を使用して、誘導軸と従動軸間の相対伝動を開始できます。

同期中の従動軸の動的動作は、パラメータ「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

パラメータ「RatioNumerator」および「RatioDenominator」で 2 つの整数(分子/分母)の関係として、ギヤ比を指定します。 動作中に、新規"MC_GearIn"ジョブを配置して、ギヤ比を変更できます。 移行プロセスは、指定された動的値を使用して実行されます。

ギヤ比の分子は正または負で指定されます。 その結果、以下の応答が発生します。

- **正のギヤ比:**

  誘導軸および従動軸が同じ方向に移動します。
- **負のギヤ比**:

  従動軸が誘導軸と反対方向に回転します。

誘導軸が停止しているか、動作中に同期動作を開始できます。

同期中、誘導軸および従動軸間で位置のオフセットが発生する場合があります。 この位置オフセットは、「MC_GearIn」の開始時刻と、同期プロセスの持続時間によって異なります。 この位置オフセットは補正されません。

適用対象

- 同期軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- 誘導軸が、位置決め軸または同期軸であること。
- 従動軸が同期軸であること。
- 誘導軸が、[テクノロジーオブジェクト|設定|マスタ値の相互接続]の従動軸の設定で可能な誘導軸として指定されていること。
- 従動軸が有効になっていること。

応答の無効化

「MC_GearIn」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による従動軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ

- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ

「MC_GearIn」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

「MC_Power.Enable」= FALSE による誘導軸の無効化では、同期動作は中止されません。 ブレーキランプ中および誘導軸の再有効化後にも、従動軸は誘導軸に追従します。

パラメータ

次の表に、[MC_GearIn]モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Master	InOut	TO_Positioning-Axis	-	誘導軸テクノロジーオブジェクト
Slave	InOut	TO_Synchronous-Axis	-	従動軸テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始
RatioNumerator	INPUT	DINT	1	ギア比率の分子 許可された整数値: -2147483648 ~ 2147483648 (値 0 は許可されません)
RatioDenominator	INPUT	DINT	1	ギヤ比の分母 許可された整数値: 1 ~ 2147483648
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可

				値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0:台形速度プロファイル 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
InGear	OUT-PUT	BOOL	FALSE	TRUE	同期動作に到達 従動軸は誘導軸に同期され、同期して移動します。
Busy	OUT-PUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAbor-ted	OUT-PUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUT-PUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。このジョブは拒否されます。このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUT-PUT	WORD	0	パラメータ「Error」のエラー ID	

同期動作の開始

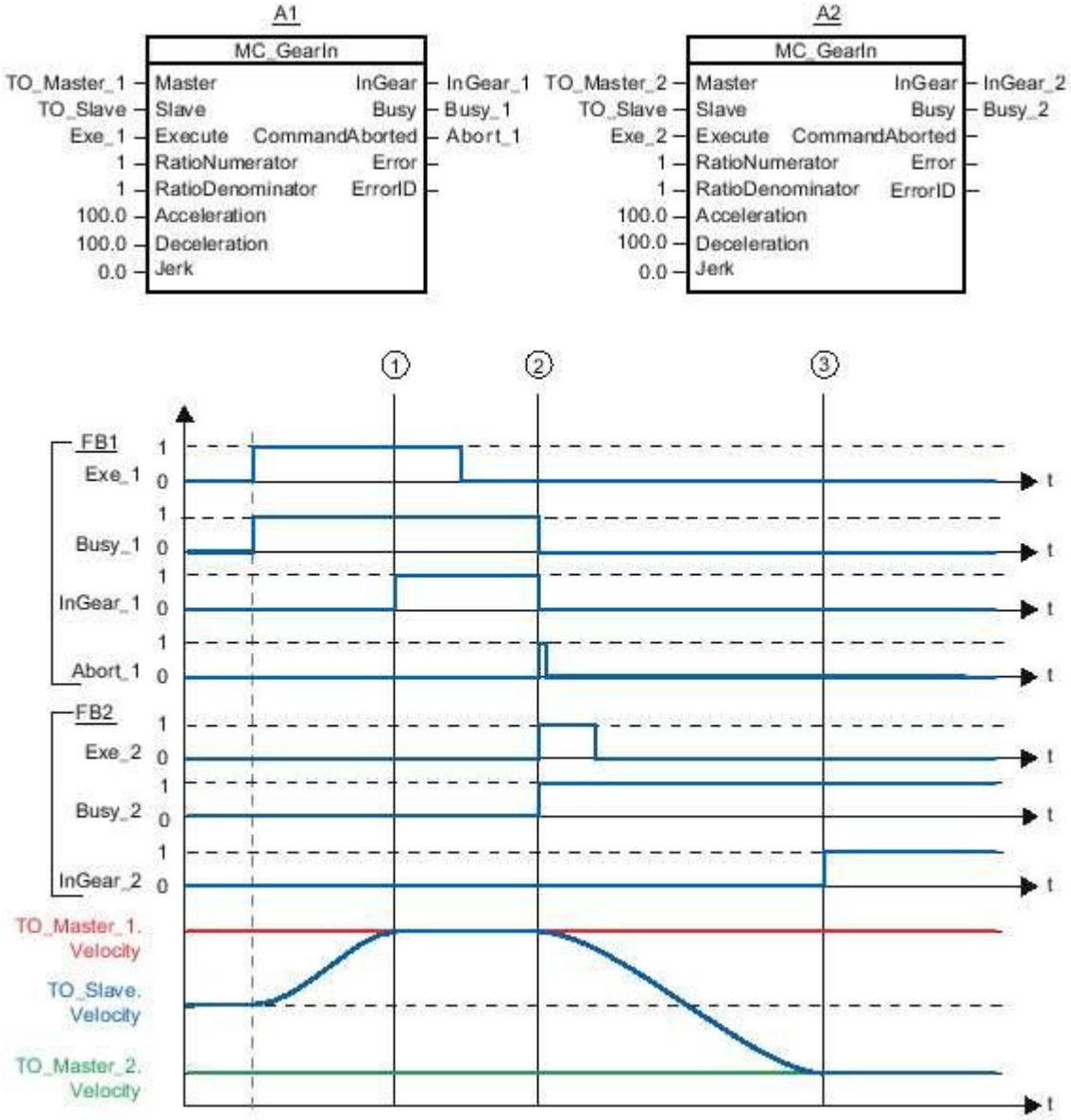
[MC_GearIn]モーションコントロール命令で同期動作を開始するには、次の手順を実行します。

1. 上記の必要条件をチェックします。
2. 対応しているパラメータで誘導軸、従動軸およびギヤ比を指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_GearIn」ジョブを開始します。

従動軸は誘導軸のマスタ値に同期されます。 「InGear」パラメータが真の値を表示する場合、従動軸は同期され、誘導軸と同期して移動します。 パラメータ「InGear」および「Busy」は、「MC_GearIn」ジョブが別のモーションコントロールジョブによってオーバーライドされるまで、真の値を示します。

MC_GearIn: ファンクションチャート V2

ファンションチャート: マスタ値の同期と切り替え



「Exe_1」を使用して、「MC_GearIn」ジョブ(A1)が開始されます。 従動軸(TO_Slave)が、誘導軸(TO_Master_1)に同期されます。 "InGear_1」は、時刻①に従動軸が同期して、誘導軸に同期して移動することを示します。

時刻②で、同期動作が別の「MC_GearIn」ジョブ(A2)によってオーバーライドされます。 この中止は、「Abort_1」によって通知されます。 従動軸が新規の誘導軸(TO_Master_2)に同期されます。 "InGear_2」は、時刻③に従動軸が同期して、誘導軸に同期して移動することを示します。

MC_Halt

この章には下記に関する情報が記載されています：

- MC_Halt: 軸の停止 V2 (S7-1500)
- MC_Halt: ファンクションチャート V2 (S7-1500)

MC_Halt: 軸の停止 V2

説明

モーションコントロール命令「MC_Halt」では、軸を停止状態まで制動することができます。

制動プロセス中の動的な動作は、パラメータ「Jerk」および「Deceleration」で定義されます。

適用対象

- 同期軸
- 位置決め軸
- 速度軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_Halt」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ

「MC_Halt」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「Mode」 = 3、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ
- "MC_GearIn」ジョブ
- "MC_MoveSuperimposed」ジョブ

パラメータ

次の表に、「MC_Halt」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Speed-Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	速度 0 に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

「MC_Halt」による軸の制動

軸を停止状態まで減速するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Deceleration」および「Jerk」に必要な値を設定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_Halt」ジョブを開始します。

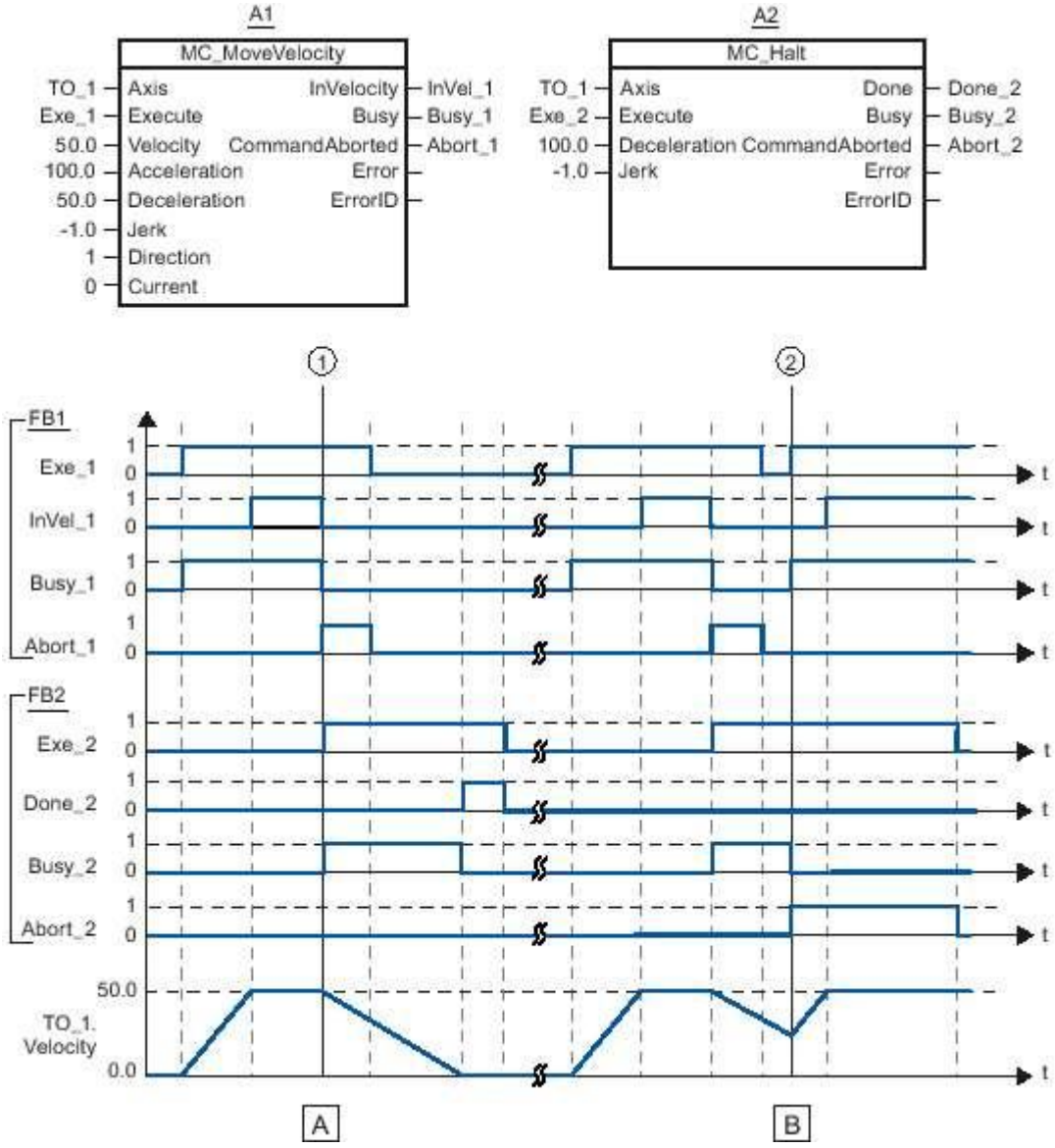
現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。軸の停止は、[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス|停止] (<TO>.StatusWord.X7 (Standstill))で示されます。

追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_Halt: ファンクションチャート V2

ファンションチャート: 軸の停止とオーバーライドジョブに対する応答



セクション A	軸は、「MC_MoveVelocity」ジョブ(A1)によって移動します。 セットポイント速度 50.0 に達すると、「InVel_1」によって通知されます。 時刻①に、この「MC_MoveVelocity」が、「MC_Halt」ジョブ(A2)によって無効にされます。 このジョブ終了は、「Abort_1」によって通知されます。 軸は停止状態まで制動します。 「MC_Halt」ジョブの正常終了が、「Done_2」によって通知されます。
セクション	軸は、「MC_MoveVelocity」ジョブ(A1)によって移動します。 セットポイント速度 50.0 に達すると、「InVel_1」によって通知されます。 次に、この「MC_MoveVelocity」が、「MC_Halt」ジョブ(A2)によって無効にされます。 このジョブ終了は、「Abort_1」によって通知されます。 制動プロセス中の時刻②に、「MC_Halt」が、「MC_MoveVelocity」ジョブ

B	(A1)によって無効にされます。 このジョブ終了は、「Abort_2」によって通知されます。 セットポイント速度 50.0 に達すると、「InVel_1」によって通知されます。 この後、軸は一定速度で移動します。

MC_Reset

この章には下記に関する情報が記載されています：

- MC_Reset: アラームの確認、テクノロジーオブジェクト V2 の再起動 (S7-1500)

MC_Reset: アラームの確認、テクノロジーオブジェクト V2 の再起動

説明

ユーザープログラムで確認応答できるすべてのテクノロジーアラームは、モーションコントロール命令「MC_Reset」で確認応答されます。 確認応答を行うと、テクノロジーデータブロックの「Error」および「Warning」ビットもリセットされます。

テクノロジーオブジェクトは、「Restart」 = TRUE のモーションコントロール命令「MC_Reset」を使用して再初期化(再起動)されます。 テクノロジーオブジェクトの再起動時、新しい設定データがテクノロジーデータブロックに適用されます。

適用対象

- 同期軸
- 位置決め軸
- 速度軸
- 外部エンコーダ

必要条件

- 保留中のテクノロジーアラームのエラーの原因が解決済みであること。
- 再起動するには、テクノロジーオブジェクトを無効にする必要があります。
  (「MC_Power.Status」 = FALSE および「MC_Power.Busy」 = FALSE)

応答の無効化

- パラメータ「Restart」 = FALSE:

  命令「MC_Reset」の処理は、他のモーションコントロールジョブが中止できます。 この MC_Reset ジョブは、すべての実行中のモーションコントロールジョブを中止しません。

- パラメータ「Restart」 = TRUE:

  パラメータ「Restart」 = TRUE の命令「MC_Reset」の処理は、他のすべてのモーションコントロールジョブが中止できません。

パラメータ

次の表に、「MC_Reset」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Restart	INPUT	BOOL	FALSE	TRUE	"Restart" テクノロージーオブジェクトの再初期化と、保留中のテクノロジーアラームの確認応答。 テクノロジーオブジェクトは、

					設定されたスタート値で、再初期化されます。
				FALSE	キューに登録されたテクノロジーアラームの確認応答。
Done	OUTPUT	BOOL	FALSE	TRUE	エラーが確認応答済みです。 再起動が実行済みです。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
Com-mandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

テクノロジーアラームへの確認応答

テクノロジーアラームに確認応答するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Restart」 = FALSE を設定します。
3. パラメータ「Execute」での立ち上がりエッジで、エラーの確認応答を開始します。

Done パラメータが値 TRUE を示す場合、エラーに確認応答済みです。

注記

「Restart」 = FALSE での確認応答

テクノロジアラームのみを確認する場合、「Restart」 = FALSE を設定します。 再起動中は、テクノロジーオブジェクトを使用できません。

テクノロジーオブジェクトの再起動

テクノロジーオブジェクトを再起動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Restart」 = TRUE を設定します。
3. 入力パラメータ「Execute」の立ち上がりエッジで、再起動を実行します。

この「Done」パラメータが値 TRUE を示す場合は、このテクノロジーオブジェクトの再起動が完了しています。

再起動に関する追加情報は、「テクノロジーオブジェクトの再起動」セクションを参照してください。

S7-1500 モーションコントロール V1

この章には下記に関する情報が記載されています：

- MC_Power (S7-1500)
- MC_Home (S7-1500)
- MC_MoveJog (S7-1500)
- MC_MoveVelocity (S7-1500)
- MC_MoveRelative (S7-1500)
- MC_MoveAbsolute (S7-1500)
- MC_Halt (S7-1500)
- MC_Reset (S7-1500)

MC_Power

この章には下記に関する情報が記載されています：

- MC_Power: テクノロジーオブジェクトの有効化/無効化 V1 (S7-1500)
- MC_Power: ファンクションチャート V1 (S7-1500)

MC_Power: テクノロジーオブジェクトの有効化/無効化 V1 

説明

モーションコントロール命令「MC_Power」は、テクノロジーオブジェクトを有効/無効にするために使用されます。

適用対象

- 位置決め軸
- 速度軸
- 外部エンコーダ

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。

応答の無効化

- MC_Power ジョブは、他のすべてのモーションコントロールジョブで中止することはできません。
- パラメータが Enable = TRUE の MC_Power ジョブは、テクノロジーオブジェクトを有効にしますが、それによって、他のすべてのモーションコントロール命令を中止することはしません。
- テクノロジーオブジェクトの無効化(入力パラメータ「Enable」 = FALSE)は、選択した「Stop-Mode」に従って、関連するテクノロジーオブジェクトに対するすべてのモーションコントロールジョブを中止します。 ユーザーは、このプロセスを中止できません。

パラメータ

次の表に、「MC_POWER」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Axis	-	テクノロジーオブジェクト	
Enable	INPUT	BOOL	FALSE	TRUE	このテクノロジーオブジェクトが有効になります。
				FALSE	このテクノロジーオブジェクトは無効です。 このテクノロジーオブジェクトに対するすべての現在のジョブが、設定された「StopMode」に従って中止されます。
Stop-Mode	INPUT	INT	0	外部エンコーダテクノロジーオブジェクトには適用できません。 Enable パラメータの立ち下がりエッジでテクノロジーオブジェクトを無効にすると、軸は選択した StopMode に従って減速します。	
				0	緊急停止

					このテクノロジーオブジェクトを無効にすると、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|緊急停止ランプ]で設定された緊急停止減速を使用して、ジャーク制限なしで停止状態まで制動します。 この場合、有効化は解除されます。 (<TO>.DynamicDefaults.EmergencyDeceleration)
				1	直ちに停止 テクノロジーオブジェクトを無効にすると、セットポイント 0 が出力され、有効化が解除されます。 軸はドライブでの設定に応じて制動し、停止します。
				2	最大動的値による停止 このテクノロジーオブジェクトを無効にすると、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|動的な制限値]で設定された最大減速を使用して、停止状態まで制動します。 同時に、設定された最大ジャークも考慮されます。 この場合、有効化は解除されます。 (<TO>.DynamicLimits.MaxDeceleration; <TO>.DynamicLimits.MaxJerk)
Status	OUTPUT	BOOL	FALSE	テクノロジーオブジェクトの有効ステータス	
				FALSE	無効 - 位置決め軸または速度軸では、モーションコントロールジョブはすべて承認されません。 - 速度コントロールおよび位置コントロールが動作中ではありません。 - テクノロジーオブジェクトの実際の値の有効性チェックは行われません。
				TRUE	有効 - 有効な位置決め軸または速度軸では、モーションコントロールジョブが承認されます。 - 速度コントロールおよび位置コントロールが動作中です。 - テクノロジーオブジェクトの実際の値が有効です。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
Error	OUTPUT	BOOL	FALSE	TRUE	モーションコントロール命令 MC_Power で、エラーが発生しました。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」の[エラー ID]	

テクノロジーオブジェクトの有効化

テクノロジーオブジェクトを有効にするには、その Enable パラメータを TRUE に設定します。

この Status パラメータが値 TRUE を示す場合、このテクノロジーオブジェクトは有効になります。

軸が移動中(現在速度が存在)にこのテクノロジーオブジェクトを有効にすると、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|動的な制限値](<TO>.DynamicLimits.MaxDeceleration)で設定された最大減速を使用して、セットポイント 0 まで制動します。 この制動ランプは、モーションコントロールジョブによって無効にできます。

注記

テクノロジーアラームの確認応答後の自動有効化

テクノロジーオブジェクトがテクノロジーアラームによって無効になると、アラームの原因が除去されてアラームが確認応答された後に、このオブジェクトは自動的に再び有効になります。 このためには、このプロセスの間、Enable パラメータが値 TRUE を保持していることが必要です。

テクノロジーオブジェクトの無効化

テクノロジーオブジェクトを無効にするには、その Enable パラメータを FALSE に設定します。

軸が移動中は、選択した「StopMode」に従って、軸が停止状態まで制動します。

「Busy」および「Status」パラメータが値 FALSE を示す場合は、テクノロジーオブジェクトの無効化が完了済みです。

PROFIdrive によるドライブ接続

ドライブを PROFIdrive と接続すると、セットポイント、有効化、およびドライブステータスが PROFIdrive フレームを使用して転送されます。

- **テクノロジーオブジェクトおよびドライブの有効化**

  テクノロジーオブジェクトの有効化には、パラメータ「"Enable" = TRUE」が使用されます。 ドライブは、PROFIdrive 標準に従って有効になります。

  タグ<TO>.StatusDrive.InOperation が値 TRUE を示す場合、ドライブがセットポイントを実行する準備が完了しています。 "Status"パラメータが値 TRUE に設定されます。

- **テクノロジーオブジェクトおよびドライブの無効化**

  パラメータ"Enable" = FALSE の場合、"Status"パラメータが値 FALSE に設定され、軸は選択した「StopMode」に従って制動します。 ドライブは、PROFIdrive 標準に従って無効になります。

アナログドライブ接続

セットポイントは、アナログ出力によって出力されます。 オプションでは、デジタル出力(<TO>.Actor.Interface.EnableDriveOutput)によって、許可信号とデジタル入力(<TO>.Actor.Interface.DriveReadyInput で準備完了信号を設定できます。

- **テクノロジーオブジェクトおよびドライブの有効化**

  パラメータ"Enable" = TRUE の場合、許可出力(「Enable drive output」)が設定されます。

  ドライブが、準備完了入力(「Drive ready input」)によって準備完了信号を返すと、"Status"パラメータと、テクノロジーオブジェクトの<TO>.StatusDrive.InOperation タグが TRUE に設定され、セットポイントがアナログ出力で切り替えられます。

- **テクノロジーオブジェクトおよびドライブの無効化**

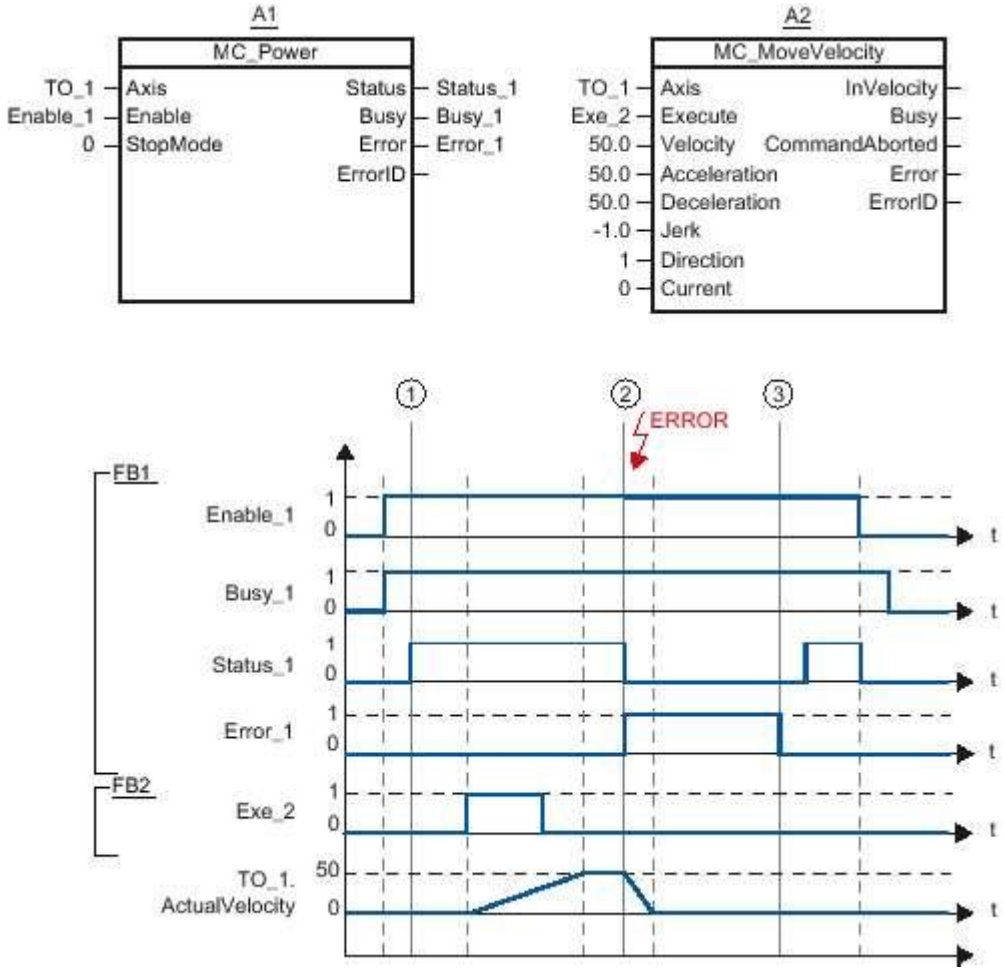
  パラメータ"Enable" = FALSE の場合、"Status"パラメータが値 FALSE に設定され、軸は選択した「StopMode」に従って制動します。 セットポイント 0 に達すると、許可出力が FALSE に設定されます。

追加情報

テクノロジーオブジェクトおよびドライブの有効化および無効化に関する追加情報は、付録「MC_Power ファンクションチャート」を参照してください。

MC_Power: ファンクションチャート V1

ファンションチャート: テクノロジーオブジェクトの有効化とアラーム応答の例



テクノロジーオブジェクトは、「Enable_1 = TRUE」で有効になります。 有効化の成功は、時刻①に、「Status_1」から読み取ることができます。 この後、軸は「MC_MoveVelocity」ジョブによって移動します(A2)。 軸の速度プロファイルは、「Velocity Axis_1」から読み取ることができます。

時刻②に、テクノロジーオブジェクトのエラーが発生し、その結果、テクノロジーオブジェクトが無効になります(アラーム応答:有効化の解除)。 軸はドライブでの設定に応じて制動し、停止します。 テクノロジーオブジェクトが無効になると、Status_1 がリセットされます。 軸は「Enable_1」= FALSE を使用して無効にならなかったため、選択した「StopMode」は適用されません。 エラーの原因を訂正し、時刻③にアラームに確認応答します。

「Enable_1」はまだ設定されているため、テクノロジーオブジェクトが再び有効になります。 有効化の成功は、「Status_1」から読み取ることができます。 次に、テクノロジーオブジェクトは、「Enable_1」 = FALSE で無効になります。

MC_Home

この章には下記に関する情報が記載されています：

- MC_Home: Home テクノロジーオブジェクト、原点復帰位置の設定 V1 (S7-1500)

MC_Home: Home テクノロジーオブジェクト、原点復帰位置の設定 V1

説明

モーションコントロール命令「MC_Home」を使用して、テクノロジーオブジェクトの位置と機械の位置の間の関係を作成します。 同時に、テクノロジーオブジェクトの位置の値が、原点復帰マークに割り当てられます。 この原点復帰マークは、既知の機械の位置を表します。

原点復帰プロセスは、「HomingMode」パラメータで選択したモードと、[テクノロジーオブジェクト|設定|拡張パラメータ|原点復帰]での設定に従って、行われます。

[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定値]で事前に設定された値は、動的な値(加速、減速、ジャーク)で使用されます。

適用対象

- 位置決め軸
- 外部エンコーダ

次の表は、テクノロジーオブジェクトごとに、可能であるモードを示します。

動作モード	インクリメンタルエンコーダ付き位置決め軸	アブソリュートエンコーダ付き位置決め軸	外部インクリメンタルエンコーダ	外部アブソリュートエンコーダ
アクティブ原点復帰(HomingMode = 4、5)	X	-	-	-
パッシブ原点復帰(HomingMode = 2、3、8)	X	-	X	-
位置決め値の設定 ("HomingMode" = 0)	X	X	X	X
現在値の相対シフト ("HomingMode" = 1)	X	X	X	X
アブソリュートエンコーダの原点復帰 ("HomingMode" = 6, 7)	-	X	-	X

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- "HomingMode」 = 2、3、4、5、8
  テクノロジーオブジェクトが有効であること。

- "HomingMode" = 0、1、2、6、7
  エンコーダの値が有効であること。 (<TO>.StatusSensor[n].State = 2)

応答の無効化

- パッシブ原点復帰用の MC_Home ジョブは、以下によって中止されます。
  - 「MC_Power.Enable」 = FALSE によるテクノロジーオブジェクトの無効化
  - "MC_Home」ジョブ(パラメータ「HomingMode」 = 4、5、9)
- パッシブ原点復帰用の「MC_Home」ジョブは、他のすべてのモーションコントロールジョブを中止しません。
- アクティブ原点復帰用の「MC_Home」ジョブは、以下によって中止されます。
  - 「MC_Power.Enable」 = FALSE のテクノロジーオブジェクトの無効化
  - "MC_Home」ジョブ 「HomingMode」 = 4、5
  - "MC_Halt」ジョブ
  - "MC_MoveAbsolute」ジョブ
  - "MC_MoveRelative」ジョブ
  - "MC_MoveVelocity」ジョブ
  - "MC_MoveJog」ジョブ
- アクティブ原点復帰用の「MC_Home」ジョブは、以下の動作中のモーションコントロールジョブを中止します。
  - "MC_Home」ジョブ(パラメータ「HomingMode」 = 2、3、4、5)
  - "MC_Halt」ジョブ
  - "MC_MoveAbsolute」ジョブ
  - "MC_MoveRelative」ジョブ
  - "MC_MoveVelocity」ジョブ
  - "MC_MoveJog」ジョブ

パラメータ

次の表に、「MC_Home」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Position	INPUT	LREAL	0.0	選択した「HomingMode」モードに従って、指定した値が使用されます。	
HomingMode	INPUT	INT	4	動作モード	
				0	直接原点復帰(絶対値) テクノロジーオブジェクトの現在の位置として、パラメータ「Position」の値が設定されます。
				1	直接原点復帰(相対値) テクノロジーオブジェクトの現在の位置が、パラメータ「Position」の値だけシフトされます。
				2	パッシブ原点復帰

	原点復帰マークが検出されると、現在値が「Position」パラメータの値に設定されます。
3	パッシブ原点復帰 (「Position」パラメータは影響しません) 原点復帰マークが検出されると、現在値として、[テクノロジーオブジェクト|設定|拡張パラメータ|原点復帰|パッシブ原点復帰]で設定された原点復帰位置が設定されます。 (<TO>.Homing.HomePosition)
4	アクティブ原点復帰 TO 位置決め軸は、設定に従って、原点復帰移動を実行します。 この移動が完了した後、軸は「Position」パラメータの値に位置決めされます。
5	アクティブ原点復帰 (「Position」パラメータは影響しません) TO 位置決め軸は、設定に従って、原点復帰移動を実行します。 この移動が完了した後、軸は[テクノロジーオブジェクト|設定|拡張パラメータ|原点復帰|アクティブ原点復帰]で設定された原点復帰位置に位置決めされます。 (<TO>.Homing.HomePosition)
6	アブソリュートエンコーダの調整(相対値) 現在の位置が、パラメータ「Position」の値だけシフトされます。 計算された絶対値オフセットが、CPU に保持されて格納されます。 (<TO>.StatusSensor[n].AbsEncoderOffset)
7	アブソリュートエンコーダの調整(絶対値) 現在の位置が、パラメータ「Position」の値に設定されます。 計算された絶対値オフセットが、CPU に保持されて格納されます。 (<TO>.StatusSensor[n].AbsEncoderOffset)
8	パッシブ原点復帰 (リセットなし)

					ファンクションが有効になったときに[原点復帰済み]ステータスがリセット**されない**[HomingMode] 2 などのファンクション。
				9	パッシブ原点復帰のキャンセル パッシブ原点復帰用の実行中のジョブが中止されます。
Done	OUTPUT	BOOL	FALSE	TRUE	ジョブ完了
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

「原点復帰済み」ステータスのリセット

テクノロジーオブジェクトの「原点復帰済み」ステータスは以下の条件でリセットされます(<TO>.StatusWord.X5 (HomingDone))。

- **現在の増分値を持つテクノロジーオブジェクト:**
  - 「MC_Home」ジョブの「HomingMode」 = 2、3、4、5 での開始

    (原点復帰動作の正常終了後、「原点復帰済み」ステータスはリセットされます。)
  - エンコーダシステムのエラー、またはエンコーダ障害
  - テクノロジーオブジェクトの再起動
  - CPU の電源オフ -> 電源オンの後
  - メモリリセット
  - エンコーダ設定の変更
- **絶対現在値を持つテクノロジーオブジェクト:**
  - CPU 工場設定の復元
  - エンコーダ設定の変更
  - CPU の交換

「HomingMode」 = 1、8 でのテクノロジーオブジェクトの原点復帰

テクノロジーオブジェクトを原点復帰するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 「HomingMode」パラメータで、必要な原点復帰ファンクションを指定します。
3. 必要なパラメータを値で初期化し、「Execute」パラメータでの立ち上がりエッジで原点復帰動作を開始します。

「Done」パラメータが値 TRUE を示す場合、「MC_Home」ジョブが、選択した「HomingMode」に従って完了済みです。テクノロジーオブジェクトの「原点復帰済み」ステータスは、[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス|原点復帰済み]で示されます(<TO>.StatusWord.X5 (HomingDone))。

「HomingMode」 = 9 によるパッシブ原点復帰プロセスの終了

「HomingMode」 = 9 の場合、テクノロジーオブジェクトは原点復帰されません。実行中のパッシブ原点復帰用「MC_Home」ジョブ(「HomingMode」 = 2、3、8)が別の「MC_Home」ジョブ(「HomingMode」 = 9)によってオーバーライドされると、この実行中のジョブはパラメータ「CommandAborted」 = TRUE で終了します。オーバーライドジョブ(「HomingMode」 = 9)は、パラメータ「Done」 = TRUE で実行が成功したことを通知します。

追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_MoveJog

この章には下記に関する情報が記載されています：

- MC_MoveJog: ジョグモードでの軸の移動 V1 (S7-1500)
- MC_MoveJog: ファンクションチャート V1 (S7-1500)

MC_MoveJog: ジョグモードでの軸の移動 V1

説明

モーションコントロール命令「MC_MoveJog」では、ジョグモードで軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

- 位置決め軸

  速度は、「Velocity」パラメータで指定されます。
- 速度軸:

  速度は、「Velocity」パラメータで指定されます。

適用対象

- 位置決め軸
- 速度軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveJog」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

「MC_MoveJog」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

パラメータ

次の表に、「MC_MoveJog」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Speed-Axis	-	テクノロジーオブジェクト	
JogForward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸はパラメータ「Velocity」で指定された速度で、正転方向に移動します。	
JogBackward	INPUT	BOOL	FALSE	このパラメータが TRUE である限り、軸はパラメータ「Velocity」で指定された速度で、逆転方向に移動します。	
Velocity	INPUT	LREAL	100.0	セットポイント速度/モーションプロセス用のセットポイント速度 値 > 0.0: 指定された値が使用されます。 値 < 0.0: 指定された値が使用されます。 ("Velocity」 = 0.0 は許可されます)	
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0: 不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます。 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
InVelocity	OUTPUT	BOOL	FALSE	TRUE	セットポイント速度/セットポイント速さに達しました。今後この速度が保持されます。

Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

セットポイント速度/セットポイント速さが 0 の場合の動作(「Velocity」 = 0.0)

「Velocity」 = 0.0 の「MC_MoveJog」ジョブは、設定された減速で軸を停止します。 セットポイント速度/セットポイント速さ 0 に達すると、パラメータ「InVelocity」が値 TRUE を示します。

[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス]で、「一定速度」と「停止」が表示されます(<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill))。

ジョグモードでの軸の移動

ジョグモードで軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 軸を「JogForward」で正転方向に移動するか、または「JogBackward」で逆転方向に移動します。

現在のモーション状態は、「Busy」、「InVelocity」、および「Error」に示されます。

注記

オーバーライドを変更するときの動作

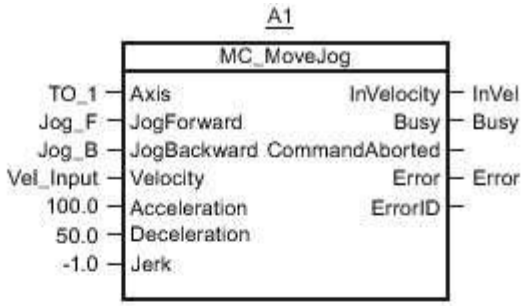
速度/速さが、一定の移動中にオーバーライド(<TO>.Override.Velocity)の変更によって影響を受けると、「InVelocity」パラメータが加速または減速中にリセットされます。 新しく計算された速度に達すると(「Velocity」 ×「Override」 %)、「InVelocity」が再び設定されます。

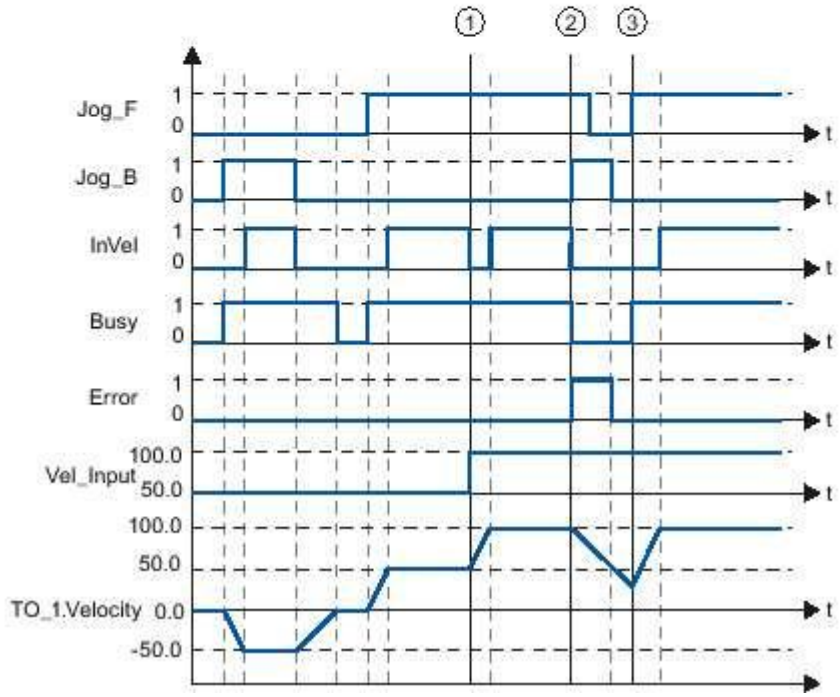
追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_MoveJog: ファンクションチャート V1

ファンションチャート: ジョグモードでの軸の移動





軸は、「Jog_B」によって、ジョグモードで逆転方向に移動します。 セットポイント速度-50.0 に達すると、このことが「InVel」 = TRUE によって通知されます。 「Jog_B」がリセットされた後、軸は制動して停止します。 この後、「Jog_F」によって、正転方向に移動します。 セットポイント速度 50.0 に達すると、このことが「InVel」 = TRUE によって通知されます。

時刻①に「Jog_F」が設定されると、「Vel_Input」によって、セットポイント速度が 100.0 に変更されます。 その代わりに、速度オーバーライドによって、セットポイント速度を変更することもできます。 "InVel」がリセットされます。 軸が加速中です。 新しいセットポイント速度 100.0 に達すると、このことが「InVel」 = TRUE によって通知されます。

「Jog_F」が設定されると、「Jog_B」が時刻②に同様に設定されます。 「Jog_F」と「Jog_B」の両方が設定されると、軸は適用可能な最後の減速で制動します。 エラーは「Error」によって示され、エラー 16#8007 の「ErrorID」(不正な方向指定)が出力されます。

このエラーは、2 つの入力「Jog_F」および「Jog_B」をリセットすることによって解決します。

制動ランプ中の時刻③に、「Jog_F」が設定されます。 軸は、最後に設定された速度まで加速されます。 セットポイント速度 100.0 に達すると、このことが「InVel」 = TRUE によって通知されます。

MC_MoveVelocity

この章には下記に関する情報が記載されています：

- MC_MoveVelocity: あらかじめ定義された指定速度での軸の移動 V1 (S7-1500)
- MC_MoveVelocity: ファンクションチャート V1 (S7-1500)

MC_MoveVelocity: あらかじめ定義された指定速度での軸の移動 V1

説明

モーションコントロール命令「MC_MoveVelocity」では、一定の速度で軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

- 位置決め軸

  速度は、「Velocity」パラメータで指定されます。
- 速度軸:

  速度は、「Velocity」パラメータで指定されます。

適用対象

- 位置決め軸
- 速度軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveVelocity」は以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

「MC_MoveVelocity」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

パラメータ

次の表に、「MC_MoveVelocity」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Axis	InOut	TO_Speed-Axis	-	テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始
Velocity	INPUT	LREAL	100.0	セットポイント速度/モーションプロセス用のセットポイント速度 ("Velocity」 = 0.0 は許可されます)
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます。 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	0	軸の回転方向 0: 「Velocity」パラメータで指定された速度の符号が、回転方向を定義します。 1: 正転の回転方向 「Velocity」の値が使用されます。 2: 逆転の回転方向 「Velocity」の値が使用されます。
Current	INPUT	BOOL	FALSE	現在の速度の維持

				FALSE	無効 パラメータ「Velocity」および「Direction」の値が考慮されます。
				TRUE	有効 パラメータ「Velocity」および「Direction」の値は考慮されません。ファンクション開始時の現在の速度と方向が保持されます。 軸がファンクション開始時に現在速度だった速度でモーションを再開すると、「InVelocity"」パラメータが値 TRUE.を返します。
InVelocity	OUTPUT	BOOL	FALSE	TRUE	セットポイント速度/セットポイント速さに達しました。今後この速度が保持されます。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

セットポイント速度/セットポイント速さが 0 の場合の動作(「Velocity」 = 0.0)

「Velocity」 = 0.0 の「MC_MoveVelocity」ジョブは、設定された減速で軸を停止します。 セットポイント速度/セットポイント速さ 0 に達すると、パラメータ「InVelocity」が値 TRUE を示します。

[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス]で、「一定速度」と「停止」が表示されます(<TO>.StatusWord.X12 (ConstantVelocity); <TO>.StatusWord.X7 (Standstill))。

パラメータ「InVelocity」および「Busy」は、「MC_MoveVelocity」ジョブが別のモーションコントロールジョブによってオーバーライドされるまで、値 TRUE を示します。

一定の速度/速さでの軸の移動

一定の速度/速さで軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Velocity」で、軸を移動する速度/速さを指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveVelocity」ジョブを開始します。

現在のモーション状態は、「Busy」、「InVelocity」、および「Error」に示されます。

「InVelocity」パラメータが値 TRUE を示す場合、セットポイント速度/セットポイント速さに達しています。 軸は、この一定の速度で移動し続けます。 パラメータ「InVelocity」および「Busy」は、「MC_MoveVelocity」ジョブが別のモーションコントロールジョブによってオーバーライドされるまで、値 TRUE を示します。

注記

オーバーライドを変更するときの動作

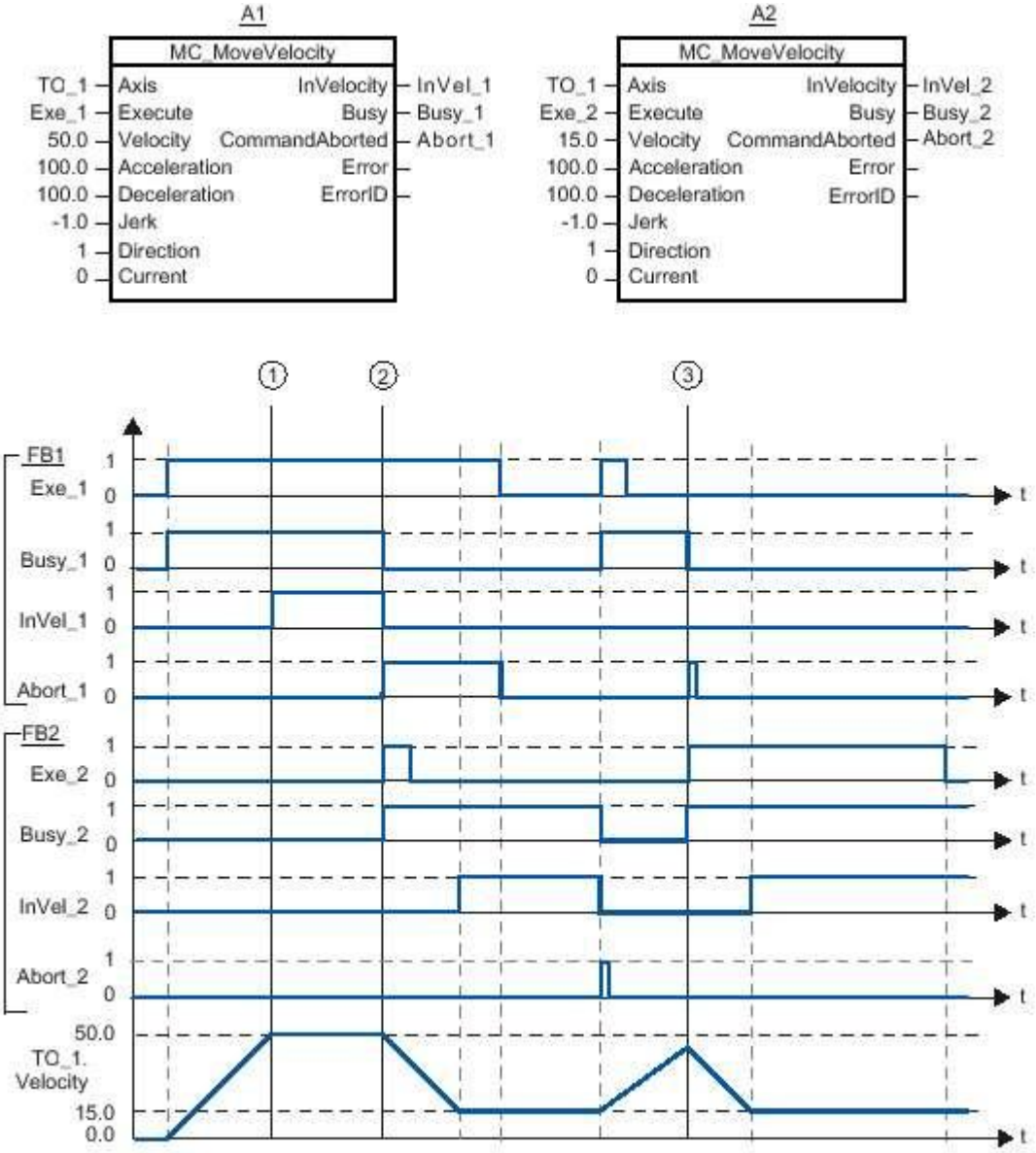
速度/速さが、一定の移動中にオーバーライド(<TO>.Override.Velocity)の変更によって影響を受けると、「InVelocity」パラメータが加速または減速中にリセットされます。 新しく計算された速度/速さに達すると(「Velocity」 ×「Override」 %)、「InVelocity」がリセットされます。

追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_MoveVelocity: ファンクションチャート V1

ファンションチャート: 速度の指定による軸の移動とオーバーライドジョブに対する応答



「Exe_1」によって開始された「MC_MoveVelocity」ジョブ(A1)は、軸を加速し、時刻①に、「InVel_1」によって、セットポイント速度 50.0 に達したことを通知します。

時刻②に、このタスクは、別の「MC_MoveVelocity」ジョブ(A2)によって無効にされます。 この中止は、「Abort_1」によって通知されます。 新しいセットポイント速度 15.0 に達すると、「InVel_2」によって通知されます。 この後、軸は一定の速度 15.0 で動作し続けます。

この実行中の「MC_MoveVelocity」ジョブ(A2)は、別の「MC_MoveVelocity」ジョブ(A1)によってオーバーライドされます。 この中止は、「Abort_2」によって通知されます。 軸は、新しいセットポイント速度 50.0 まで加速されます。 このセットポイント速度に達する前に、現在の「MC_MoveVelocity」ジョブ(A1)は、時刻③に、別の「MC_MoveVelocity」ジョブ(A2)によって無効にされます。 この中止

は、「Abort_1」によって通知されます。 新しいセットポイント速度 15.0 に達すると、「InVel_2」によって通知されます。 この後、軸は一定の速度 15.0 で動作し続けます。

MC_MoveRelative

この章には下記に関する情報が記載されています：

- MC_MoveRelative: 軸の相対位置決め V1 (S7-1500)
- MC_MoveRelative: ファンクションチャート V1 (S7-1500)

MC_MoveRelative: 軸の相対位置決め V1

説明

モーションコントロール命令「MC_MoveRelative」では、ジョブ処理の開始時の位置を基準にして、相対的に軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

適用対象

- 位置決め軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_MoveRelative」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

「MC_MoveRelative」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

パラメータ

次の表に、「MC_MoveRelative」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Positioning-Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Distance	INPUT	LREAL	0.0	位置決めプロセス用の距離 (逆転または正転)	
Velocity	INPUT	LREAL	-1.0	位置決め用のセットポイント速度 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された速度が使用されます。 (<TO>.DynamicDefaults.Velocity)	
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)	
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0:指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	ターゲット位置に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。

Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「Error-ID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

開始位置を基準にして相対的に軸を移動

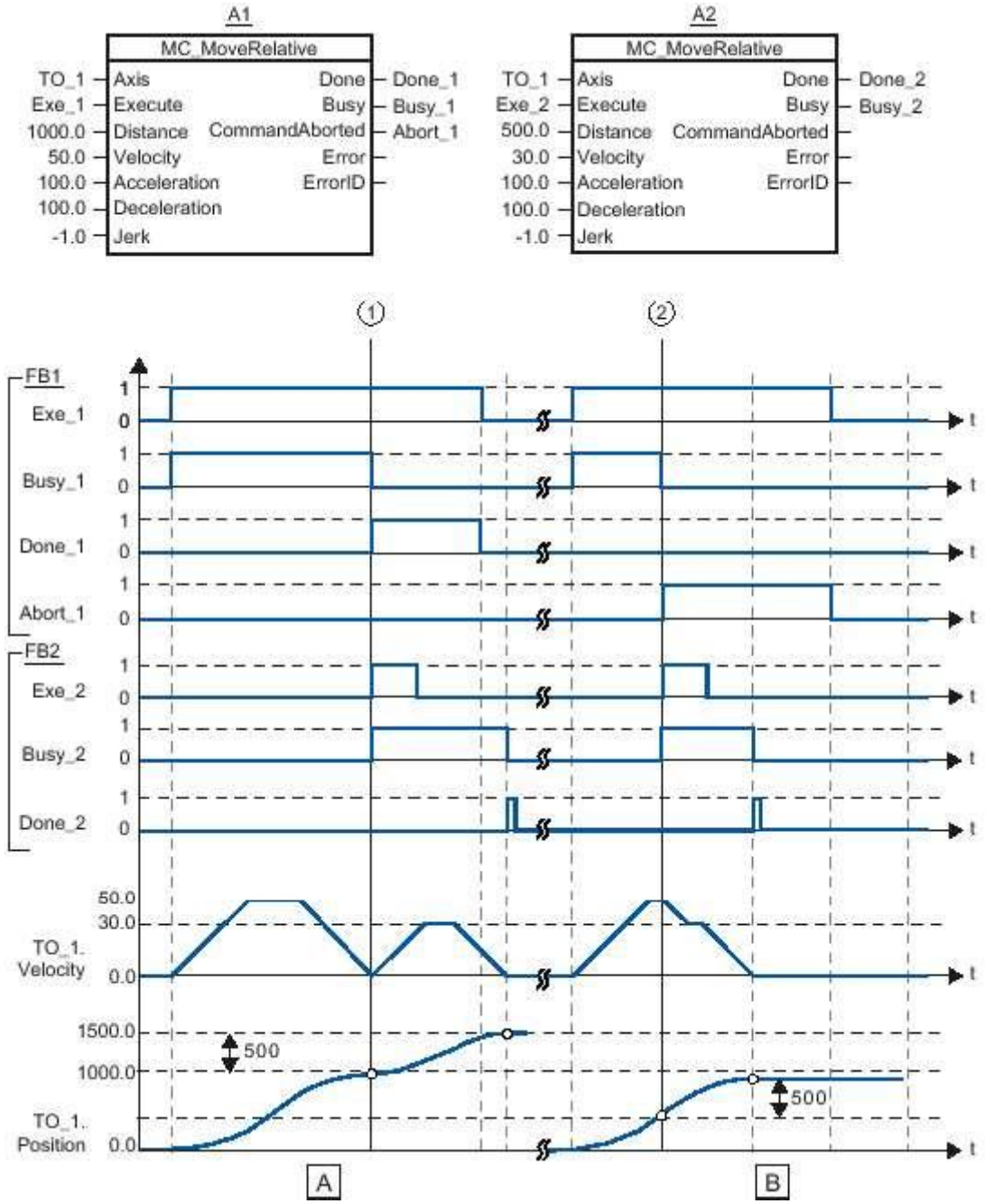
開始位置を基準にして相対的に軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 「Distance」パラメータで、移動する距離を指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveRelative」ジョブを開始します。

現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。

MC_MoveRelative: ファンクションチャート V1

ファンションチャート: 軸の相対位置決めと、オーバーライドジョブへの応答



セクション	軸は、「MC_MoveRelative」ジョブ(A1)によって、距離(「Distance」) 1000.0 だけ移動します (この場合、開始位置は 0.0 です)。 軸がターゲット位置に到達すると、このことが時刻① に「Done_1」によって通知されます。 この時刻①に、距離が 500.0 の別の「MC_MoveRelative」ジョブ(A2)が開始されます。 軸が新しいターゲット位置に到達すると、「Done_2」

A	によって通知されます。「Exe_2」が以前にリセットされたため、「Done_2」は 1 つのサイクルのみに適用されます。
セクション B	実行中の「MC_MoveRelative」ジョブ(A1)が、別の「MC_MoveRelative」ジョブ(A2)によってオーバーライドされます。 この中止は、時刻②に、「Abort_1」によって通知されます。 この後、軸は新しい速度で距離(「Distance」) 500.0 だけ移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_MoveAbsolute

この章には下記に関する情報が記載されています：

- MC_MoveAbsolute: 軸の絶対位置決め V1 (S7-1500)
- MC_MoveAbsolute: ファンクションチャート V1 (S7-1500)

MC_MoveAbsolute: 軸の絶対位置決め V1

説明

モーションコントロール命令「MC_MoveAbsolute」では、絶対位置まで軸を移動できます。

移動中の動的動作は、パラメータ「Velocity」、「Jerk」、「Acceleration」、および「Deceleration」で定義されます。

適用対象

- 位置決め軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。
- テクノロジーオブジェクトが原点復帰済みであること。

応答の無効化

「MC_MoveAbsolute」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

「MC_MoveAbsolute」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

パラメータ

次の表に、「MC_MoveAbsolute」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明
Axis	InOut	TO_Positioning-Axis	-	テクノロジーオブジェクト
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始
Position	INPUT	REAL	0.0	絶対ターゲット位置
Velocity	INPUT	LREAL	-1.0	位置決め用のセットポイント速度 値 > 0.0: 指定された値が使用されます。 値 = 0.0: 不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された速度が使用されます。 (<TO>.DynamicDefaults.Velocity)
Acceleration	INPUT	LREAL	-1.0	加速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された加速が使用されます。 (<TO>.DynamicDefaults.Acceleration)
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0: 固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0: 台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)
Direction	INPUT	INT	3	軸の回転方向 「モジュロ」が有効な場合のみ評価されます。 [テクノロジーオブジェクト|設定|基本パラメータ|モジュロの有効化] 1: 正転の回転方向 2: 逆転の回転方向

				3	最短パス
Done	OUTPUT	BOOL	FALSE	TRUE	ターゲット位置に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「Error-ID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

絶対位置まで軸を移動

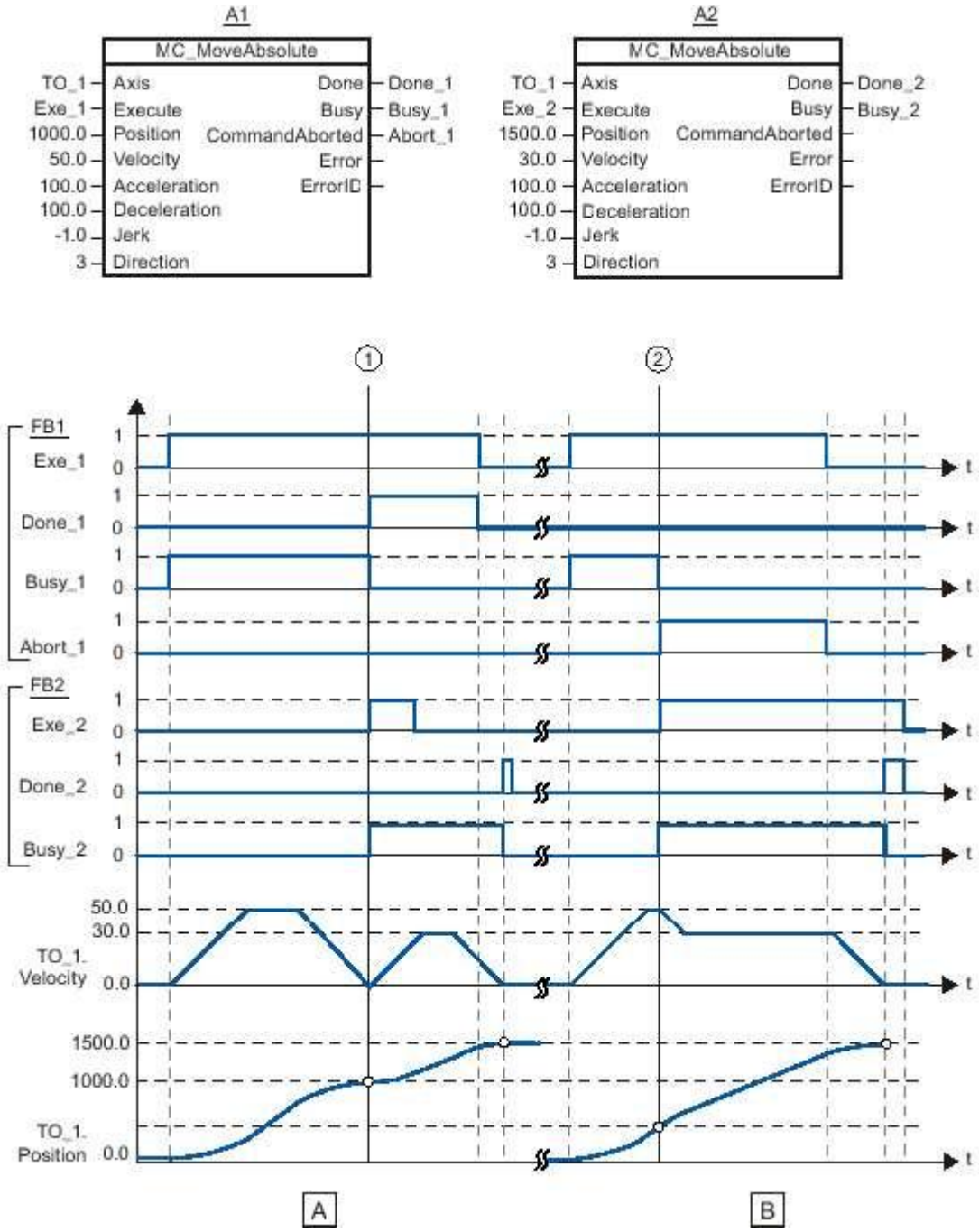
絶対位置まで軸を移動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. 「Position」パラメータで、必要なターゲット位置を指定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_MoveAbsolute」ジョブを開始します。

現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。

MC_MoveAbsolute: ファンクションチャート V1

ファンションチャート: 軸の絶対位置決めと、オーバーライドジョブへの応答



セクション	軸が、「MC_MoveAbsolute」ジョブ(A1)によって、絶対位置 1000.0 まで移動します。 軸がターゲット位置に到達すると、このことが時刻①に「Done_1」によって通知されます。 この時刻①に、ターゲット位置が 1500.0 の別の「MC_MoveAbsolute」ジョブ(A2)が開始されます。 軸がターゲット位置 1500.0 に到達すると、このことが「Done_2」によって通知

A	されます。 「Exe_2」が以前にリセットされたため、「Done_2」は 1 つのサイクルのみに適用されます。
セクション B	実行中の「MC_MoveAbsolute」ジョブ(A1)が、時刻②に、別の「MC_MoveAbsolute」ジョブ(A2)によって無効にされます。 この中止は、「Abort_1」によって通知されます。 軸は変更された速度まで制動し、新しいターゲット位置 1500.0 まで移動します。 新しいターゲット位置に到達すると、「Done_2」によって通知されます。

MC_Halt

この章には下記に関する情報が記載されています：

- MC_Halt: 軸の停止 V1 (S7-1500)
- MC_Halt: ファンクションチャート V1 (S7-1500)

MC_Halt: 軸の停止 V1

説明

モーションコントロール命令「MC_Halt」では、軸を停止状態まで制動することができます。

制動プロセス中の動的な動作は、パラメータ「Jerk」および「Deceleration」で定義されます。

適用対象

- 位置決め軸
- 速度軸

必要条件

- テクノロジーオブジェクトが正しく設定済みであること。
- このテクノロジーオブジェクトが有効であること。

応答の無効化

「MC_Halt」ジョブは以下によって中止されます。

- 「MC_Power.Enable」 = FALSE による軸の無効化
- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

「MC_Halt」ジョブを開始すると、以下の動作中のモーションコントロールジョブが中止されます。

- "MC_Home」ジョブ 「HomingMode」 = 4、5
- "MC_Halt」ジョブ
- "MC_MoveAbsolute」ジョブ
- "MC_MoveRelative」ジョブ
- "MC_MoveVelocity」ジョブ
- "MC_MoveJog」ジョブ

パラメータ

次の表に、「MC_Halt」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Speed-Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Deceleration	INPUT	LREAL	-1.0	減速 値 > 0.0: 指定された値が使用されます。 値 = 0.0:不可 値 < 0.0:[テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定された減速が使用されます。 (<TO>.DynamicDefaults.Deceleration)	
Jerk	INPUT	LREAL	-1.0	ジャーク 値 > 0.0:固定加速度プロファイル; 指定されたジャークが使用されます 値 = 0.0:台形速度プロファイル 値 < 0.0: [テクノロジーオブジェクト|設定|拡張パラメータ|動的な既定]で設定されたジャークが使用されます。 (<TO>.DynamicDefaults.Jerk)	
Done	OUTPUT	BOOL	FALSE	TRUE	速度 0 に達しました。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
CommandAborted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

「MC_Halt」による軸の制動

軸を停止状態まで減速するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Deceleration」および「Jerk」に必要な値を設定します。
3. パラメータ「Execute」の立ち上がりエッジで、「MC_Halt」ジョブを開始します。

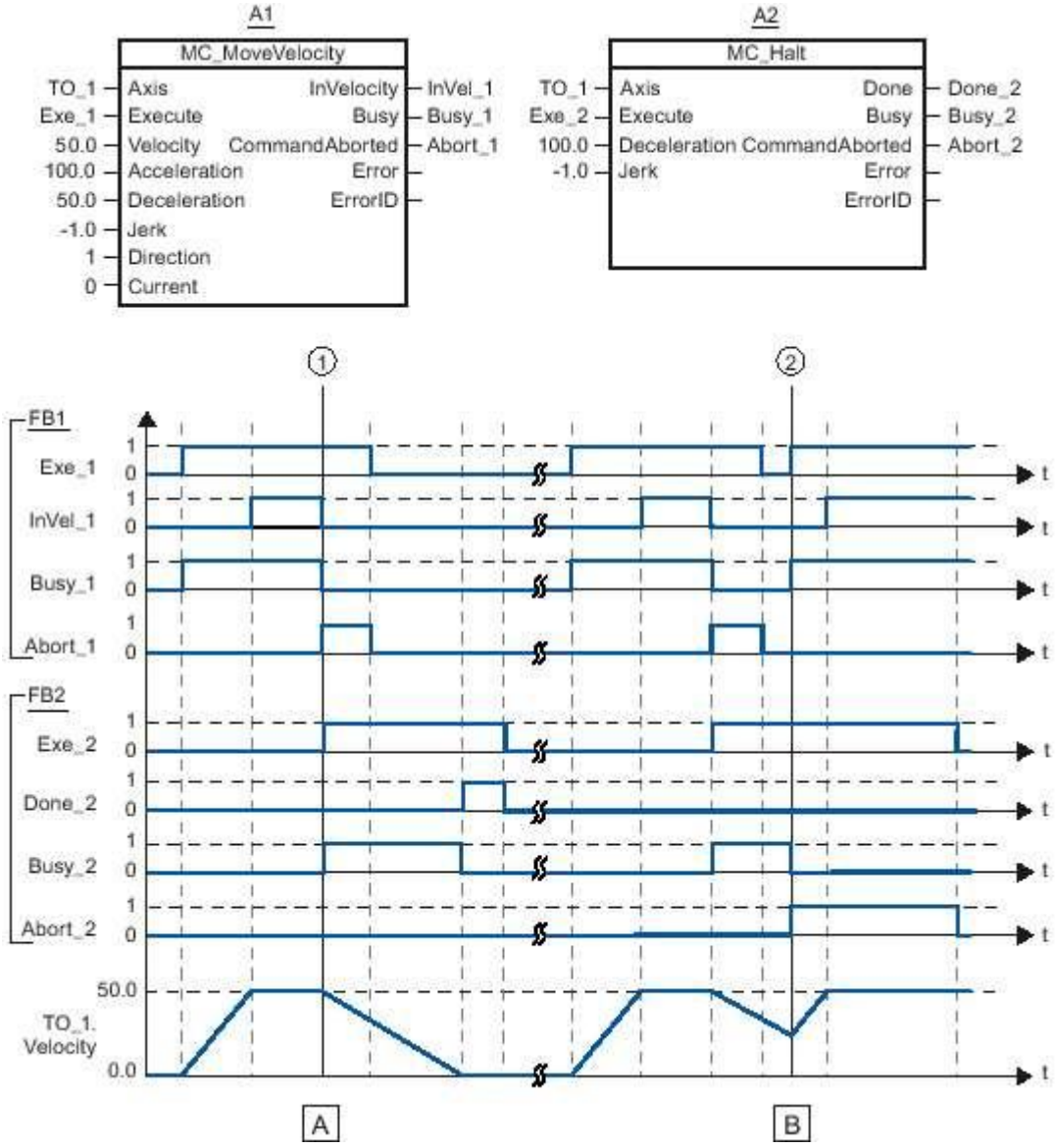
現在のモーション状態は、「Busy」、「Done」、および「Error」に示されます。軸の停止は、[テクノロジーオブジェクト|診断|ステータスおよびエラービット|モーションステータス|停止] (<TO>.StatusWord.X7 (Standstill))で示されます。

追加情報

個々のステータスビットの評価用オプションについては、「StatusWord、ErrorWord、および Warning-Word の評価」セクションを参照してください。

MC_Halt: ファンクションチャート V1

ファンションチャート: 軸の停止とオーバーライドジョブに対する応答



セクション A	軸は、「MC_MoveVelocity」ジョブ(A1)によって移動します。 セットポイント速度 50.0 に達すると、「InVel_1」によって通知されます。 時刻①に、この「MC_MoveVelocity」が、「MC_Halt」ジョブ(A2)によって無効にされます。 このジョブ終了は、「Abort_1」によって通知されます。 軸は停止状態まで制動します。 「MC_Halt」ジョブの正常終了が、「Done_2」によって通知されます。
セクション	軸は、「MC_MoveVelocity」ジョブ(A1)によって移動します。 セットポイント速度 50.0 に達すると、「InVel_1」によって通知されます。 次に、この「MC_MoveVelocity」が、「MC_Halt」ジョブ(A2)によって無効にされます。 このジョブ終了は、「Abort_1」によって通知されます。 制動プロセス中の時刻②に、「MC_Halt」が、「MC_MoveVelocity」ジョブ

B	(A1)によって無効にされます。 このジョブ終了は、「Abort_2」によって通知されます。 セットポイント速度 50.0 に達すると、「InVel_1」によって通知されます。 この後、軸は一定速度で移動します。

MC_Reset

この章には下記に関する情報が記載されています：

- MC_Reset: アラームの確認、テクノロジーオブジェクト V1 の再起動 (S7-1500)

MC_Reset: アラームの確認、テクノロジーオブジェクト V1 の再起動

説明

ユーザープログラムで確認応答できるすべてのテクノロジーアラームは、モーションコントロール命令「MC_Reset」で確認応答されます。 確認応答を行うと、テクノロジーデータブロックの「Error」および「Warning」ビットもリセットされます。

テクノロジーオブジェクトは、「Restart」 = TRUE のモーションコントロール命令「MC_Reset」を使用して再初期化(再起動)されます。 テクノロジーオブジェクトの再起動時、新しい設定データがテクノロジーデータブロックに適用されます。

適用対象

- 位置決め軸
- 速度軸
- 外部エンコーダ

必要条件

- 保留中のテクノロジーアラームのエラーの原因が解決済みであること。
- 再起動するには、テクノロジーオブジェクトを無効にする必要があります。
  (「MC_Power.Status」 = FALSE および「MC_Power.Busy」 = FALSE)

応答の無効化

- パラメータ「Restart」 = FALSE:

  命令「MC_Reset」の処理は、他のモーションコントロールジョブが中止できます。 この MC_Reset ジョブは、すべての実行中のモーションコントロールジョブを中止しません。

- パラメータ「Restart」 = TRUE:

  パラメータ「Restart」 = TRUE の命令「MC_Reset」の処理は、他のすべてのモーションコントロールジョブが中止できません。

パラメータ

次の表に、「MC_Reset」モーションコントロール命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定値	説明	
Axis	InOut	TO_Axis	-	テクノロジーオブジェクト	
Execute	INPUT	BOOL	FALSE	立ち上がりエッジでのジョブの開始	
Restart	INPUT	BOOL	FALSE	TRUE	"Restart" テクノロージーオブジェクトの再初期化と、保留中のテクノロジーアラームの確認応答。 テクノロジーオブジェクトは、設定されたスタート値で、再初期化されます。

				FALSE	キューに登録されたテクノロジーアラームの確認応答。
Done	OUTPUT	BOOL	FALSE	TRUE	エラーが確認応答済みです。 再起動が実行済みです。
Busy	OUTPUT	BOOL	FALSE	TRUE	ジョブが実行中です。
Com-mandA-borted	OUTPUT	BOOL	FALSE	TRUE	実行中に、ジョブが別のジョブによって中止されました。
Error	OUTPUT	BOOL	FALSE	TRUE	ジョブの実行中にエラーが発生しました。 このジョブは拒否されます。 このエラーの原因は、「ErrorID」パラメータで検出できます。
ErrorID	OUTPUT	WORD	16#0000	パラメータ「Error」のエラー ID	

テクノロジーアラームへの確認応答

テクノロジーアラームに確認応答するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Restart」 = FALSE を設定します。
3. パラメータ「Execute」での立ち上がりエッジで、エラーの確認応答を開始します。

Done パラメータが値 TRUE を示す場合、エラーに確認応答済みです。

注記

「Restart」 = FALSE での確認応答

テクノロジアラームのみを確認する場合、「Restart」 = FALSE を設定します。 再起動中は、テクノロジーオブジェクトを使用できません。

テクノロジーオブジェクトの再起動

テクノロジーオブジェクトを再起動するには、以下の手順に従います。

1. 上記の必要条件をチェックします。
2. パラメータ「Restart」 = TRUE を設定します。
3. 入力パラメータ「Execute」の立ち上がりエッジで、再起動を実行します。

この「Done」パラメータが値 TRUE を示す場合は、このテクノロジーオブジェクトの再起動が完了しています。

再起動に関する追加情報は、「テクノロジーオブジェクトの再起動」セクションを参照してください。

高速カウンタ

この章には下記に関する情報が記載されています：

- CTRL_HSC: 高速カウンタの制御 (S7-1200)
- CTRL_HSC_EXT: 高速カウンタの制御(拡張) (S7-1200)

CTRL_HSC: 高速カウンタの制御

パラメータ

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	INPUT	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	OUTPUT	BOOL	I、Q、M、D、L	許可出力
HSC	INPUT	HW_HSC	I、Q、M または定数	高速カウンタのハードウェアアドレス(HW ID)
DIR	INPUT	BOOL	I、Q、M、D、L、または定数	新しいカウント方向を有効にします(NEW_DIR を参照)
CV	INPUT	BOOL	I、Q、M、D、L、または定数	新しいカウンタ値を有効にします(NEW_CV を参照)
RV	INPUT	BOOL	I、Q、M、D、L、または定数	新しい基準値を有効にします(NEW_RV を参照)
PERIOD	INPUT	BOOL	I、Q、M、D、L、または定数	新しい周波数測定時間を有効にします(NEW_PERIOD を参照)
NEW_DIR	INPUT	INT	I、Q、M、D、L、または定数	DIR = TRUE の場合、カウント方向が読み込まれます。
NEW_CV	INPUT	DINT	I、Q、M、D、L、または定数	CV = TRUE の場合、カウンタ値が読み込まれます。
NEW_RV	INPUT	DINT	I、Q、M、D、L、または定数	RV = TRUE のときに読み込まれた基準値
NEW_PERIOD	INPUT	INT	I、Q、M、D、L、または定数	PERIOD = TRUE の場合、周波数測定時間が読み込まれます。
BUSY	OUTPUT	BOOL	I、Q、M、D、L	処理ステータス
STATUS	OUTPUT	WORD	I、Q、M、D、L	演算のステータス

説明

「高速カウンタの制御」命令を使用すると、パラメータ設定を行い、CPU でサポートされている高速カウンタを制御することができます。これは、新しい値をカウンタにロードすることによって行われます。 命令を実行するには、制御される高速カウンタが有効である必要があります。 高速カウンタによっては、プログラム内で複数の「高速カウンタの制御」命令を同時に実行することはできません。

[高速カウンタ制御]演算を使用して、以下のパラメータ値を高速カウンタに読み込むことができます。

- カウント方向(NEW_DIR): このカウント方向では、高速カウンタのカウントをアップさせるかダウンさせるかを定義します。 カウント方向を定義するには、NEW_DIR 入力で次の値を使用します。 1 = アップ、-1 = ダウン。

  「高速カウンタの制御」命令を使用してカウント方向を変えられるのは、プログラムのパラメータで方向コントロールが設定されている場合のみです。 DIR 入力のビットがセットされている場合、NEW_DIR 入力で指定されたカウント方向が高速カウンタに読み込まれます。
- カウンタ値 (NEW_CV): このカウンタ値は、高速カウンタがカウントを開始する際の初期値です。 カウンタ値の範囲は、-2147483648～2147483647 です。

  CV 入力のビットがセットされている場合、NEW_CV 入力で指定されたカウンタ値が高速カウンタに読み込まれます。
- 基準値(NEW_RV): この基準値と現在のカウンタ値を比較してアラームをトリガすることができます。 カウンタ値と同様、基準値の範囲は-2147483648～2147483647 です。

  RV 入力のビットがセットされている場合、NEW_RV 入力で指定された基準値が高速カウンタに読み込まれます。
- 周波数測定時間(NEW_PERIOD): 周波数測定時間を指定するには、NEW_PERIOD 入力で次の値を使用します。 10 = 0.01s、100 = 0.1s、1000 = 1s。

  指定された高速カウンタに「周波数測定」ファンクションが設定されている場合、時間を更新することが可能です。 PERIOD 入力のビットがセットされている場合、NEW_PERIOD 入力で指定された時間が高速カウンタに読み込まれます。

「高速カウンタの制御」命令を実行できるのは、EN 入力のシグナル状態が「1」の場合のみです。 演算が実行されている間は、BUSY 出力のビットがセットされます。 演算の実行が完了すると、BUSY 出力のビットがリセットされます。

EN 許可入力のシグナル状態が「1」で、かつ演算の実行時にエラーが発生しない場合にのみ、ENO 許可出力が設定されます。

「高速カウンタの制御」命令を挿入すると、演算データを保存するインスタンスデータブロックが作成されます。

STATUS パラメータ

STATUS 出力では、「高速カウンタの制御」命令の実行時にエラーが発生したかどうかを照会できます。 次の表に、STATUS 出力で出力される値の意味を示します。

エラーコード(16 進数)	説明
0	エラーは発生していません。
80A1	高速カウンタのハードウェア識別子が無効です。
80B1DB	カウント方向(NEW_DIR)が無効です。
80B2	カウンタ値(NEW_CV)が無効です。
80B3	基準値(NEW_RV)が無効です。
80B4	周波数測定時間(NEW_PERIOD)が無効です。
80C0	高速カウンタへの複数アクセス
80D0	高速カウンタ(HSC)が CPU ハードウェアコンフィグレーションで有効になっていません。

CTRL_HSC_EXT: 高速カウンタの制御(拡張)

パラメータ

パラメータ	宣言	データタイプ	メモリ領域	説明
EN	INPUT	BOOL	I、Q、M、D、L、T、C	許可入力
ENO	OUTPUT	BOOL	I、Q、M、D、L	許可出力
HSC	INPUT	HW_HSC	I、Q、M または定数	高速カウンタのハードウェアアドレス(HW ID)
CTRL	INOUT	VARIANT	M、D	システムデータタイプ(SDT)の使用
DONE	OUTPUT	BOOL	I、Q、M、D、L	命令処理正常終了後のフィードバック
BUSY	OUTPUT	BOOL	I、Q、M、D、L	処理ステータス
ERROR	OUTPUT	BOOL	I、Q、M、D、L	命令処理異常のフィードバック
STATUS	OUTPUT	WORD	I、Q、M、D、L	演算のステータス

説明

「高速カウンタの制御(拡張)」命令を使用すると、CPU でサポートされている高速カウンタを制御し、パラメータを設定することができます。これはソフトウェアで、新しい値をカウンタにロードすることによって行われます。 命令を実行するには、制御される高速カウンタが有効である必要があります。 高速カウンタによっては、プログラム内で複数の「高速カウンタの制御(拡張)」命令を同時に実行することはできません。

「高速カウンタの制御(拡張)」命令を実行できるのは、EN 入力のシグナル状態が「1」の場合のみです。演算が実行されている間は、BUSY 出力のビットがセットされます。演算の実行が完了すると、BUSY 出力のビットがリセットされます。

EN 許可入力のシグナル状態が「1」で、かつ演算の実行時にエラーが発生しない場合にのみ、ENO 許可出力が設定されます。

「高速カウンタの制御(拡張)」命令を挿入すると、演算データを保存するインスタンスデータブロックが作成されます。

システムデータタイプ HSC_Period の使用

「高速カウンタの制御(拡張)」命令は、周期測定に関するシステムデータタイプ SDT 381「HSC_Period」をサポートしています。

Byte (バイト)	パラメータ	データタイプ	説明
0 ... 3	Elapsed-Time	UDINT	Edge_Count の立ち上がりエッジ間の時間
4 ... 7	EdgeCount	UDINT	Elapsed_Time 内の立ち上がりエッジの数。 Edge_Count = 0 の場合、Elapsed_Time が最後の立ち上がりエッジ後の時間です。
8.0	EnHSC	BOOL	ゲート制御による許可入力として使用します。 • FALSE: 測定停止 • TRUE: 測定有効
8.6	EnPeriod	BOOL	周期の更新 • FALSE: 更新なし • TRUE: 更新周期
10 ... 11	NewPeriod	INT	周期測定の間隔(ミリ秒単位)。 許可値は 10、100 および 1000 です。

STATUS パラメータ

STATUS 出力では、「高速カウンタの制御(拡張)」命令の実行時にエラーが発生したかどうかを照会できます。次の表に、STATUS 出力で出力される値の意味を示します。

エラーコード(16 進数)	説明
0	エラーは発生していません。
80A1	高速カウンタのハードウェア識別子が無効です。
80C0	高速カウンタへの複数アクセス
80D0	高速カウンタ(HSC)が CPU ハードウェアコンフィグレーションで有効になっていません。

カウントと測定

この章には下記に関する情報が記載されています：

- High_Speed_Counter (S7-1500)

High_Speed_Counter

この章には下記に関する情報が記載されています：

- High_Speed_Counter (S7-1500)

High_Speed_Counter

この章には下記に関する情報が記載されています：

- High_Speed_Counter の説明 (S7-1500)
- High_Speed_Counter 入力パラメータ (S7-1500)
- High_Speed_Counter 出力パラメータ (S7-1500)
- ErrorID パラメータ (S7-1500)
- High_Speed_Counter 静的変数 (S7-1500)

High_Speed_Counter の説明

説明

High_Speed_Counter 命令は、ユーザープログラムによるテクノロジーモジュールのカウントおよび測定ファンクションを制御するために使用します。

呼び出し

命令 High_Speed_Counter は、周期的な場合でも時間制御されたプログラムの場合でも、カウンタごとに 1 回ずつ呼び出す必要があります。 この呼び出しは、イベント制御割り込みプログラムでは許可されていません。

ファンクションの説明

カウンタ値: カウンタ値は出力パラメータ CountValue で表示されます。 カウンタ値は、High_Speed_Counter 命令が呼び出されるたびに更新されます。

測定値: 測定値は出力パラメータ MeasuredValue で表示されます。 測定値は、High_Speed_Counter 命令が呼び出されるたびに更新されます。

測定値とカウンタ値は、フィードバックインターフェースで同時に表示されます。

Capture: 出力パラメータ CaptureStatus = TRUE は、出力パラメータ CapturedValue の値 Capture 値が有効であることを示します。

- Capture 値は、以下の条件で取り込まれます。
  - デジタル入力に、パラメータ割り当て「Capture」がある。
  - CaptureEnable = TRUE
  - デジタル入力のエッジで、Capture ファンクションを使用。
- 出力パラメータ CaptureStatus は入力パラメータ CaptureEnable の立ち下がりエッジでリセットされます。

同期: 出力パラメータ SyncStatus = TRUE は、同期が行われていることを示します。

- カウンタ値は、以下の条件で同期されます。
  - デジタル入力にパラメータ割り当て「同期」があるか、 **または** インクリメンタルエンコーダにパラメータ割り当て「信号 N での同期」がある。
  - SyncEnable = TRUE
  - SyncUpDirection (または SyncDownDirection) = TRUE
  - 同期ファンクションによるデジタル入力のエッジ、**または**エンコーダ入力の信号 N の立ち上がりエッジ
- 出力パラメータ SyncStatus は、立ち下がりエッジで以下でリセットされます。
  - 入力パラメータ SyncEnable または
  - 静的タグ SyncDownDirection または
  - 静的タグ SyncUpDirection

ユーザープログラムによるパラメータ変更

ユーザープログラムを使用してパラメータを変更するには、以下の手順に従います。

1. 関連する Set タグをチェックして、テクノロジーオブジェクトがパラメータ変更の準備が完了しているか(Set タグ = FALSE)、変更コマンドがまだ実行中であるか(Set タグ = TRUE)を判定します。

2. テクノロジーオブジェクトがパラメータ変更の準備が完了している場合、関連する静的タグを変更します。
3. 対応する Set タグを変更コマンドの実行用に設定します。
4. 出力パラメータ Error を使用して、エラーが発生しているかどうかをチェックします。
エラーが発生しておらず、Set タグがテクノロジーオブジェクトによって自動的にリセットされている場合、パラメータ変更は正常に行われました。

カウンタ上限値または下限値: 2 つの限界値は、ユーザープログラムで以下のように変更できます。

1. 静的タグ SetUpperLimit (または SetLowerLimit) = FALSE をチェックします。
2. 静的タグ NewUpperLimit (または NewLowerLimit)を変更します。
3. SetUpperLimit (または SetLowerLimit) = TRUE
4. 静的タグ CurUpperLimit または CurLowerLimit の新しいカウンタ限界値をチェックします。

注記
新しいカウント上限値が現在のカウント値より小さい場合、カウント値は設定に従ってカウント下限値または開始値に設定されます。 新しいカウント下限値が現在のカウント値より大きい場合、カウント値は設定に従ってカウント上限値または開始値に設定されます。

比較値: 2 つの比較値は、ユーザープログラムで以下のように変更できます。

1. 静的タグ SetReferenceValue0 (または ReferenceValue1) = FALSE をチェックします。
2. 静的タグ NewReferenceValue0 (または NewReferenceValue1)を変更します。
3. SetReferenceValue0 (または SetReferenceValue1) = TRUE
4. 静的タグ CurReferenceValue0 (または CurReferenceValue1)の新しい比較値のチェック

カウンタ値: 現在のカウンタ値をユーザープログラムから新しいカウンタ値に設定することができます。 現在のカウンタ値は次のように計算されます。

1. 入力パラメータ SetCountValue = FALSE をチェックします。
2. 静的タグ NewCountValue を変更します。
3. SetCountValue = TRUE
4. 出力パラメータ CountValue の新しいカウンタ値をチェックします。

開始値: 開始値は、ユーザープログラムを使用して以下のように変更できます。

1. 静的タグ SetStartValue (または) = FALSE をチェックします。
2. 静的タグ NewStartValue を変更します。
3. SetStartValue = TRUE
4. 静的タグ CurStartValue の新しい開始値をチェックします。

イベントの確認

通知されたイベントの確認は、入力パラメータ EventAck の立ち上がりエッジで行うことができます。EventAck は、テクノロジーオブジェクトがカウントチャンネルの以下のイベントのステータスビットをリセットするまで、設定状態であることが必要です。

- CompResult0
- CompResult1
- ZeroStatus
- PosOverflow

- NegOverflow

デジタル入力のステータス

デジタル入力のステータスは、静的タグ StatusDI0、StatusDI1、または StatusDI2 で入手できます。

ユーザープログラムによるデジタル出力の使用

デジタル出力を High_Speed_Counter 命令で設定できます。

- 設定[ユーザープログラムによる使用]が[出力のセット]に対して設定されている場合。
- 設定[CPU から比較値までのコマンドのセット後]が[出力のセット]に対して設定されている場合。
- 対応する静的タグ ManualCtrlDQm (一時的な上書き)を設定している場合。

静的タグ SetDQ0 および SetDQ1 は、上記の場合にのみ有効です。 1 番目と 3 番目のケースでは、DQm は SetDQm の値に従います。 2 番目のケースでは、DQm は SetDQm によりエッジ(立ち上がりまたは立ち下がり)で設定されます。DQm は、カウンタ値が比較値と等しい場合、リセットされます。

エラーに対する応答

命令の呼び出し中またはテクノロジーモジュールでエラーが発生した場合、出力パラメータ Error がセットされます。 詳細なエラー情報は、出力パラメータ ErrorID に表示されます。

エラーの原因を取り除き、入力パラメータ ErrorAck のセットによってエラーメッセージの確認応答をします。 保留中のエラーがしれ以上存在しない場合は、テクノロジーオブジェクトが出力パラメータ、Error をリセットします。 直前のエラーを確認するまで、新しいエラーは通知されません。

カウント方向の変更

カウント方向は、「パルス(A)」が信号タイプとして設定されている場合、ユーザープログラムによってのみ変更できます。 それ以外の場合では、カウンタ方向はテクノロジーモジュールの入力信号によってすべて決定されます。 カウント方向は静的タグ NewDirection で制御されます。

- +1: 上向きのカウント方向
- -1: 下向きのカウント方向

変更コマンドを実行するには、タグ SetNewDirection = TRUE をセットする必要があります。

High_Speed_Counter 入力パラメータ

パラメータ	宣言	データタイプ	既定	説明
SwGate	INPUT	BOOL	FALSE	ソフトウェアゲートの制御: • 信号立ち上がり: ソフトウェアゲートが開きます • 信号立ち下がり: ソフトウェアゲートが閉じます ハードウェアゲートと共に、SwGate が内部ゲートを有効にします。
SetCount-Value	INOUT	BOOL	FALSE	立ち上がりエッジにより、静的タグ NewCountValue の新しいカウンタ値のテクノロジーモジュールへの伝送が開始されます。 カウンタ値は、伝送が完了すると同時に有効になります。
CaptureEn-able	INPUT	BOOL	FALSE	Capture ファンクションを有効にする 有効にすると、関連するデジタル入力の次の設定済みのエッジで Capture イベントが成立します。 CaptureEnable の立ち下がりエッジは、出力パラメータ CaptureStatus をリセットします。 CaptureEnable の立ち下がりエッジは、Capture イベントが成立していない場合でも有効化をリセットします。
SyncEnable	INPUT	BOOL	FALSE	同期を有効にする 同期に対して有効になった方向は、静的タグ SyncUpDirection および SyncDownDirection で示されます。 SyncEnable の立ち下がりエッジは、出力パラメータ SyncStatus をリセットします。
ErrorAck	INPUT	BOOL	FALSE	立ち上がりエッジで、報告されたエラーステータスが確認されます。
EventAck	INPUT	BOOL	FALSE	立ち上がりエッジは、以下の出力パラメータをリセットします。 • CompResult0 • CompResult1 • ZeroStatus • PosOverflow • NegOverflow

High_Speed_Counter 出力パラメータ

パラメータ	宣言	データタイプ	既定	説明
StatusHW	OUTPUT	BOOL	FALSE	テクノロジーモジュール:のステータスビット: モジュールが設定済みで、操作準備ができています。 モジュールデータが有効です。
StatusGate	OUTPUT	BOOL	FALSE	ステータスビット: パラメータがセットされている場合、内部ゲートがリリースされます。
StatusUp	OUTPUT	BOOL	FALSE	ステータスビット: 最後の出力パルスがカウンタを繰り上げ、発生から 0.5 s 経過していません。
Status-Down	OUTPUT	BOOL	FALSE	ステータスビット: 最後の出力パルスがカウンタを繰り下げ、発生から 0.5 s 経過していません。
CompRe-sult0	OUTPUT	BOOL	FALSE	ステータスビット: DQ0 の比較イベントが成立しました。 入力パラメータ EventAck の立ち上がりエッジを使用して CompResult0 をリセットします。
CompRe-sult1	OUTPUT	BOOL	FALSE	ステータスビット: DQ1 の比較イベントが成立しました。 入力パラメータ EventAck の立ち上がりエッジを使用して CompResult1 をリセットします。
SyncStatus	OUTPUT	BOOL	FALSE	ステータスビット: 同期が行われました 入力パラメータ SyncEnable がセットされている場合、設定済みのエッジによって個々のデジタル入力でステータスビット SyncStatus がセットされます。 SyncStatus は以下の立ち下がりエッジでリセットされます。 • SyncEnable (入力パラメータ)または • SyncUpDirection (静的タグ)または • SyncDownDirection (静的タグ)
CaptureS-tatus	OUTPUT	BOOL	FALSE	ステータスビット: Capture イベントが成立しました 入力パラメータ CaptureEnable がセットされている場合、設定済みのエッジによって個々のデジタル入力でステータスビット CaptureStatus.がセットされます。 入力パラメータ CaptureEnable の立ち下がりエッジで CaptureStatus がリセットされます。
ZeroStatus	OUTPUT	BOOL	FALSE	ステータスビット: CountValue が「0」値に達しました。 入力パラメータ EventAck の立ち上がりエッジを使用して ZeroStatus をリセットします。
PosOver-flow	OUTPUT	BOOL	FALSE	ステータスビット: CountValue がカウント上限値をオーバーしました。

				入力パラメータ EventAck の立ち上がりエッジを使用して PosOverflow をリセットします。
NegOver-flow	OUTPUT	BOOL	FALSE	ステータスビット: CountValue がカウント下限値をオーバーしました。 入力パラメータ EventAck の立ち上がりエッジを使用して NegOverflow をリセットします。
Error	OUTPUT	BOOL	FALSE	エラーが発生しました。 エラーの原因は、出力パラメータ ErrorID を参照してください。
ErrorID	OUTPUT	WORD	0	ErrorID パラメータに、エラーメッセージの番号が表示されます。 ErrorID = 0000_H: 保留エラーなし
CountValue	OUTPUT	DINT	0	カウンタ現在値
Captured-Value	OUTPUT	DINT	0	最後の取り込まれた Capture 値; CaptureStatus = TRUE、 新しい Capture イベントが成立した場合。
Measured-Value	OUTPUT	REAL	0.0	周波数、持続時間、または速度の現在の測定値(設定による)

ErrorID パラメータ

エラーコード (W#16#...)	説明
0000	エラーは発生していません。
テクノロジーモジュールからのエラーメッセージ	
80A1	フィードバックインターフェースからの POWER_ERROR: 不正な電源電圧 L+
80A2	フィードバックインターフェースからの ENC_ERROR : 不正なエンコーダ信号
80A3	フィードバックインターフェースからの LD_ERROR: 制御インターフェースを介したローディング時のエラー
命令のエラーメッセージ High_Speed_Counter	
80B1	無効なカウント方向
80B4	新しいカウント下限値が以下の条件を満たしていません。 • カウント下限値 < カウント上限値 • カウント下限値 <= 比較値/カウンタ値/開始値
80B5	新しいカウント上限値が以下の条件を満たしていません。 • カウント下限値 < カウント上限値 • カウント上限値 >= 比較値/カウンタ値/開始値
80B6	新しい開始値が以下の条件を満たしていません。 • カウント下限値 <= 開始値 <= カウント上限値
80B7	新しいカウンタ値が以下の条件を満たしていません。 • カウント下限値 <= カウンタ値 <= カウント上限値
80B8	新しい比較値 0 が以下の条件を満たしていません。 • カウント下限値 <= 比較値 0 <= カウント上限値 • 比較値 0 < 比較値 1
80B9	新しい比較値 1 が以下の条件を満たしていません。 • カウント下限値 <= 比較値 1 <= カウント上限値 • 比較値 0 < 比較値 1
80C0	命令 High_Speed_Counter が同じインスタンス(DB)で複数回呼び出されました。
80C1	テクノロジーモジュールとの通信が失敗しました(データレコードの読み出し): RDREC のエラー情報が静的タグ AdditionalErrorID に保存されました。
80C2	テクノロジーモジュールとの通信が失敗しました(データレコードの書き込み): WRREC のエラー情報が静的タグ AdditionalErrorID に保存されました。
80C3	入力データへのアクセス(フィードバックインターフェース)が失敗しました:GETIO_PART のエラー情報が静的タグ AdditionalErrorID に保存されました。
80C4	出力データへのアクセス(制御インターフェース)が失敗しました:SETIO_PART のエラー情報が静的タグ AdditionalErrorID に保存されました。
80C5	OB の現在の開始情報の読み取りが失敗しました:RD_SINFO のエラー情報が静的タグ AdditionalErrorID に保存されました。

80C6	テクノロジーモジュール:の I/O アドレスの取得に失敗しました:RD_ADDR のエラー情報が静的タグ AdditionalErrorID に保存されました。

High_Speed_Counter 静的変数

タグ	データタイプ	既定	アクセス	説明
NewCountValue	DINT	L#0	書き込み	新しいカウンタ値
NewReferenceVal-ue0	DINT	L#0	書き込み	新しい比較値 0
NewReferenceVal-ue1	DINT	L#10	書き込み	新しい比較値 1
NewUpperLimit	DINT	L#2147483647	書き込み	新しいカウント上限値
NewLowerLimit	DINT	L#-2147483648	書き込み	新しいカウント下限値
NewStartValue	DINT	L#0	書き込み	新しい開始値
CurReferenceValue0	DINT	L#0	読み出し	現在の比較値 0
CurReferenceValue1	DINT	L#10	読み出し	現在の比較値 1
CurUpperLimit	DINT	L#2147483647	読み出し	現在のカウント上限値
CurLowerLimit	DINT	L#-2147483648	読み出し	現在のカウント下限値
CurStartValue	DINT	L#0	読み出し	現在の開始値
NewDirection	INT	0	書き込み	新しいカウント方向: +1: 上向きのカウント方向 -1: 下向きのカウント方向
AdditionalErrorID	WORD	W#16#0000	読み出し	内部命令、たとえば RDREC のエラー情報
UserCmdFlags	STRUCT	-		
SetNewDirection	BOOL	FALSE	書き込み	新しいカウント方向の設定
SetUpperLimit	BOOL	FALSE	書き込み	カウント上限値の設定
SetLowerLimit	BOOL	FALSE	書き込み	カウント下限値の設定
SetReference-Value0	BOOL	FALSE	書き込み	比較値 0 の設定
SetReference-Value1	BOOL	FALSE	書き込み	比較値 1 の設定
SetStartValue	BOOL	FALSE	書き込み	開始値のセット
SyncDownDirec-tion	BOOL	TRUE	書き込み	下方向のカウントの同期を有効にします
SyncUpDirection	BOOL	TRUE	書き込み	上方向のカウントの同期を有効にします
SetDQ0	BOOL	FALSE	書き込み	デジタル出力 DQ0 の設定
SetDQ1	BOOL	FALSE	書き込み	デジタル出力 DQ1 の設定

	ManualCtrlDQ0	BOOL	FALSE	書き込み	デジタル出力 DQ0 の設定を有効にします TRUE: • SetDQ0 設定 DQ0 • 制御ビット TM_CTRL_DQ0 = FALSE FALSE: • 設定が有効ではありません • 制御ビット TM_CTRL_DQ0 = TRUE
	ManualCtrlDQ1	BOOL	FALSE	書き込み	デジタル出力 DQ1 の設定を有効にします TRUE: • SetDQ1 設定 DQ1 • 制御ビット TM_CTRL_DQ1 = FALSE FALSE: • 設定が有効ではありません • 制御ビット TM_CTRL_DQ1 = TRUE
UserStatusFlags		STRUCT	-		
	StatusDI0	BOOL	FALSE	読み出し	デジタル入力 DI0 の現在のステータス
	StatusDI1	BOOL	FALSE	読み出し	デジタル入力 DI1 の現在のステータス
	StatusDI2	BOOL	FALSE	読み出し	デジタル入力 DI2 の現在のステータス
	StatusDQ0	BOOL	FALSE	読み出し	デジタル出力 DQ0 の現在のステータス
	StatusDQ1	BOOL	FALSE	読み出し	デジタル出力 DQ1 の現在のステータス

PID 制御

この章には下記に関する情報が記載されています：

- PID_Compact (S7-1200, S7-1500)
- PID_3Step (S7-1200, S7-1500)
- PID_Temp (S7-1200, S7-1500)
- PID 基本ファンクション (S7-1500)

PID_Compact

この章には下記に関する情報が記載されています：

- PID_Compact の新機能 (S7-1200, S7-1500)
- CPU および FW との互換性 (S7-1200, S7-1500)
- PID_Compact V2.x の CPU 処理時間およびメモリ要件 (S7-1200, S7-1500)
- PID_Compact V2 (S7-1200, S7-1500)
- PID_Compact V1 (S7-1200)

PID_Compact の新機能

PID_Compact V2.2

- **S7-1200 の使用**

  PID_Compact V2.2 以降では、V2 機能による命令をファームウェアバージョン 4.0 以上の S7-1200 でも使用できます。

PID_Compact V2.0

- **エラーに対する応答**

  エラーに対する応答が完全に見直されました。PID_Compact は、初期設定で応答のフォールトトレラント性が大幅に高くなりました 。 この応答は、PID_Compact V1.X を S7-1200 CPU から S7-1500 CPU にコピーする時にセットされます。

通知
システムが損傷することがあります。 初期設定を使用する場合、プロセス値の限界値を超えると PID_Compact が自動モードのままになります。 そのため、システムが損傷することがあります。 システムを損傷から守るために、エラーの場合に制御システムがどのように応答するかを設定する必要があります。

  Error パラメータは、エラーが保留中かどうかを示します。 エラーが保留中でない場合、Error = FALSE となります。 ErrorBits パラメータは、どのエラーが発生したかを示します。 コントローラの再起動や積分動作のクリアなしでエラーおよび警告を確認するには、ErrorAck を使用します。 動作モードを切り替えても、保留でなくなったエラーをクリアすることはできません。

  エラーに対する応答は、SetSubstituteOutput および ActivateRecoverMode で設定できます。

- **代替出力値**

  エラーが発生した場合に出力する代替出力値を設定できます。

- **動作モードの切り替え**

  ModeIN/OUT パラメータで動作モードを指定し、ModeActivate の立ち上がりエッジを使用して動作モードを開始します。 sRet.i_Mode タグは省略されています。

- **マルチインスタンス機能**

  PID_Compact をマルチインスタンス DB として呼び出すことができます。 この場合、テクノロジーオブジェクトは作成されず、パラメータ割り当てインターフェースまたはコミッショニングインターフェースは使用できません。 複数インスタンス DB で直接に PID_Compact のパラメータを割り当て、ウォッチテーブル経由でそれをコミッショニングする必要があります。

- **スタートアップ特性**

  Mode パラメータで指定された動作モードも、Reset の立ち下がりエッジおよび CPU コールドリスタート時に開始されます(RunModeByStartup = TRUE の場合)。

- **ENO 特性**

  ENO は動作モードに応じて設定されます。

  State = 0 の場合、ENO = FALSE です。

  State ≠ 0 の場合、ENO = TRUE です。

- **同調時のセットポイント値の指定**

  CancelTuningLevel タグで、同調時のセットポイントの許容変動を設定します。
- **出力制限値の値の範囲**

  値 0.0 が出力制限値内である必要がなくなりました。
- **積分動作の事前割り当て**

  タグ IntegralResetMode および OverwriteInitialOutputValue を使用して、「無効」動作モードから「自動モード」に切り替える時に、積分動作の事前割り当てを決定することができます。
- **外乱変数オン**

  Disturbance パラメータで外乱変数をオンにすることができます。
- **PID パラメータの既定値**

  以下の初期設定が変更されています。
  - 比例動作の重み付け (PWeighting)が 0.0 から 1.0 に
  - 微分動作の重み付け(DWeighting)が 0.0 から 1.0 に
  - 微分の遅延係数(TdFiltRatio)が 0.0 から 0.2 に
- **タグ名の変更**

  静的タグに、PID_3Step と互換性のある新しい名前が付けられています。

PID_Compact V1.2

- **CPU スタートアップ時の手動モード**

  CPU の開始時に ManualEnable = TRUE の場合、PID_Compact が手動モードで開始されます。ManualEnable の立ち上がり工ッジは必要ありません。
- **プレチューニング**

  プレチューニング時に CPU がオフになった場合、CPU が再びオンになるとプレチューニングが開始されます。

PID_Compact V1.1

- **CPU スタートアップ時の手動モード**

  CPU のスタートアップ時に、PID_Compact は ManualEnable の立ち上がり工ッジでのみ手動モードに切り替わります。 立ち上がり工ッジがない場合は、ManualEnable が FALSE であった最後の動作モードで PID_Compact が開始されます。
- **リセットへの応答**

  Reset での立ち上がりはエラーおよび警告をリセットし、積分動作を解除します。 リセットでの立ち下がりは、直近の有効な動作モードへの変更をトリガします。
- **プロセス値の上限値の既定**

  r_Pv_Hlm の既定値が 120.0 に変更されています。
- **サンプリング時間のモニタ**
  - 現在のサンプリング時間が >= 1.5 x 現在の平均値の場合、または現在のサンプリング時間 <= 0.5 x 現在の平均値の場合、エラーが出力されなくなりました。 サンプリング時間の変動は、自動モードの場合よりはるかに大きくなります。
  - PID_Compact は V2.0 以降の FW と互換性があります。
- **タグへのアクセス**

  以下のタグがユーザープログラムで使用できるようになりました。
  - i_Event_SUT
  - i_Event_TIR

- r_Ctrl_Ioutv

- **トラブルシューティング**

  短いオン時間が短いオフ時間と等しくない場合、PID_Compact が正しいパルスを出力するようになりました。

CPU および FW との互換性

下の表に、PID_Compact のどのバージョンがどの CPU で使用できるかを示します。

CPU	FW	PID_Compact
S7-1200	≥ V4.x	V2.2 V1.2
S7-1200	≥ V3.x	V1.2 V1.1
S7-1200	≥ V2.x	V1.2 V1.1
S7-1200	≥ V1.x	V1.0
S7-1500	≥ V1.5	V2.2 V2.1 V2.0
S7-1500	≥ V1.1	V2.1 V2.0
S7-1500	≥ V1.0	V2.0

PID_Compact V2.x の CPU 処理時間およびメモリ要件

CPU 処理時間

バージョン V2.0 以降の PID_Compact テクノロジーオブジェクトの通常の CPU 処理時間(CPU タイプにより異なる)。

CPU	代表値 PID_Compact V2.x の CPU 処理時間
CPU 1211C ≥ V4.0	300 µs
CPU 1215C ≥ V4.0	300 µs
CPU 1217C ≥ V4.0	300 µs
CPU 1505S ≥ V1.0	45 µs
CPU 1510SP-1 PN ≥ V1.6	85 µs
CPU 1511-1 PN ≥ V1.5	85 µs
CPU 1512SP-1 PN ≥ V1.6	85 µs
CPU 1516-3 PN/DP ≥ V1.5	50 µs
CPU 1518-4 PN/DP ≥ V1.5	4 µs

メモリ要件

バージョン V2.0 以降の PID_Compact テクノロジーオブジェクトのインスタンス DB のメモリ要件。

	PID_Compact V2.x のインスタンス DB のメモリ要件
ロードメモリ要件	約 12000 バイト
合計ワークメモリ要件	788 バイト
保持ワークメモリ要件	44 バイト

PID_Compact V2

この章には下記に関する情報が記載されています：

PID_Compact V2 の説明

説明

PID_Compact 命令は PID コントローラに、比例動作付きのアクチュエータ用の統合された同調機能を提供します。

次の動作モードが使用可能です。

- 無効
- プレチューニング
- 微調整
- 自動モード
- 手動モード
- エラーモニタリング付きの代替出力値

動作モードの詳細については、State パラメータを参照してください。

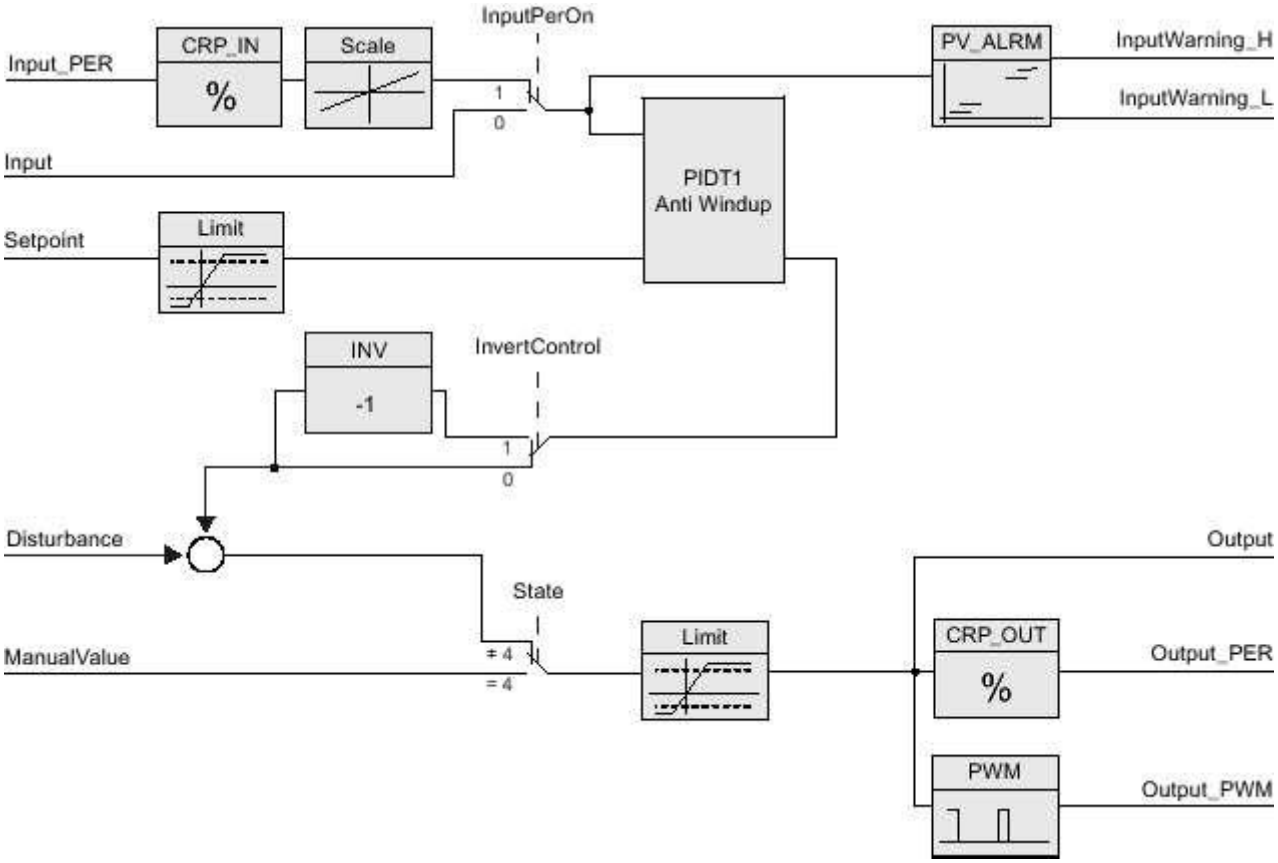
PID アルゴリズム

PID_Compact は、比例および微分動作のワインドアップ防止および重み付け付きの PIDT1 コントローラです。 PID アルゴリズムは、以下の等式に従って動作します。

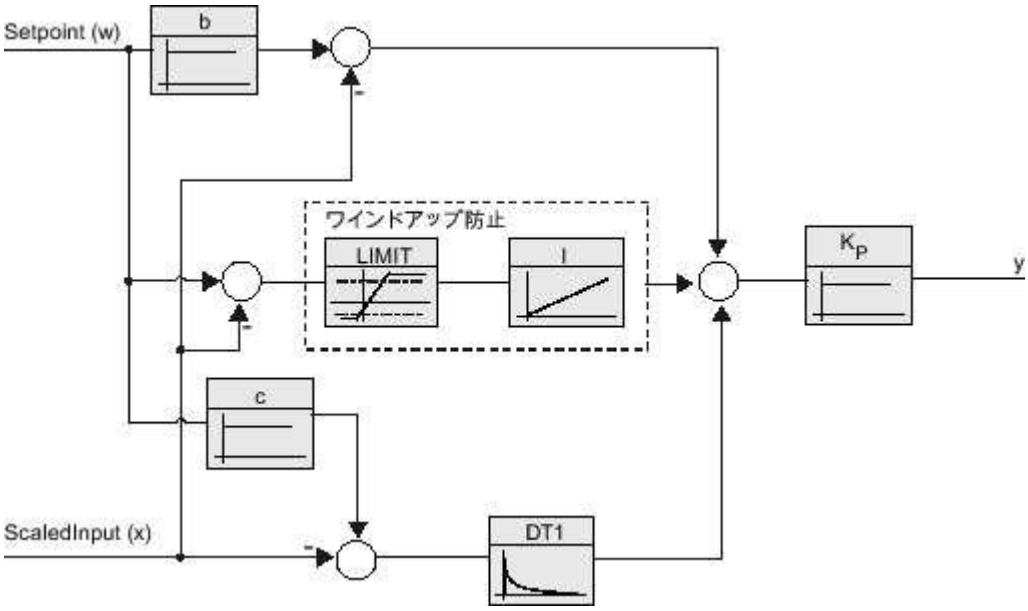
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

シンボル	説明
y	PID アルゴリズムの出力値
K_p	比例ゲイン
s	ラプラス演算子
B	比例動作の重み付け
w	セットポイント
x	プロセス値
T_I	積分動作時間
T_D	微分動作時間
A	微分の遅延係数(微分の遅延 T1 = a × T_D)
C	微分動作の重み付け

PID_Compact のブロックダイアグラム



ワインドアップ防止付きの PIDT1 のブロックダイアグラム



呼び出し

PID_Compact は、周期割り込み DB の一定のタイムスケールで呼び出されます。

マルチインスタンス DB として PID_Compact を呼び出すと、テクノロジーオブジェクトは作成されません。 パラメータ割り当てインターフェースおよびコミッショニングインターフェースは使用でき

ません。 複数インスタンス DB で直接に PID_Compact のパラメータを割り当て、ウォッチテーブル経由でそれをコミッショニングする必要があります。

デバイスへのダウンロード

保持変数の現在値は、PID_Compact 全体をダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

スタートアップ

CPU のスタートアップ時に、PID_Compact が ModeIN/OUT パラメータに保存された動作モードで開始されます。 スタートアップ中に「無効」動作モードに切り替えるには、RunModeByStartup = FALSE を設定します。

エラーに対する応答

自動モード時およびコミッショニング時は、エラーに対する応答は SetSubstituteOutput および ActivateRecoverMode 変数によって決まります。 手動モードでは、応答は SetSubstituteOutput および ActivateRecoverMode には依存しません。 ActivateRecoverMode = TRUE の場合、応答はさらに発生したエラーによっても異なります。

SetSubstituteOutput	ActivateRecoverMode	コンフィグレーションエディタ > 出力値 > Output を以下に設定	応答
関連なし	FALSE	ゼロ(無効)	「無効」モード(State = 0)に切り替え 値 0.0 0 がアクチュエータに転送されます。
FALSE	TRUE	エラー保留中の現在の出力値	「エラーモニタ付きの代替出力値」モード(State = 5)に切り替え 現在の出力値が、エラーの保留中にアクチュエータに転送されます。
TRUE	TRUE	エラーが保留中の間、代替出力値	「エラーモニタ付きの代替出力値」モード(State = 5)に切り替え SubstituteOutput の値がエラーの保留中にアクチュエータに転送されます。

手動モードでは、PID_Compact は ManualValue を出力値として使用します(ManualValue が無効でない場合)。 ManualValue が無効の場合、SubstituteOutput が使用されます。 ManualValue および SubstituteOutput が無効の場合、Config.OutputLowerLimit が使用されます。

Error パラメータは、エラーが保留中かどうかを示します。 エラーが保留中でない場合、Error = FALSE となります。 ErrorBits パラメータは、どのエラーが発生したかを示します。ErrorBits は、Reset または ErrorAck の立ち上がりエッジでリセットされます。

PID_Compact V2 動作モード

プロセス値制限値のモニタリング

Config.InputUpperLimit および Config.InputLowerLimit タグで、プロセス値の上限値と下限値を指定します。 プロセス値がこの制限値を超えている場合、エラーが発生します(ErrorBits = 0001h)。

Config.InputUpperWarning および Config.InputLowerWarning タグで、プロセス値の警告上限値と警告下限値を指定します。 プロセス値がこの警告制限値を超えている場合、警告が発生し(Warning = 0040h)、InputWarning_H または InputWarning_L 出力パラメータが TRUE に変わります。

セットポイントの制限

Config.SetpointUpperLimit および Config.SetpointLowerLimit タグで、セットポイントの上限値と下限値を指定します。PID_Compact は自動的にセットポイントをプロセス値の制限値に制限します。セットポイントを小さい範囲に制限することができます。PID_Compact は、この範囲がプロセス制限値の範囲内であるかどうかをチェックします。 セットポイントがこの制限値を越えると、上限値または下限値がセットポイントとして使用され、出力パラメータ SetpointLimit_H または SetpointLimit_L が TRUE に設定されます。

セットポイントは、すべての動作モードで、制限されます。

出力値の制限

Config.OutputUpperLimit および Config.OutputLowerLimit タグで、出力値の上限値と下限値を指定します。 Output、ManualValue、および SubstituteOutput がこの値に制限されます。 出力制限値は、制御口ジックと一致している必要があります。

有効な出力制限値は、使用している Output によって決まります。

Output	-100.0 ~ 100.0%
Output_PER	-100.0 ~ 100.0%
Output_PWM	0.0 ~ 100.0%

ルール:

OutputUpperLimit > OutputLowerLimit

代替出力値

エラーが発生した場合、PID_Compact はタグ SubstituteOutput で定義している代替出力値を出力できます。 代替出力値は、出力制限値内でなければなりません。

信号妥当性のモニタリング

以下のパラメータの値は、使用時に有効性をモニタされます。

- Setpoint
- Input
- Input_PER
- Disturbance
- ManualValue

- SubstituteOutput
- Output
- Output_PER
- Output_PWM

サンプリング時間のモニタ PID_Compact

理想的には、サンプリング時間は呼び出し OB のサイクルタイムと等値です。PID_Compact 命令は、2 つの呼び出し間の時間間隔を測定します。 これは、現在のサンプル時間です。 動作モードの変更ごと、また、初期スタートアップ時に、最初の 10 個のサンプリング時間からサンプリング時間の平均値が求められます。現在のサンプリング時間とこの平均値の相違が大きすぎると、エラー(Error = 0800h)がトリガされます。

以下の場合、同調時にエラーが発生します。

- 新しい平均値 >= 1.1 x 古い平均値
- 新しい平均値 <= 0.9 x 古い平均値

以下の場合、自動モードでエラーが発生します。

- 新しい平均値 >= 1.5 x 古い平均値
- 新しい平均値 <= 0.5 x 古い平均値

サンプリング時間のモニタを無効にすると(CycleTime.EnMonitoring = FALSE)、PID_Compact を OB1 で呼び出すこともできます。 この場合、サンプリング時間のずれによる制御品質の低下を受け入れる必要があります。

PID アルゴリズムのサンプリング時間

コントロールされたシステムは、出力値の変更に応答するために、一定量の時間を必要とします。 このため、すべてのサイクルごとに出力値を計算することはお奨めしません。 PID アルゴリズムのサンプリング時間は、出力の 2 つの計算の間の時間を表わします。 それは同調時に計算され、サイクルタイムの倍数に四捨五入されます。PID_Compact の他のすべてのファンクションは、呼び出しの都度実行されます。

Output_PWM を使用する場合、出力信号の精度は OB のサイクルタイムに対する PID アルゴリズムのサンプリング時間の比率によって決まります。サイクルタイムは、最低でも PID アルゴリズムのサンプリング時間の 10 倍でなければなりません。

制御論理

出力値を上げるのは通常、プロセス値を上げるために行われます。 これは、標準制御ロジックと呼ばれます。 冷却および放電制御システムでは、制御ロジックを反転させる必要があります。PID_Compact は、負の比例ゲインでは機能しません。 InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。 制御ロジックは、プリチューニングと微調整時にも考慮されます。

PID_Compact V2 の入力パラメータ

パラメータ	データタイプ	デフォルト	説明
Setpoint	REAL	0.0	自動モードでの PID コントローラのセットポイント
Input	REAL	0.0	ユーザープログラムのタグが、プロセス値のソースとして使用されます。 Input パラメータを使用する場合は、Config.InputPerOn = FALSE を設定する必要があります。
Input_PER	INT	0	アナログ入力が、プロセス値のソースとして使用されます。 Input_PER パラメータを使用する場合は、Config.InputPerOn = TRUE を設定する必要があります。
Disturbance	REAL	0.0	外乱変数または事前制御値
ManualEnable	BOOL	FALSE	• FALSE -> TRUE エッジでは「手動モード」 が有効になり、State = 4、Mode はそのまま変わりません。 ManualEnable = TRUE である限り、ModeActivate の立ち上がりエッジで動作モードを変更したり、コミッショニングダイアログを使用することはできません。 • TRUE -> FALSE エッジにより、Mode で指定された動作モードが有効になります。 ModeActivate を使用した動作モードの変更のみをお勧めします。
ManualValue	REAL	0.0	手動値 この値は、手動モードで、出力値として使用されます。 Config.OutputLowerLimit ~ Config.OutputUpperLimit の値が許可されます。
ErrorAck	BOOL	FALSE	• FALSE -> TRUE エッジ ErrorBits および Warning がリセットされます。
Reset	BOOL	FALSE	コントローラを再起動します。 • FALSE -> TRUE エッジ ○「無効」モードに切り替え ○ ErrorBits および Warnings がリセットされます。 ○ 積分動作がクリアされます。 (PID パラメータは保持されます) • Reset = TRUE である限り、PID_Compact は「無効」モードのままです(State = 0)。

			• TRUE -> FALSE エッジ PID_Compact が Mode パラメータに保存された動作モードに切り替わります。
ModeActivate	BOOL	FALSE	• FALSE -> TRUE エッジ PID_Compact が Mode パラメータに保存された動作モードに切り替わります。

PID_Compact V2 の出力パラメータ

Parameter	データタイプ	デフォルト	説明
ScaledInput	REAL	0.0	スケーリングされたプロセス値
「Output」、「Output_PER」、および「Output_PWM」出力は同時に使用することができます。			
Output	REAL	0.0	REAL フォーマットの出力値
Output_PER	INT	0	アナログ出力値
Output_PWM	BOOL	FALSE	パルス幅調整された出力値 出力値は、可変オン/オフ回数によって形成されます。
SetpointLimit_H	BOOL	FALSE	SetpointLimit_H = TRUE の場合、セットポイント絶対上限値に達します(Setpoint ≥ Config.SetpointUpperLimit)。 セットポイントは Config.SetpointUpperLimit に制限されます。
SetpointLimit_L	BOOL	FALSE	SetpointLimit_L = TRUE の場合、セットポイント絶対下限値に達します(Setpoint ≤ Config.SetpointLowerLimit)。 セットポイントは Config.SetpointLowerLimit に制限されます。
InputWarning_H	BOOL	FALSE	InputWarning_H = TRUE の場合、プロセス値が警告上限値に到達済みか、警告上限値を超えています。
InputWarning_L	BOOL	FALSE	InputWarning_L = TRUE の場合、プロセス値が警告下限値に到達済みか、警告下限値未満になっています。
State	INT	0	State パラメータは、PID コントローラの現在の動作モードを 示します。 入力パラメータ Mode および ModeActivate の立ち上がりエッジを使用して、動作モードを変更することができます。 • State = 0: 無効 • State = 1: プリチューニング • State = 2: 微調整 • State = 3: 自動モード • State = 4: 手動モード • State = 5 エラーモニタリング付きの代替出力値
Error	BOOL	FALSE	Error = TRUE の場合、少なくとも 1 つのエラーがこのサイクルで保留中です。
ErrorBits	DWORD	DW#16#0	ErrorBits パラメータは、どのエラーメッセージが保留中かを示します。ErrorBits は保持され、

			Reset または ErrorAck の立ち上がりエッジでリセットされます。

PID_Compact V2 の IN/OUT パラメータ

Parameter	データタイプ	デフォルト	説明
Mode	INT	4	Mode で、PID_Compact の切り替え先である動作モードを指定します。オプションを以下に示します。 • Mode = 0: 無効 • Mode = 1: プリチューニング • Mode = 2: 微調整 • Mode = 3: 自動モード • Mode = 4: 手動モード 動作モードは以下によって有効になります。 • ModeActivate の立ち上がりエッジ • Reset の立ち下がりエッジ • ManualEnable の立ち下がりエッジ • RunModeByStartup = TRUE の場合は CPU のコールドリスタート Mode は保持されます。 動作モードの詳細については、パラメータ State と Mode V2 を参照してください。

PID_Compact V2 の静的タグ

リストにない変数を変更してはいけません。 これらは、システム内部の目的のためだけに使用されます。

タグ	データタイプ	既定	説明
IntegralResetMode	INT	1	タグ IntegralResetMode は、「無効」動作モードから「自動モード」に切り替える時に PIDCtrl.IntegralSum をあらかじめどのように割り当てするかを決定します。 この設定は 1 サイクルの間だけ有効です。 オプションを以下に示します。 • IntegralResetMode = 0: まるめ IntegralSum の値は、切り替えがバンプレスとなるように事前割り当てされます。 • IntegralResetMode = 1: 削除 IntegralSum の値が削除されます。 制御偏差によって出力値がジャンプ変動しません。 • IntegralResetMode = 2: 保持 IntegralSum の値が変更されません。 ユーザープログラムを使用して新しい値を定義できます。 • IntegralResetMode = 3: 事前に割り当て IntegralSum の値が、Output が値 OverwriteInitialOutputValue を基準にして計算されるよう自動的に事前割り当てされます。この設定は、たとえば、オーバーライドコントローラなどに有用です。
OverwriteInitialOutputValue	REAL	0.0	IntegralResetMode = 3 の場合、次のサイクルで Output = OverwriteInitialOutputValue となるよう、IntegralSum の値が自動的に事前割り当てされます。
RunModeByStartup	BOOL	TRUE	CPU のリスタート後に、Mode パラメータの動作モードを有効にします。 RunModeByStartup = TRUE の場合、PID_Compact は CPU のリスタート後に、Mode パラメータに保存された動作モードで開始されます。 RunModeByStartup = FALSE の場合、PID_Compact は CPU のリスタート後も「無効」モードのままです。
LoadBackUp	BOOL	FALSE	LoadBackUp = TRUE の場合、最後の PID パラメータセットが再読み込みされます。このセットは、最後の同調前に保存されたもの

			です。LoadBackUp は自動的に FALSE に再セットされます。
PhysicalUnit	INT	0	プロセス値およびセットアポイントの測定単位(たとえば、℃または ºF)
PhysicalQuantity	INT	0	プロセス値およびセットポイントの物理量(たとえば、温度)
ActivateRecoverMode	BOOL	TRUE	タグ ActivateRecoverMode V2 はエラーに対する応答を決定します。
Warning	DWORD	0	タグ Warning V2 は、Reset = TRUE または ErrorAck =TRUE 以降の警告を示します。Warning は保持されます。
Progress	REAL	0.0	同調の進捗状況(パーセンテージ(0.0 ~ 100.0)で表示)
CurrentSetpoint	REAL	0.0	CurrentSetpoint は常に、現在のセットポイントを示します。 この値は同調中は固定されます。
CancelTuningLevel	REAL	10.0	同調中のセットポイントの許容変動。 同調は以下の状態になるまでキャンセルされません。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel
SubstituteOutput	REAL	0.0	代替出力値 次の条件が満たされている場合、代替出力値が使用されます。 • 自動モードでエラーが発生しました。 • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE
SetSubstituteOutput	BOOL	TRUE	SetSubstituteOutput = TRUE および ActivateRecoverMode = TRUE の場合、エラーが保留になっている間は設定された代替出力値が出力されます。 SetSubstituteOutput = FALSE および ActivateRecoverMode = TRUE の場合、エラーが保留になっている間、アクチュエータは現在の出力値のままです。 ActivateRecoverMode = FALSE の場合、SetSubstituteOutput は無効です。 SubstituteOutput が無効の場合(ErrorBits = 20000h)、代替出力値は出力できません。
Config.InputPerOn	BOOL	TRUE	InputPerOn = TRUE の場合、Input_PER パラメータが使用されます。 InputPerOn = FALSE の場合、Input パラメータが使用されます。
Config.InvertControl	BOOL	FALSE	反転制御ロジック

			InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。
Config.InputUpperLimit	REAL	120.0	プロセス値の上限値 この制限値が守られていることを確認するために、Input と Input_PER をモニタします。 I/O 入力で、プロセス値が標準範囲よりも最大 18%大きくなる場合があります(範囲オーバー)。 この事前割り当てにより、「プロセス値の上限値」のオーバーによるエラーは通知されません。 断線および短絡のみが認識され、PID_Compact が設定されたエラーに対する応答に従って応答します。 InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	プロセス値の下限値 この制限値が守られていることを確認するために、Input と Input_PER をモニタします。 InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	3.402822e+38	プロセス値の警告上限値 プロセス値制限範囲外の InputUpperWarning を設定すると、設定されたプロセス値の上限絶対値が警告上限値として使用されます。 プロセス値制限範囲内の InputUpperWarning を設定すると、この値が警告上限値として使用されます。 InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	プロセス値の警告下限値 プロセス値制限範囲外の InputLowerWarning を設定すると、設定されたプロセス値の下限絶対値が警告下限値として使用されます。 プロセス値制限範囲内の InputLowerWarning を設定すると、この値が警告下限値として使用されます。 InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	出力値の上限値 詳細は、OutputLowerLimit を参照してください。 OutputUpperLimit > OutputLowerLimit
Config.OutputLowerLimit	REAL	0.0	出力値の下限値 Output または Output_PER の場合、-100.0 ～ +100.0 の値の範囲(0 を含めて)が有効で

			す。 -100.0 では Output_PER = -27648、+100.0 では Output_PER = 27648 です。 Output_PWM の場合、値の範囲 0.0～+100.0 が適用されます。 出力制限値は、制御ロジックと一致している必要があります。 OutputLowerLimit < OutputUpperLimit
Config.SetpointUpperLimit	REAL	3.402822e+38	セットポイントの上限値 プロセス値制限範囲外の SetpointUpperLimit を設定すると、設定されたプロセス値の上限絶対値がセットポイントの上限値として使用されます。 プロセス値制限範囲内の SetpointUpperLimit を設定すると、この値がセットポイントの上限値として使用されます。
Config.SetpointLowerLimit	REAL	-3.402822e+38	セットポイントの下限値 プロセス値制限範囲外の SetpointLowerLimit を設定すると、設定されたプロセス値の下限絶対値がセットポイントの下限値として使用されます。 プロセス値制限範囲内の SetpointLowerLimit を設定すると、この値がセットポイントの下限値として使用されます。
Config.MinimumOnTime	REAL	0.0	パルス幅変調方式の最小オン時間(秒)が、以下に丸められます。 MinimumOnTime = n×CycleTime.Value
Config.MinimumOffTime	REAL	0.0	パルス幅変調方式の最小オフ時間(秒)が、以下に丸められます。 MinimumOffTime = n×CycleTime.Value
Config.InputScaling.Upper-PointIn	REAL	27648.0	Input_PER (High)のスケーリング Input_PER は、2 つの値のペア UpperPoint-Out、 UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.Lower-PointIn	REAL	0.0	Input_PER (Low)のスケーリング Input_PER は、2 つの値のペア UpperPoint-Out、 UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.Upper-PointOut	REAL	100.0	スケーリングされた High プロセス値 Input_PER は、2 つの値のペア UpperPoint-Out、 UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.Lower-PointOut	REAL	0.0	スケーリングされた Low プロセス値

			Input_PER は、2 つの値のペア UpperPoint-Out、 UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
CycleTime.StartEstimation	BOOL	TRUE	CycleTime.StartEstimation = TRUE の場合、サイクルタイムの自動測定が開始されます。CycleTime.StartEstimation = FALSE の場合、すぐに、測定が完了します。
CycleTime.EnEstimation	BOOL	TRUE	CycleTime.EnEstimation = TRUE の場合、PID_Compact サンプリング時間が計算されます。 CycleTime.EnEstimation = FALSE の場合、PID_Compact サンプリング時間は計算されず、CycleTime.Value の設定を手動で修正する必要があります。
CycleTime.EnMonitoring	BOOL	TRUE	CycleTime.EnMonitoring = FALSE の場合、PID_Compact サンプリング時間はモニタされません。 PID_Compact をサンプリング時間内に実行できない場合、エラー(Error-Bits=0800h)は出力されず、PID_Compact は「無効」モードに切り替わりません。
CycleTime.Value	REAL	0.1	PID_Compact サンプリング時間(秒単位) CycleTime.Value が自動的に測定されます。通常、呼び出し OB のサイクルタイムと同じ値です。
CtrlParamsBackUp.Gain	REAL	1.0	比例ゲインの保存エリア LoadBackUp = TRUE で、CtrlParamsBack-Up 構造体から値を再読み込みできます。
CtrlParamsBackUp.Ti	REAL	20.0	積分動作時間[s]の保存エリア
CtrlParamsBackUp.Td	REAL	0.0	微分動作時間[s]の保存エリア
CtrlParamsBackUp.TdFiltRa-tio	REAL	0.2	微分の遅延係数の保存エリア
CtrlParamsBackUp.PWeight-ing	REAL	1.0	比例動作重み係数の保存エリア
CtrlParamsBackUp.DWeight-ing	REAL	1.0	微分操作重み係数の保存エリア
CtrlParamsBackUp.Cycle	REAL	1.0	PID アルゴリズムのサンプリング時間の保存エリア
PIDSelfTune.SUT.Calculate-Params	BOOL	FALSE	制御システムのプロパティは同調中に保存されます。 SUT.CalculateParams = TRUE の場合、プレチューニングのパラメータはこのプロパティに従って再計算されます。 これによって、コントローラ同調を繰り返すことなく、パラメータ計算方法を変更できます。 計算後、SUT.CalculateParams が FALSE に設定されます。
PIDSelfTune.SUT.TuneRule	INT	0	プレチューニング時のパラメータの計算に使用される方法:

			• SUT.TuneRule = 0: Chien、Hrones、および Reswick に従った PID • SUT.TuneRule = 1: Chien、Hrones、および Reswick に従った PI
PIDSelfTune.SUT.State	INT	0	SUT.State タグは、プレチューニングの現在のフェーズを示します。 • State = 0 プレチューニングの初期化 • State = 100: 標準偏差の計算 • State = 200 変曲点の決定 • State = 9900 プレチューニングが成功 • State = 1 プレチューニングが失敗
PIDSelfTune.TIR.RunIn	BOOL	FALSE	RunIn タグで、微調整をプレチューニングなしで実行することも指定できます。 • RunIn = FALSE 無効なモードまたは手動モードで微調整が開始されると、プレチューニングが開始されます。 プレチューニングの要件が満たされない場合、PID_Compact は、RunIn = TRUE の場合と同様に応答します。 自動モードで微調整が開始されると、システムは既存の PID パラメータを使用してセットポイントを制御します。 この場合のみ、微調整が開始します。 プレチューニングが行えない場合、PID_Compact は同調が開始されたモードに切り替わります。 • RunIn = TRUE プレチューニングはスキップされます。 PID_Compact は、最小または最大出力値でセットポイントに到達することを試みます。 この結果、オーバーシュートが増大する可能性があります。 微調整が自動的に開始します。 微調整後、RunIn が FALSE に設定されます。
PIDSelfTune.TIR.Calculate-Params	BOOL	FALSE	制御システムのプロパティは同調中に保存されます。 TIR.CalculateParams = TRUE の場合、微調整のパラメータはこのプロパティに従って再計算されます。 これによって、コントローラ同調を繰り返すことなく、パラメータ計算方法を変更できます。 計算後、TIR.CalculateParams が FALSE に設定されます。
PIDSelfTune.TIR.TuneRule	INT	0	微調整時のパラメータの計算に使用される方法: • TIR.TuneRule = 0: PID (自動) • TIR.TuneRule = 1: PID (高速) • TIR.TuneRule = 2: PID (低速)

			• TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	TIR.State タグは、微調整の現在のフェーズを示します。 • State = -100 微調整が実行可能ではありません。 プレチューニングが最初に実行されます。 • State = 0: 微調整の初期化 • State = 200: 標準偏差の計算 • State = 300: セットポイントへの到達の試み • State = 400: 既存の PID パラメータによるセットポイントへの到達の試み (プレチューニングが正常に行われた場合) • State = 500: 振動の決定とパラメータの計算 • State = 9900: 微調整が成功 • State = 1: 微調整が成功
PIDCtrl.IntegralSum	REAL	0.0	現在の積分動作
Retain.CtrlParams.Gain	REAL	1.0	有効な比例ゲイン 制御ロジックを反転するには、Config.InvertControl タグを使用します。 Gain の負の値も制御ロジックを反転させます。 制御ロジックの設定には、InvertControl だけを使用することをお勧めします。 制御ロジックは、InvertControl = TRUE および Gain < 0.0 の場合にも反転します。 Gain は保持されます。
Retain.CtrlParams.Ti	REAL	20.0	• CtrlParams.Ti > 0.0: 有効な積分動作時間 • CtrlParams.Ti = 0.0: 積分動作が無効です Ti は保持されます。
Retain.CtrlParams.Td	REAL	0.0	• CtrlParams.Td > 0.0: 有効な微分動作時間 • CtrlParams.Td = 0.0: 微分動作が無効です Td は保持されます。
Retain.CtrlParams.TdFiltRatio	REAL	0.2	動作中の微分の遅延係数 微分の遅延係数によって、微分動作の影響を遅らせます。 微分の遅延 = 微分動作時間 × 微分の遅延係数 • 0.0: 微分動作は 1 サイクルに限って有効であり、ほとんど影響はありません。 • 0.5: この値は、主要な時定数が 1 つの制御システムで実際に有効であることが実証されています。

			• > 1.0: この係数が大きいほど、微分動作の影響が遅くなります。 TdFiltRatio は保持されます。
Retain.CtrlParams.PWeighting	REAL	1.0	有効な比例動作重み 比例動作はセットポイントの変更とともに弱くなることがあります。 0.0 ~ 1.0 の値を適用できます。 • 1.0: セットポイントの変更に対する比例動作が完全に有効です • 0.0: セットポイントの変更に対する比例動作が有効でありません プロセス値が変更になった場合は、比例動作は常に完全に有効です。 PWeighting は保持されます。
Retain.CtrlParams.DWeighting	REAL	1.0	有効な微分動作重み 微分動作はセットポイントの変更とともに弱くなることがあります。 0.0 ~ 1.0 の値を適用できます。 • 1.0: セットポイントの変更時に微分動作が完全に有効です • 0.0: セットポイントの変更時に微分動作が有効でありません プロセス値が変更になった場合は、微分動作は常に完全に有効です。 DWeighting は保持されます。
Retain.CtrlParams.Cycle	REAL	1.0	PID アルゴリズムの有効なサンプリング時間 CtrlParams.Cycle が同調時に計算され、CycleTime.Value の整数倍に丸められます。 Cycle は保持されます。

注記
PID コントローラの誤動作を防止するには、「無効」モードでこの表に記載されているタグを変更します。

PID_Compact V2 インターフェースの変更

下の表に、PID_Compact 命令インターフェースで何が変更されたかを示します。

PID_Compact V1	PID_Compact V2	変更
Input_PER	Input_PER	データタイプがワードから整数に
	Disturbance	新規作成
	ErrorAck	新規作成
	ModeActivate	新規作成
Output_PER	Output_PER	データタイプがワードから整数に
Error	ErrorBits	名前の変更
	Error	新規作成
	Mode	新規作成
sb_RunModeByStartup	RunModeByStartup	ファンクション
	IntegralResetMode	
	OverwriteInitialOutputValue	新規作成
	SetSubstituteOutput	新規作成
	CancelTuningLevel	新規作成
	SubstituteOutput	新規作成

下の表に名前を変更した変数を示します。

PID_Compact V1.x	PID_Compact V2
sb_GetCycleTime	CycleTime.StartEstimation
sb_EnCyclEstimation	CycleTime.EnEstimation
sb_EnCyclMonitoring	CycleTime.EnMonitoring
sb_RunModeByStartup	RunModeByStartup
si_Unit	PhysicalUnit
si_Type	PhysicalQuantity
sd_Warning	Warning
sBackUp.r_Gain	CtrlParamsBackUp.Gain
sBackUp.r_Ti	CtrlParamsBackUp.Ti
sBackUp.r_Td	CtrlParamsBackUp.Td
sBackUp.r_A	CtrlParamsBackUp.TdFiltRatio
sBackUp.r_B	CtrlParamsBackUp.PWeighting
sBackUp.r_C	CtrlParamsBackUp.DWeighting
sBackUp.r_Cycle	CtrlParamsBackUp.Cycle

sPid_Calc.r_Cycle	CycleTime.Value
sPid_Calc.b_RunIn	PIDSelfTune.TIR.RunIn
sPid_Calc.b_CalcParamSUT	PIDSelfTune.SUT.CalculateParams
sPid_Calc.b_CalcParamTIR	PIDSelfTune.TIR.CalculateParams
sPid_Calc.i_CtrlTypeSUT	PIDSelfTune.SUT.TuneRule
sPid_Calc.i_CtrlTypeTIR	PIDSelfTune.TIR.TuneRule
sPid_Calc.r_Progress	Progress
sPid_Cmpt.r_Sp_Hlm	Config.SetpointUpperLimit
sPid_Cmpt.r_Sp_Llm	Config.SetpointLowerLimit
sPid_Cmpt.r_Pv_Norm_IN_1	Config.InputScaling.LowerPointIn
sPid_Cmpt.r_Pv_Norm_IN_2	Config.InputScaling.UpperPointIn
sPid_Cmpt.r_Pv_Norm_OUT_1	Config.InputScaling.LowerPointOut
sPid_Cmpt.r_Pv_Norm_OUT_2	Config.InputScaling.UpperPointOut
sPid_Cmpt.r_Lmn_Hlm	Config.OutputUpperLimit
sPid_Cmpt.r_Lmn_Llm	Config.OutputLowerLimit
sPid_Cmpt.b_Input_PER_On	Config.InputPerOn
sPid_Cmpt.b_LoadBackUp	LoadBackUp
sPid_Cmpt.b_InvCtrl	Config.InvertControl
sPid_Cmpt.r_Lmn_Pwm_PPTm	Config.MinimumOnTime
sPid_Cmpt.r_Lmn_Pwm_PBTm	Config.MinimumOffTime
sPid_Cmpt.r_Pv_Hlm	Config.InputUpperLimit
sPid_Cmpt.r_Pv_Llm	Config.InputLowerLimit
sPid_Cmpt.r_Pv_HWrn	Config.InputUpperWarning
sPid_Cmpt.r_Pv_LWrn	Config.InputLowerWarning
sParamCalc.i_Event_SUT	PIDSelfTune.SUT.State
sParamCalc.i_Event_TIR	PIDSelfTune.TIR.State
sRet.i_Mode	sRet.i_Mode は省略されています。 動作モードは Mode および ModeActivate を使用して変更されています。
sRet.r_Ctrl_Gain	Retain.CtrlParams.Gain
sRet.r_Ctrl_Ti	Retain.CtrlParams.Ti
sRet.r_Ctrl_Td	Retain.CtrlParams.Td
sRet.r_Ctrl_A	Retain.CtrlParams.TdFiltRatio
sRet.r_Ctrl_B	Retain.CtrlParams.PWeighting
sRet.r_Ctrl_C	Retain.CtrlParams.DWeighting
sRet.r_Ctrl_Cycle	Retain.CtrlParams.Cycle

パラメータ State と Mode V2

パラメータの相互関係

State パラメータは、PID コントローラの現在の動作モードを示します。 State パラメータを変更することはできません。

ModeActivate の立ち上がりエッジで、PID_Compact が ModeIN-OUT パラメータに保存された動作モードに切り替わります。

CPU がオンになるか、Stop から RUN モードに切り替わると、PID_Compact が Mode パラメータに保存された動作モードで開始されます。 PID_Compact を「無効」モードのままにするには、RunModeByStartup = FALSE を設定してください。

値の意味

State / Mode	動作モードの説明
0	無効 「無効」動作モードでは、出力値 0.0 は Config.OutputUpperLimit および Config.OutputLowerLimit とは関係なく常に出力されます。 パルス幅変調がオフ。
1	プリチューニング プリチューニングでは、出力値のジャンプ変動に対するプロセス応答が決定され、変曲点が検索されます。 PID パラメータは、最大立ち上がり速度と制御システムのデッドタイムから計算されます。 プリチューニングと微調整を実行すると最良の PID パラメータを取得できます。 プリチューニングの必要条件 • 無効(State = 0)、手動モード(State = 4)、または自動モード(State = 3)であること。 • ManualEnable = FALSE • Reset = FALSE • プロセス値は、セットポイントに近すぎる値であってはいけません。 |Setpoint - Input| > 0.3 * | Config.InputUpperLimit - Config.InputLowerLimit|および |Setpoint - Input| > 0.5 * |Setpoint| • セットポイントとプロセス値が、設定済みの制限範囲内であること。 プロセス値が安定しているほど、PID パラメータの計算が簡単になり、結果が正確になります。 プロセス値の上昇率がノイズと比べて大幅に高ければ、プロセス値のノイズは許容できます。 セットポイントが、CurrentSetpoint タグで固定されます。 同調は以下の場合にキャンセルされます。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel PID パラメータは再計算される前にバックアップされ、LoadBackUp で再度有効になります。

	プリチューニングが正常に終了すると、コントローラが自動モードに切り替わります。プリチューニングが失敗した場合、動作モードの切り替えは ActivateRecoverMode に依存します。 プリチューニングのフェーズは、 PIDSelfTune.SUT.State で示されます。
2	微調整 微調整は、プロセス値の一定の制限された振動を生成します。 PID パラメータは、この振動の振幅と周波数に基づいて再計算されます。 通常は微調整に基づく PID パラメータの方が、プリチューニングに基づく PID パラメータよりも優れたマスタコントロールと外乱特性を実現します。 プリチューニングと微調整を実行すると最良の PID パラメータを取得できます。 PID_Compact は、プロセス値のノイズよりも大きな振動の生成を自動的に試行します。プロセス値の安定性によって微調整が受ける影響は最小限です。 セットポイントが、CurrentSetpoint タグで固定されます。 同調は以下の場合にキャンセルされます。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel PID パラメータは再計算される前にバックアップされ、LoadBackUp で再度有効になります。 微調整の必要条件 • 外乱が予測されないこと。 • セットポイントとプロセス値が、設定済みの制限範囲内であること。 • ManualEnable = FALSE • Reset = FALSE • 自動(State = 3)、無効(State = 0)、または手動(State = 4)モードであること。 微調整は、開始されたときのモードに従って以下のように実行されます。 • 自動モード(State = 3) 同調を使用して既存の PID パラメータを改善する場合は、自動モードで微調整を開始します。 PID_Compact は、制御ループが安定し、微調整の必要条件が満たされるまでは、既存の PID パラメータを使用してシステムを制御します。 制御ループが安定して微調整の必要条件が満たされた場合のみ、微調整が開始します。 • 無効モード(State = 0)または手動モード(State = 4) プリチューニングの必要条件が満たされると、プリチューニングが開始されます。 制御ループが安定し、微調整の必要条件が満たされるまでは、決定された PID パラメータが制御に使用されます。 プリチューニングのプロセス値が既にセットポイントに非常に近くなっているか、PIDSelfTune.TIR.RunIn = TRUE の場合、最小出力値または最大出力値でセットポイントに到達しようと試みます。この結果、オーバーシュートが増大する可能性があります。 この場合のみ、微調整が開始されます。 微調整が正常に終了すると、コントローラが自動モードに切り替わります。 微調整が失敗した場合、動作モードの切り替えは ActivateRecoverMode に依存します。 「微調整」のフェーズは、 PIDSelfTune.TIR.State で示されます。
3	自動モード

	自動モードでは、PID_Compact は、指定されたパラメータに従って、コントロールされるシステムを訂正します。 以下の必要条件の 1 つが満たされた場合、コントローラは自動モードに切り替わります。 • プリチューニングの正常終了 • 微調整の正常終了 • ModeIN-OUT パラメータの値 3 への変更と ModeActivate の立ち上がりエッジ 手動モードから自動モードへの切り替えは、コミッショニングエディタで実行した場合にのみバンプなしです。 ActivateRecoverMode タグは、自動モードで考慮されます。
4	手動モード 手動モードでは、手動出力値を ManualValue パラメータに指定します。 この動作モードは、ManualEnable = TRUE を使用して有効にすることもできます。Mode および ModeActivate だけを使用して動作モードを変更することをお勧めします。. 手動モードから自動モードへの切り替えはバンプなしです。 手動モードは、エラーが保留中でも使用可能です。
5	エラーモニタリング付きの代替出力値 制御アルゴリズムは無効です。 SetSubstituteOutput タグは、この動作モードでどの出力値を出力するかを決定します。 • SetSubstituteOutput = FALSE: 最後の有効な出力値 • SetSubstituteOutput = TRUE: 代替出力値 この動作モードは、Mode = 5 を使用して有効にすることはできません エラー発生時は、以下のすべての条件が満たされている場合、「無効」モードの代わりに有効になります。 • 自動モード(Mode = 3) • ActivateRecoverMode = TRUE • ActivateRecoverMode が有効になる 1 つまたは複数のエラーが発生しました。 保留中のエラーがなくなると、PID_Compact は直ちに自動モードに戻ります。

ENO 特性

State = 0 の場合、ENO = FALSE です。

State ≠ 0 の場合、ENO = TRUE です。

コミッショニング時の動作モードの自動切り替え

プリチューニングまたは微調整が正常に行われると、自動モードが有効になります。 下の表に、プリチューニングが正常に行われた場合に Mode および State がどのように切り替わるかを示します。

サイクル番号	Mode	State	操作
0	4	4	セット Mode = 1
1	1	4	セット ModeActivate = TRUE

1	4	1	State の値が Mode パラメータに保存されます。 プリチューニングが開始されます。
n	4	1	プリチューニングの正常終了
n	3	3	自動モードが開始されます。

PID_Compact は、エラー発生時に自動的に動作モードを切り替えます。 下の表に、プリチューニングでエラーが発生した場合 Mode および State がにどのように切り替わるかを示します。

サイクル番号	Mode	State	操作
0	4	4	セット Mode = 1
1	1	4	セット ModeActivate = TRUE
1	4	1	State の値が Mode パラメータに保存されます。 プリチューニングが開始されます。
n	4	1	プリチューニングがキャンセルされました
n	4	4	手動モードが開始されます。

ActivateRecoverMode = TRUE の場合、Mode パラメータに保存された動作モードが有効になります。 プリチューニングまたは微調整の開始時に、PID_Compact が State の値を ModeIN-OUT パラメータに保存しました。そのため、PID_Compact は同調が開始された動作モードに切り替わります。

ActivateRecoverMode = FALSE の場合、システムは「無効」動作モードに切り替わります。

パラメータ ErrorBits V2

複数のエラーが同時に保留中の場合、ErrorBits の値がバイナリ加算で表示されます。 たとえば、ErrorBits = 0003h の表示はエラー 0001h および 0002h が同時に保留中であることを示します。

手動モードでは、PID_Compact は ManualValue を出力値として使用します。 例外は Errorbits = 10000h です。

ErrorBits (DW#16#...)	説明
0000	エラーはありません。
0001	「Input」パラメータがプロセス値制限範囲外です。 • Input > Config.InputUpperLimit または • Input < Config.InputLowerLimit エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Compact は自動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。
0002	「Input_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Compact は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Compact は直ちに自動モードに戻ります。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。
0004	微調整中のエラー プロセス値の振幅を維持できませんでした。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0008	プレチューニングの開始時のエラー プロセス値がセットポイントに近すぎます。 微調整を開始してください。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0010	セットポイントが、同調時に変更されました。 CancelTuningLevel タグで、セットポイントの許容変動を設定できます。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0020	プレチューニングを微調整中に行うことはできません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は微調整モードのままです。
0080	プレチューニング中のエラー 出力制限値の設定が不正です。

	出力値の制限値が正しく設定されていて、制御ロジックと一致しているかどうかをチェックします。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0100	微調整時のエラーによって、パラメータが無効になりました。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0200	「Input」パラメータの値が無効です。 値が無効な番号書式です。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Compact は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Compact は直ちに自動モードに戻ります。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。
0400	出力値の計算が失敗しました。 PID パラメータをチェックしてください。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Compact は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Compact は直ちに自動モードに戻ります。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。
0800	サンプリング時間エラー: PID_Compact が、サイクル割り込み OB のサンプリング時間内に呼び出されていません。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Compact は自動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。
1000	「Setpoint」パラメータの値が無効です。 値が無効な番号書式です。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Compact は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Compact は直ちに自動モードに戻ります。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。
10000	「ManualValue」パラメータの値が無効です。 値が無効な番号書式です。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Compact は SubstituteOutput を出力値として使用します。 ManualValue で有効な値を指定するとすぐに、PID_Compact はそれを出力値として使用します。
20000	SubstituteOutput タグの値が無効です。 値が無効な番号書式です。 PID_Compact は、出力下限値を出力値として使用します。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_Compact は自動モードに戻ります。
40000	「Disturbance」パラメータの値が無効です。値が無効な番号書式です。

	エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、Disturbance は 0 に設定されます。PID_Compact は自動モードのままです。 プレチューニングまたは微調整がエラー発生前に有効になっていて、 ActivateRecoverMode = TRUE の場合、PID_Compact は Mode パラメータに保存されている動作モードに切り替わります。 現在のフェーズの Disturbance が出力値に影響しない場合、同調はキャンセルされません。

タグ ActivateRecoverMode V2

ActivateRecoverMode タグは、エラーに対する応答を決定します。 Error パラメータは、エラーが保留中かどうかを示します。 エラーが保留中でない場合、Error = FALSE となります。 ErrorBits パラメータは、どのエラーが発生したかを示します。

自動モード

通知
システムが損傷することがあります。 ActivateRecoverMode = TRUE の場合、エラーが発生していてプロセス制限値を越えていても PID_Compact は自動モードのままです。 そのため、システムが損傷することがあります。 システムを損傷から守るために、エラーの場合に制御システムがどのように応答するかを設定する必要があります。

ActivateRecoverMode	説明
FALSE	PID_Compact は、エラー発生時に自動的に「無効」モードに切り替わります。 コントローラは、Reset の立ち下りエッジまたは ModeActivate の立ち上がりエッジでのみ有効になります。
TRUE	自動モードでエラーが頻繁に発生する場合、この設定は制御応答にマイナスの影響を及ぼします。エラーが発生するたびに、PID_Compact が計算された出力値と代替出力値の間で切り替わるからです。 この場合は、ErrorBits パラメータをチェックして、エラーの原因を取り除いてください。 以下のエラーの 1 つまたは複数が発生しても、PID_Compact は自動モードのままです。 • 0001h:「Input」パラメータがプロセス値制限範囲外です。 • 0800h: サンプリング時間のエラー • 40000h:パラメータ「Disturbance」の値が無効です。 以下の 1 つまたは複数のエラーが発生する場合、PID_Compact は、「エラーモニタ付き代替出力値」モードに切り替わります。 • 0002h: 「Input_PER」パラメータの値が無効です。 • 0200h: 「Input」パラメータの値が無効です。 • 0400h: 出力値の計算が失敗しました。 • 1000h: 「Setpoint」パラメータの値が無効です。 以下のエラーが発生した場合、PID_Compact は「エラーモニタ付きつき代替出力値」モードに切り替わり、アクチュエータを Config.OutputLowerLimit に移動します。 • 20000h: SubstituteOutput タグの値が無効です。 値が無効な番号フォーマットです。 この特性は SetSubstituteOutput とは無関係です。 保留中のエラーがなくなると、PID_Compact は直ちに自動モードに戻ります。

プリチューニングと微調整

ActivateRe-coverMode	説明
FALSE	PID_Compact は、エラー発生時に自動的に「無効」モードに切り替わります。 コントローラは、Reset の立ち下りエッジまたは ModeActivate の立ち上がりエッジでのみ有効になります。
TRUE	以下のエラーが発生した場合、PID_Compact はアクティブモードのままです。 • 0020h: プリチューニングを微調整中に行うことはできません。 以下のエラーは無視されます。 • 10000h: 「ManualValue」パラメータの値が無効です。 • 20000h: SubstituteOutput タグの値が無効です。 これ以外のエラーが発生した場合、PID_Compact は同調をキャンセルして同調が開始されたモードに切り替わります。

手動モード

ActivateRecoverMode は手動モードでは無効です。

タグ Warning V2

複数の警告が同時に保留中の場合、Warning タグの値がバイナリ加算で表示されます。 たとえば、警告 0003h の表示は警告 0001h および 0002h が同時に保留中であることを示します。

Warning (DW#16#....)	説明
0000	保留中の警告がありません。
0001	プレチューニング時に変曲点が見つかりませんでした。
0004	セットポイントが、設定済みの制限値に制限されました。
0008	選択された計算方法で、コントロールされるシステムの必要なプロパティの中に定義されなかったものがあります。 代わりに、PID パラメータが TIR.TuneRule = 3 メソッドによって計算されました。
0010	Reset = TRUE または ManualEnable = TRUE.のために、動作モードを変更できませんでした。
0020	呼び出し OB のサイクルタイムが、PID アルゴリズムのサンプリング時間を制限します。 さらに短い OB サイクルタイムを使用して、結果を改善してください。
0040	プロセス値が、その警告制限値の 1 つを超えました。
0080	Mode での値が不正です。 動作モードは変更されません。
0100	手動値が、コントローラ出力の制限値に制限されました。
0200	調整の指定されたルールはサポートされていません。 PID パラメータは計算されません。
1000	代替出力値が出力制限値を超えているため、代替出力値に達することができません。

以下の警告は、原因が取り除かれるとすぐに削除されます。

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h

他のすべての警告は、Reset または ErrorAck の立ち上がりエッジで解除されます。

PID_Compact V1

この章には下記に関する情報が記載されています：

PID_Compact V1 の説明

説明

PID_Compact 命令は、PID コントローラに、自動および手動モードでの統合された同調機能を提供します。

呼び出し

PID_Compact は、一定の間隔(呼び出し OB(可能な場合、周期割り込み OB)のサイクルタイム)で呼び出されます。

デバイスへのダウンロード

保持タグの現在値は、PID_Compact を完全にダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

スタートアップ

CPU のスタートアップ時には、PID_Compact は、最後に有効であった動作モードで開始されます。PID_ Compact を[無効]モードのままに保持するには、sb_RunModeByStartup = FALSE を設定してください。

サンプリング時間のモニタ PID_Compact

理想的には、サンプリング時間は呼び出し OB のサイクルタイムと等値です。PID_Compact 命令は、2 つの呼び出し間の時間間隔を測定します。 これは、現在のサンプル時間です。 動作モードの変更ごと、また、初期スタートアップ時に、最初の 10 個のサンプリング時間からサンプリング時間の平均値が求められます。 現在のサンプリング時間がこの平均値から大きくずれ過ぎている場合は、 Error = 0800 hex が発生し、PID_Compact が「無効」モードに切り替わります。

以下の条件が発生すると、PID_Compact、バージョン 1.1 以降はコントローラの同調中に「無効」モードに設定されます。

- 新しい平均値 >= 1.1 x 古い平均値
- 新しい平均値 <= 0.9 x 古い平均値

自動モードでは、以下の条件が発生すると、PID_Compact、バージョン 1.1 以降は「無効」モードに設定されます。

- 新しい平均値 >= 1.5 x 古い平均値
- 新しい平均値 <= 0.5 x 古い平均値

自動モードでのコントローラ同調時に、以下の条件が発生すると、PID_Compact 1.0 は「無効」動作モードに設定されます。

- 新しい平均値 >= 1.1 x 古い平均値
- 新しい平均値 <= 0.9 x 古い平均値
- 現在のサンプリング時間 >= 1.5 x 現在の平均値
- 現在のサンプリング時間 <= 0.5 x 現在の平均値

PID アルゴリズムのサンプリング時間

コントロールされたシステムは、出力値の変更に応答するために、一定量の時間を必要とします。 このため、すべてのサイクルごとに出力値を計算することはお奨めしません。 PID アルゴリズムのサン

プリング時間は、出力の 2 つの計算の間の時間を表わします。 それは同調時に計算され、サイクルタイムの倍数に四捨五入されます。 PID_Compact の他のすべてのファンクションは、呼び出しの都度実行されます。

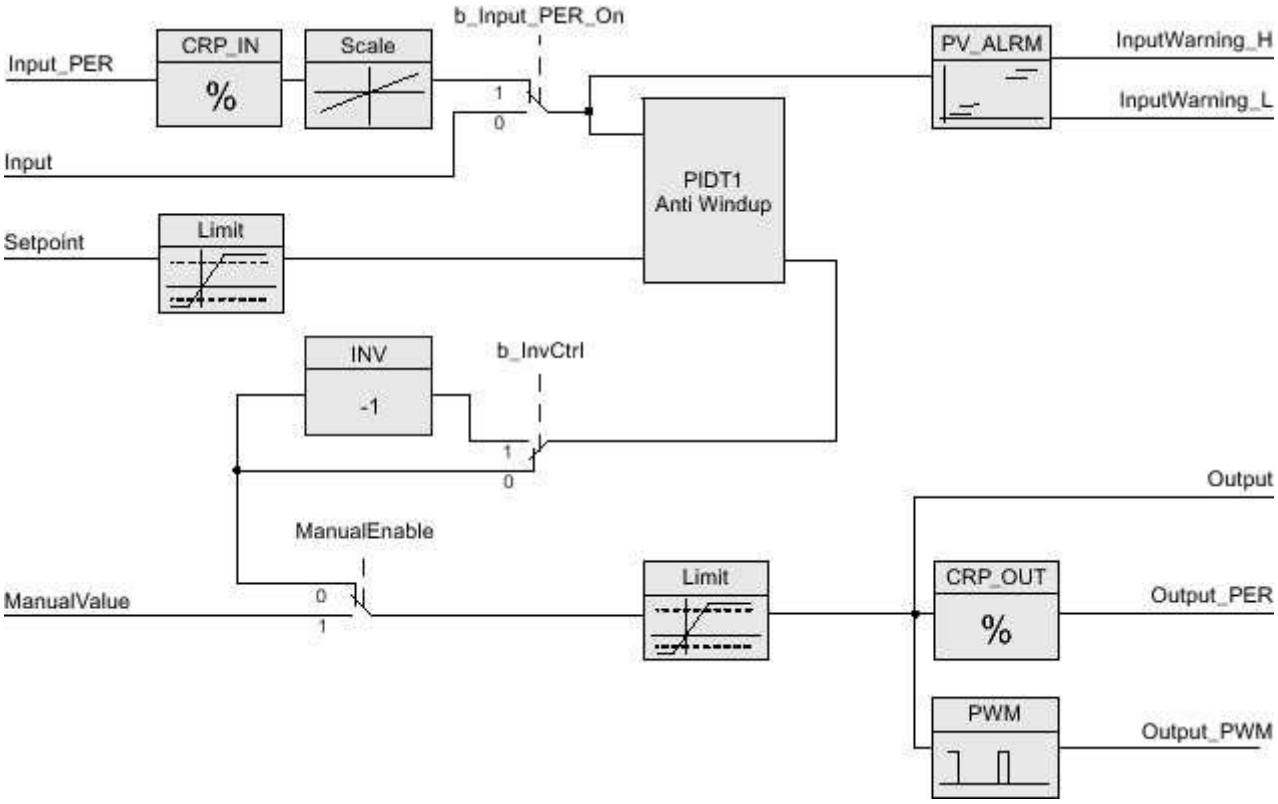
PID アルゴリズム

PID_Compact は、比例および微分動作のワインドアップ防止および重み付け付きの PIDT1 コントローラです。 出力値の計算には、次の数式が使用されます。

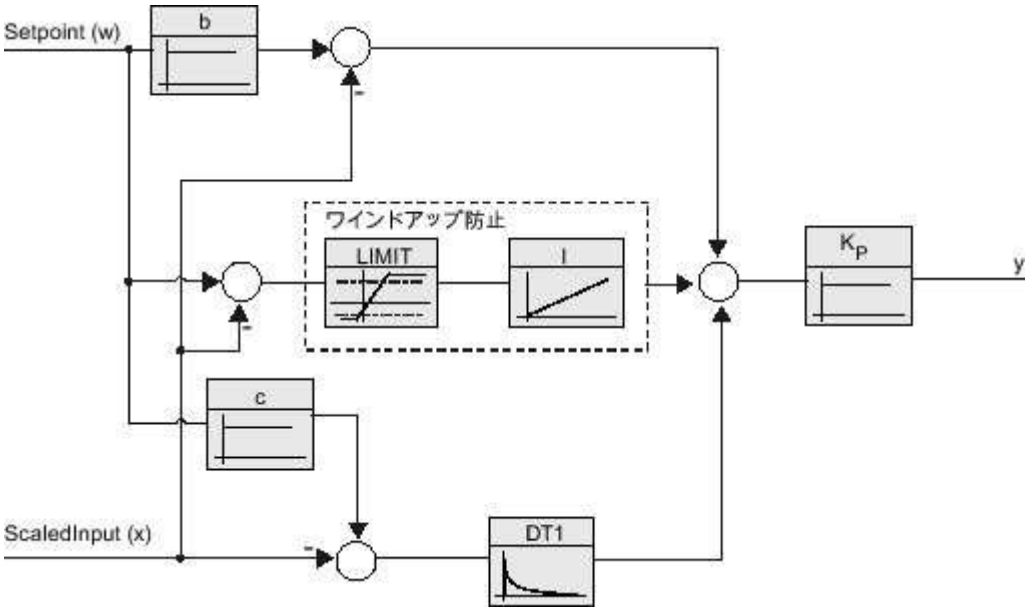
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

シンボル	説明
y	出力値
K_p	比例ゲイン
s	ラプラス演算子
B	比例動作の重み付け
w	セットポイント
x	プロセス値
T_I	積分動作時間
A	微分の遅延係数(T1 = a × T_D)
	微分動作時間
C	微分動作の重み付け

PID_Compact のブロックダイアグラム



ワインドアップ防止付きの PIDT1 のブロックダイアグラム



エラーに対する応答

発生したエラーはパラメータ Error に出力され、PID_Compact が「無効」モードに切り替わります。Reset パラメータを使用して、エラーをリセットしてください。

制御論理

出力値を上げるのは通常、プロセス値を上げるために行われます。 これは、標準制御ロジックと呼ばれます。 冷却および放電制御システムでは、制御ロジックを反転させる必要があります。PID_Com-

pact は、負の比例ゲインでは機能しません。 InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。 制御ロジックは、プリチューニングと微調整時にも考慮されます。

PID_Compact V1 の入力パラメータ

パラメータ	データタイプ	デフォルト	説明
Setpoint	REAL	0.0	自動モードでの PID コントローラのセットポイント
Input	REAL	0.0	ユーザープログラムの変数が、プロセス値のソースとして使用されます。 パラメータ Input を使用する場合は、sPid_Cmpt.b_Input_PER_On = FALSE を設定する必要があります。
Input_PER	WORD	W#16#0	プロセス値のソースとしてのアナログ入力 パラメータ Input_PER を使用する場合は、sPid_Cmpt.b_Input_PER_On = TRUE を設定する必要があります。
ManualEnable	BOOL	FALSE	• FALSE -> TRUE エッジでは「手動モード」 が選択され、State = 4, sRet.i_Mode はそのまま変わりません。 • TRUE -> FALSE エッジでは、直近の有効な動作モード State =sRet.i_Mode が選択されます。 sRet.i_Mode の変更は、ManualEnable = TRUE 時には効果を持ちません。 sRet.i_Mode の変更は、ManualEnable での TRUE -> FALSE エッジ時に考慮されます。 **PID_Compact V1.2 und PID_Compact V1.0** CPU の開始時に ManualEnable = TRUE の場合、PID_Compact が手動モードで開始されます。 ManualEnable での立ち上がりエッジ(FALSE -> TRUE)は不要です。 **PID_Compact V1.1** CPU の開始時に、ManualEnable での立ち上がりエッジ(FALSE->TRUE)の場合のみ、PID_Compact が手動モードに切り替わります。 立ち上がりエッジがない場合は、ManualEnable が FALSE であった最後の動作モードで PID_Compact が開始されます。
ManualValue	REAL	0.0	手動値 この値は、手動モードで、出力値として使用されます。
Reset	BOOL	FALSE	Reset パラメータは、コントローラを再起動します。

PID_Compact V1 の出力パラメータ

Parameter	データタイプ	デフォルト	説明
ScaledInput	REAL	0.0	スケーリングされたプロセス値の出力
出力「Output」、「Output_PER」、および「Output_PWM」を並行して使用できます。			
Output	REAL	0.0	REAL フォーマットの出力値
Output_PER	WORD	W#16#0	アナログ出力値
Output_PWM	BOOL	FALSE	パルス幅調整された出力値 出力値は、最小オン/オフ回数によって形成されます。
SetpointLimit_H	BOOL	FALSE	SetpointLimit_H = TRUE の場合、セットポイント絶対上限値に達しています。 CPU のセットポイントは、設定済みのセットポイント絶対上限値を超えることはできません。 設定済みのプロセス値絶対上限値は、セットポイント上限値のデフォルト値です。 sPid_Cmpt.r_Sp_Hlm にプロセス値の制限範囲内の値を設定すると、この値がセットポイント上限値として使用されます。
SetpointLimit_L	BOOL	FALSE	SetpointLimit_L = TRUE の場合、セットポイント絶対下限値に到達済みです。 CPU では、セットポイントは、設定済みのセットポイント絶対下限値未満であってはいけません。 設定済みのプロセス値絶対下限値は、セットポイント下限値のデフォルト設定です。 sPid_Cmpt.r_Sp_Llm にプロセス値の制限範囲内の値を設定すると、この値がセットポイント下限値として使用されます。
InputWarn-ing_H	BOOL	FALSE	InputWarning_H = TRUE の場合、プロセス値が警告上限値に到達済みか、警告上限値を超えています。
InputWarning_L	BOOL	FALSE	InputWarning_L = TRUE の場合、プロセス値が警告下限値に到達済みか、警告下限値未満になっています。
State	INT	0	State パラメータは、PID コントローラの現在の動作モードを 示します。 動作モードを変更する場合は、変数 sRet.i_Mode を使用します。 • State = 0: 無効 • State = 1: 事前同調 • State = 2: 微調整 • State = 3: 自動モード • State = 4: 手動モード
Error	DWORD	W#16#0	Error パラメータは、エラーメッセージを示します。 Error = 0000: 保留エラーなし

PID_Compact V1 の静的タグ

リストにないタグを変更してはいけません。これらは、システム内部の目的のためだけに使用されます。

タグ	データタイプ	デフォルト	説明
sb_GetCycleTime	BOOL	TRUE	sb_GetCycleTime = TRUE の場合、サイクルタイムの自動測定が開始されます。CycleTime.StartEstimation = FALSE の場合、すぐに、測定が完了します。
sb_EnCyclEstimation	BOOL	TRUE	sb_EnCyclEstimation = TRUE の場合、サンプリング時間 PID_Compact が計算されます。
sb_EnCyclMonitoring	BOOL	TRUE	sb_EnCyclMonitoring = FALSE の場合、PID_Compact はモニタされません。 サンプリング時間以内に PID_Compact を実行できないと、0800 エラーは出力されず、PID_Compact は「無効」モードに切り替わりません。
sb_RunModeByStartup	BOOL	TRUE	CPU のリスタート後にモードを有効化 sb_RunModeByStartup = FALSE の場合、コントローラは、CPU スタートアップ後、無効なままです。 CPU スタートアップ後に sb_RunModeByStartup = TRUE の場合、コントローラは直近の有効な動作モードに戻ります。
si_Unit	INT	0	プロセス値およびセットアポイントの測定単位(たとえば、℃または ºF)
si_Type	INT	0	プロセス値およびセットポイントの物理量 (たとえば、温度)
sd_Warning	DWORD	DW#16#0	変数 sd_warning には、リセット後または動作モードの最後の変更後に生成された警告が表示されます。
sBackUp.r_Gain	REAL	1.0	比例ゲインの保存エリア sPid_Cmpt.b_LoadBackUp = TRUE で、sBackUp 構造体から値を再読み込みできます。
sBackUp.r_Ti	REAL	20.0	積分動作時間[s]の保存エリア
sBackUp.r_Td	REAL	0.0	微分動作時間[s]の保存エリア
sBackUp.r_A	REAL	0.0	微分の遅延係数の保存エリア
sBackUp.r_B	REAL	0.0	比例動作重み係数の保存エリア
sBackUp.r_C	REAL	0.0	微分操作重み係数の保存エリア
sBackUp.r_Cycle	REAL	1.0	PID アルゴリズムのサンプリング時間の保存エリア

sPid_Calc.r_Cycle	REAL	0.1	PID_Compact 命令のサンプリング時間 r_Cycle が自動的に測定されます。通常、呼び出し OB のサイクルタイムと同じ値です。
sPid_Calc.b_RunIn	BOOL	FALSE	• b_RunIn = FALSE 無効なモードまたは手動モードで微調整が開始されると、プリチューニングが開始されます。 プリチューニングの要件が満たされない場合、PID_Compact は、b_RunIn = TRUE の場合と同様に応答します。 自動モードで微調整が開始されると、システムは既存の PID パラメータを使用してセットポイントを制御します。 この場合のみ、微調整が開始されます。プリチューニングが可能でない場合、PID_Compact は「無効」モードに切り替えます。 • b_RunIn = TRUE プリチューニングはスキップされます。PID_3Compact は、最小または最大出力値を使用してセットポイントに到達しようと試みます。この結果、オーバーシュートが増大する可能性があります。 微調整が自動的に開始します。 微調整後、b_RunIn が FALSE に設定されます。
sPid_Calc.b_CalcParamSUT	BOOL	FALSE	b_CalcParamSUT = TRUE の場合、プリチューニング用パラメータが再計算されます。 これによって、コントローラ同調を繰り返すことなく、パラメータ計算方法を変更できます。 計算後、b_CalcParamSUT が FALSE に設定されます。
sPid_Calc.b_CalcParamTIR	BOOL	FALSE	b_CalcParamTIR = TRUE の場合、微調整用パラメータが再計算されます。これによって、コントローラ同調を繰り返すことなく、パラメータ計算方法を変更できます。 計算後、b_CalcParamTIR が FALSE に設定されます。
sPid_Calc.i_CtrlTypeSUT	INT	0	プリチューニング時のパラメータの計算に使用される方法: • i_CtrlTypeSUT = 0: Chien、Hrones、および Reswick に従った PID • i_CtrlTypeSUT = 1: Chien、Hrones、および Reswick に従った PI
sPid_Calc.i_CtrlTypeTIR	INT	0	微調整時のパラメータの計算に使用される方法: • i_CtrlTypeTIR = 0: PID (自動)

			• i_CtrlTypeTIR = 1: PID (高速) • i_CtrlTypeTIR = 2: PID (低速) • i_CtrlTypeTIR = 3: Ziegler-Nichols PID • i_CtrlTypeTIR = 4: Ziegler-Nichols PI • i_CtrlTypeTIR = 5: Ziegler-Nichols P
sPid_Calc.r_Progress	REAL	0.0	同調の進捗状況(パーセンテージ(0.0 ~ 100.0)で表示)
sPid_Cmpt.r_Sp_Hlm	REAL	+3.402822e+38	セットポイントの上限値 プロセス値制限範囲外の sPid_Cmpt.r_Sp_Hlm を設定すると、設定されたプロセス値絶対上限値がセットポイント上限値として使用されます。 プロセス値制限範囲内の sPid_Cmpt.r_Sp_Hlm を設定すると、この値がセットポイント上限値として使用されます。
sPid_Cmpt.r_Sp_Llm	REAL	-3.402822e+38	セットポイントの下限値 プロセス値制限範囲外の sPid_Cmpt.r_Sp_Llm を設定すると、設定されたプロセス値絶対下限値がセットポイント下限値として使用されます。 プロセス値制限範囲内の sPid_Cmpt.r_Sp_Llm を設定すると、この値がセットポイント下限値として使用されます。
sPid_Cmpt.r_Pv_Norm_IN_1	REAL	0.0	低 Input_PER のスケーリング Input_PER は、sPid_Cmpt 構造体の 2 つの値ペア r_Pv_Norm_OUT_1、r_Pv_Norm_IN_1 および r_Pv_Norm_OUT_2、r_Pv_Norm_IN_2 に基づいてパーセントに変換されます。
sPid_Cmpt.r_Pv_Norm_IN_2	REAL	27648.0	高 Input_PER のスケーリング Input_PER は、sPid_Cmpt 構造体の 2 つの値ペア r_Pv_Norm_OUT_1、r_Pv_Norm_IN_1 および r_Pv_Norm_OUT_2、r_Pv_Norm_IN_2 に基づいてパーセントに変換されます。
sPid_Cmpt.r_Pv_Norm_OUT_1	REAL	0.0	スケーリングされた Low プロセス値 Input_PER は、sPid_Cmpt 構造体の 2 つの値ペア r_Pv_Norm_OUT_1、r_Pv_Norm_IN_1 および r_Pv_Norm_OUT_2、r_Pv_Norm_IN_2 に基づいてパーセントに変換されます。
sPid_Cmpt.r_Pv_Norm_OUT_2	REAL	100.0	スケーリングされた High プロセス値 Input_PER は、sPid_Cmpt 構造体の 2 つの値ペア r_Pv_Norm_OUT_1、r_Pv_Norm_IN_1 および

			r_Pv_Norm_OUT_2、r_Pv_Norm_IN_2 に基づいてパーセントに変換されます。
sPid_Cmpt.r_Lmn_Hlm	REAL	100.0	出力パラメータ「Output」の出力上限値
sPid_Cmpt.r_Lmn_Llm	REAL	0.0	出力パラメータ「Output」の出力下限値
sPid_Cmpt.b_Input_PER_On	BOOL	TRUE	b_Input_PER_On = TRUE の場合、パラメータ Input_PER が使用されます。 b_Input_PER_On = FALSE の場合、パラメータ Input が使用されます。
sPid_Cmpt.b_LoadBackUp	BOOL	FALSE	バックアップパラメータセットを有効にします。 最適化に失敗した場合は、このビットをセットすることにより、以前の PID パラメータを再度有効にすることができます。
sPid_Cmpt.b_InvCtrl	BOOL	FALSE	反転制御ロジック b_InvCtrl = TRUE の場合、立ち上がり制御偏差が出力値を低減します。
sPid_Cmpt.r_Lmn_Pwm_PPTm	REAL	0.0	パルス幅変調方式の最小オン時間(秒)が、以下に丸められます。 r_Lmn_Pwm_PPTm = r_Cycle または r_Lmn_Pwm_PPTm = n*r_Cycle
sPid_Cmpt.r_Lmn_Pwm_PBTm	REAL	0.0	パルス幅変調方式の最小オフ時間(秒)が、以下に丸められます。 r_Lmn_Pwm_PBTm = r_Cycle または r_Lmn_Pwm_PBTm = n*r_Cycle
sPid_Cmpt.r_Pv_Hlm	REAL	120.0	プロセス値の上限値 I/O 入力で、プロセス値が標準範囲よりも最大 18%大きくなる場合があります(範囲オーバー)。「プロセス値の上限値」オーバーのエラーは報告されません。 断線および短絡のみが認識され、PID_Compact が「無効」モードに切り替わります。 r_Pv_Hlm > r_Pv_Llm
sPid_Cmpt.r_Pv_Llm	REAL	0.0	プロセス値の下限値 r_Pv_Llm < r_Pv_Hlm
sPid_Cmpt.r_Pv_HWrn	REAL	+3.402822e+38	プロセス値の警告上限値 プロセス値制限範囲外の r_Pv_HWrn を設定すると、設定されたプロセス値絶対上限値が警告上限値として使用されます。 プロセス値制限範囲内の r_Pv_HWrn を設定すると、この値が警告上限値として使用されます。 r_Pv_HWrn > r_Pv_LWrn r_Pv_HWrn ≤ r_Pv_Hlm
sPid_Cmpt.r_Pv_LWrn	REAL	-3.402822e+38	プロセス値の警告下限値

			プロセス値制限範囲外の r_Pv_LWrn を設定すると、設定されたプロセス値絶対下限値が警告下限値として使用されます。 プロセス値制限範囲内の r_Pv_LWrn を設定すると、この値が警告下限値として使用されます。 r_Pv_LWrn < r_Pv_HWrn r_Pv_LWrn ≥ r_Pv_LWrn
sParamCalc.i_Event_SUT	INT	0	変数 i_Event_SUT は、「プリチューニング」の現在のフェースを示します。
sParamCalc.i_Event_TIR	INT	0	変数 i_Event_TIR は、「微調整」の現在のフェーズを示します。
sRet.i_Mode	INT	0	動作モードはエッジトリガによって変更されます。 以下への変更では、以下の動作モードが有効になります。 • i_Mode = 0: 「無効」(コントローラ停止) • i_Mode = 1:「プリチューニング」モード • i_Mode = 2: 「微調整」モード • i_Mode = 3: 「自動モード」 • i_Mode = 4: 「手動モード」 i_Mode は保持されます。
sRet.r_Ctrl_Gain	REAL	1.0	有効な比例ゲイン Gain は保持されます。
sRet.r_Ctrl_Ti	REAL	20.0	• r_Ctrl_Ti > 0.0: 有効な積分動作時間 • r_Ctrl_Ti = 0.0: 積分動作が無効です r_Ctrl_Ti は保持されます。
sRet.r_Ctrl_Td	REAL	0.0	• r_Ctrl_Td > 0.0: 有効な微分動作時間 • r_Ctrl_Td = 0.0: 微分動作が無効です r_Ctrl_Td は保持されます。
sRet.r_Ctrl_A	REAL	0.0	動作中の微分の遅延係数 r_Ctrl_A は保持されます。
sRet.r_Ctrl_B	REAL	0.0	有効な比例動作重み r_Ctrl_B は保持されます。
sRet.r_Ctrl_C	REAL	0.0	有効な微分動作重み r_Ctrl_C は保持されます。
sRet.r_Ctrl_Cycle	REAL	1.0	PID アルゴリズムの有効なサンプリング時間 r_Ctrl_Cycle がコントローラ同調時に計算され、r_Cycle の整数倍に丸められます。 r_Ctrl_Cycle は保持されます。

注記
PID コントローラの誤動作を防止するには、「無効」モードでこの表に記載されているタグを変更します。 変数「sRet.i_Mode」を「0」に設定して、「無効」モードが強制されます。

パラメータ State および sRet.i_Mode V1

パラメータの相互関係

State パラメータは、PID コントローラの現在の動作モードを示します。 State パラメータを変更することはできません。

動作モードを変更するには、sRet.i_Mode タグを変更する必要があります。 これは、新しい動作モードの値が sRet.i_Mode 内に存在するときにも適用されます。 最初に sRet.i_Mode = 0 を設定した後、sRet.i_Mode = 3 を設定します。 コントローラの現在の動作モードがこの変更をサポートする場合、State は、sRet.i_Mode の値に設定されます。

PID_Compact が自動的に動作モードを切り替えるときは、State != sRet.i_Mode が適用されます。

例:

- 成功した事前同調
  State = 3 で、sRet.i_Mode = 1
- エラー
  State = 0 で、sRet.i_Mode は同じ値のまま(たとえば、sRet.i_Mode = 3)
- ManualEnalbe = TRUE
  State = 4 で、sRet.i_Mode は直前の値のまま(たとえば、sRet.i_Mode = 3)

注記

i_Mode = 0 で自動モードを終了しないで、成功した微調整を繰り返すこともできます。

1 つのサイクルについて、sRet.i_Mode を 9999 などの不正な値に設定すると、State に対して効果を持ちません。 次のサイクルで、Mode = 2 を設定します。 最初に[無効]モードに切り替えなくても、sRet.i_Mode を変更することができます。

値の意味

State / sRet.i_Mode	動作モードの説明
0	無効 コントローラがスイッチオフされます。 コントローラが、事前同調を行う前に、[無効]モードでした。 PID コントローラは、実行時にエラーが発生するか、コミッショニングウィンドウで[コントローラの無効化]アイコンをクリックすると、[無効]モードに切り替わります。
1	事前同調 事前同調では、出力値のジャンプに対するプロセス応答が決定され、変曲点が検索されます。 最適化 PID パラメータは、コントロールされるシステムの最大上昇率とデッドタイムの関数として計算されます。 事前同調の必要条件 • コントロールが無効モードまたは手動オードであること • ManualEnable = FALSE

	• プロセス値は、セットポイントに近すぎる値であってはいけません。 |Setpoint - Input| > 0.3 * |sPid_Cmpt.r_Pv_Hlm - sPid_Cmpt.r_Pv_Llm|および |Setpoint - Input| > 0.5 * |Setpoint| • セットポイントは、同調時には変更できません。 プロセス値の安定性が高ければ高いほど、PID パラメータを計算し、結果の正確性を増すことが容易になります。 プロセス値のノイズは、プロセス値の上昇率がノイズに比べて大幅に高い限り、許可されます。 PID パラメータは再計算され、再び有効化される前に、sPid_Cmpt.b_LoadBackUp でバックアップされます。 事前同調が成功すると自動モードに切り替わり、事前同調が失敗すると[無効]モードに切り替わります。 事前同調のフェーズは、 タグ i_Event_SUT V1 で示されます。
2	微調整 微調整は、プロセス値の一定の制限された振動を生成します。 PID パラメータは、この振動の振幅と周波数に基づいて最適化されます。 事前同調時と微調整時のプロセス応答間の相違が分析されます。 この分析の結果に基づいて、すべての PID パラメータが再計算されます。 微調整に基づく PID パラメータのほうが、通常、事前同調に基づく PID パラメータより優れたマスタコントロールと外乱動作を実現します。 PID_Compact は、自動的に、プロセス値のノイズよりも大きい振動を生成しようとします。 微調整は、プロセス値の安定性によって最小限の影響しか受けません。 PID パラメータは再計算され、再び有効化される前に、sPid_Cmpt.b_LoadBackUp でバックアップされます。 微調整の必要条件 • 外乱が予測されないこと。 • セットポイントとプロセス値が、設定済みの制限範囲内であること。 • セットポイントは、微調整時には変更できないこと。 • ManualEnable = FALSE • 自動(State = 3)、無効(State = 0)、または手動(State = 4)モードであること。 微調整は、開始されたときのモードに従って以下のように実行されます。 • 自動モード(State = 3) コントローラ同調を使用して既存の PID パラメータを改善する場合は、自動モードで微調整を開始します。 PID_Comact は、コントロールループが安定し、微調整の必要条件が満たされるまでは、既存の PID パラメータを使用して調整されます。 この場合のみ、微調整が開始されます。 • 無効モード(State = 0)または手動(State = 4)モード 事前同調の必要条件が満たされると、事前同調が開始されます。 コントロールループが安定し、微調整の必要条件が満たされるまでは、確立された PID パラメータが調整で使用されます。 この場合のみ、微調整が開始されます。 事前同調が可能でない場合は、PID_Compact が[無効]モードに切り替わります。 事前同調のプロセス値がすでにセットポイントに非常に近いか、sPid_Calc.b_RunIn = TRUE の場合は、最小または最大出力値によるセットポイントへの到達が試みられます。 これによって、オーバーシュートの増加が起きる場合があります。 [微調整]が正常終了した後、コントローラは[自動モード]に切り替わり、[微調整]が正常終了しなかった場合、[無効]モードに切り替わります。

	「微調整」のフェーズは、 タグ i_Event_TIR V1 で示されます。
3	自動モード 自動モードでは、PID_Compact は、指定されたパラメータに従って、コントロールされるシステムを訂正します。 以下の条件の 1 つが満たされた場合、コントローラは自動モードに変わります。 • 事前同調の正常終了 • 微調整の正常終了 • 変数 sRet.i_Mode が値 3 へ変更 CPU スタートアップの後、または STOP から RUN モードへの変更の後に、PID_Compact は、直近の有効な動作モードで開始されます。 PID_Compact を[無効]モードのままに保持するには、sb_RunModeByStartup = FALSE を設定してください。
4	手動モード 手動モードでは、手動出力値を ManualValue パラメータに指定します。 この動作モードは、sRet.i_Mode = 4 の場合、または ManualEnable の立ち上がりエッジ時に有効になります。 ManualEnable が TRUE に変わると、 State のみが変わります。sRet.i_Mode は、その現在の値を保持します。ManualEnable の立ち下がりエッジでは、PID_Compact は直前の動作モードに戻ります。 この自動モードへの変更はバンプレスです。

パラメータエラー V1

複数のエラーが同時に保留中の場合、エラーコードの値がバイナリ加算で表示されます。 たとえば、エラーコード 0003 の表示は、エラー 0001 と 0002 が同時に保留中であることを示します。

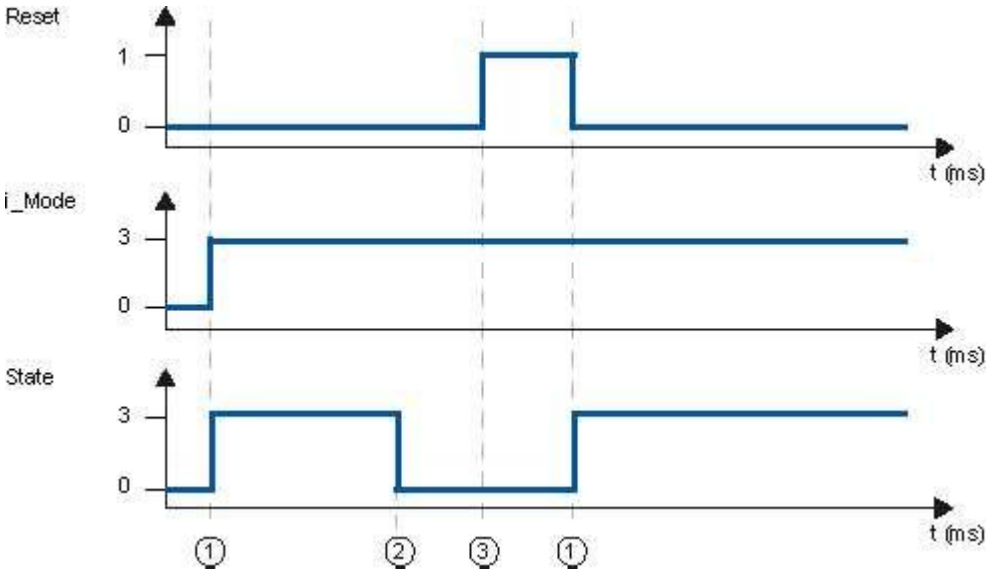
Error (DW#16#...)	説明
0000	エラーはありません。
0001	「Input」パラメータがプロセス値制限範囲外です。 • Input > sPid_Cmpt.r_Pv_Hlm または • Input < sPid_Cmpt.r_Pv_Llm エラーを取り除くまでは、アクチュエータを移動できません。
0002	「Input_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。
0004	微調整中のエラー プロセス値の振幅を維持できませんでした。
0008	プリチューニングの開始時のエラー プロセス値がセットポイントに近すぎます。 微調整を開始してください。
0010	セットポイントが、同調時に変更されました。
0020	プリチューニングを自動モードで、または微調整中に行うことはできません。
0080	出力制限値の設定が不正です。 出力値の制限値が正しく設定されていて、制御ロジックと一致しているかどうかをチェックします。
0100	同調時のエラーによって、パラメータが無効になりました。
0200	「Input」パラメータの値が無効です。 値が無効な番号フォーマットです。
0400	出力値の計算が失敗しました。 PID パラメータをチェックしてください。
0800	サンプリング時間エラー: PID_Compact が、サイクル割り込み OB のサンプリング時間内に呼び出されていません。
1000	「Setpoint」パラメータの値が無効です。 値が無効な番号フォーマットです。

パラメータ Reset V1

Reset = TRUE に対する応答は、PID_Compact 命令のバージョンに応じて異なります。

Reset に対する応答(PID_Compact V.1.1 以降)

Reset での立ち上がりはエラーおよび警告をリセットし、積分動作を解除します。 Reset での立ち下がりは、直近の有効な動作モードへの変更をトリガします。



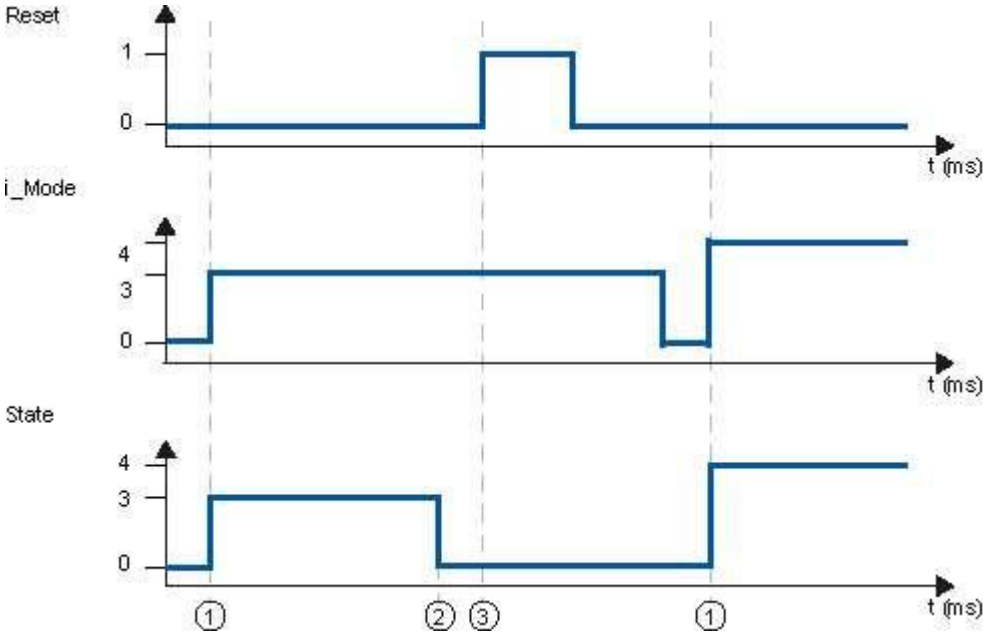
① 有効化

② Error

③ Reset

Reset に対する応答(PID_Compact V.1.0)

Reset での立ち上がりはエラーおよび警告をリセットし、積分動作を解除します。 コントローラは、i_Mode での次のエッジまでは、再度有効になりません。



① 有効化

② Error

③ Reset

タグ sd_warning V1

複数の警告が保留中の場合は、変数 sd_warning の値はバイナリ加算によって表示されます。 警告 0003 の表示は、たとえば、警告 0001 と警告 0002 が同時に保留中であることを示します。

sd_warning (DW#16#....)	説明
0000	保留中の警告がありません。
0001	事前同調時にインフレクションポイントが見つかりませんでした。
0002	微調整時に振動が増加しました。
0004	セットポイントが、設定された制限範囲外でした。
0008	選択された計算方法で､コントロールされるシステムの必要なプロパティの中に定義されなかったものがあります。 代わりに、「i_CtrlTypeTIR = 3」方法を使用して、PID パラメータが計算されました。
0010	ManualEnable = TRUE のために、動作モードを変更できませんでした。
0020	呼び出し OB のサイクルタイムが、PID アルゴリズムのサンプリング時間を制限します。 さらに短い OB サイクルタイムを使用して、結果を改善してください。
0040	プロセス値が、その警告制限値の 1 つを超えました。

以下の警告は、原因の対処が終わったらすぐに削除されます。

- 0004
- 0020
- 0040

他のすべての警告は、Reset での立ち上がりエッジで解除されます。

タグ i_Event_SUT V1

i_Event_SUT	名前	説明
0	SUT_INIT	事前同調の初期化
100	SUT_STDABW	標準偏差の計算
200	SUT_GET_POI	インフレクションポイントの検索
9900	SUT_IO	事前同調が成功
1	SUT_NIO	事前同調が失敗

タグ i_Event_TIR V1

i_Event_TIR	名前	説明
-100	TIR_FIRST_SUT	微調整が実行可能ではありません。 事前同調が最初に実行されます。
0	TIR_INIT	微調整の初期化
200	TIR_STDABW	標準偏差の計算
300	TIR_RUN_IN	セットポイントへの到達の試み
400	TIR_CTRLN	既存の PID パラメータを使用した、セットポイントへの到達の試み (事前同調が成功した場合)
500	TIR_OSZIL	振動の決定とパラメータの計算
9900	TIR_IO	微調整が成功
1	TIR_NIO	微調整が失敗

PID_3Step

この章には下記に関する情報が記載されています：

- PID_3Step の新機能 (S7-1200, S7-1500)
- CPU および FW との互換性 (S7-1200, S7-1500)
- PID_3Step V2.x の CPU 処理時間およびメモリ要件 (S7-1200, S7-1500)
- PID_3Step V2 (S7-1200, S7-1500)
- PID_3Step V1 (S7-1200)

PID_3Step の新機能

PID_3Step V2.2

- **S7-1200 の使用**

  PID_3Step V2.2 以降では、V2 機能による命令をファームウェアバージョン 4.0 以上の S7-1200 でも使用できます。

PID_3Step V2.0

- **エラーに対する応答**

  ActivateRecoverMode = TRUE に対する応答が完全に見直されました。初期設定で、PID_3Step のフォールトトレラント性が大幅に高くなりました。

通知
システムが損傷することがあります。 初期設定を使用する場合、プロセス値の限界値を超えても PID_3Step が自動モードのままになります。 そのため、システムが損傷することがあります。 システムを損傷から守るために、エラーの場合に制御システムがどのように応答するかを設定する必要があります。

  コントローラの再起動や積分動作のクリアなしでエラーおよび警告を確認するには、ErrorAck 入力パラメータを使用します。

  動作モードを切り替えても、保留でなくなったエラーを確認することはできません。

- **動作モードの切り替え**

  ModeIN/OUT パラメータで動作モードを指定し、ModeActivate の立ち上がりエッジを使用して動作モードを開始します。 Retain.Mode タグは省略されています。

  移行時間の測定が GetTransitTime.Start ではなく、Mode = 6 および ModeActivate の立ち上がりエッジでしか開始できなくなりました。

- **マルチインスタンス機能**

  PID_3Step をマルチインスタンス DB として呼び出すことができます。 この場合、テクノロジーオブジェクトは作成されず、パラメータ割り当てインターフェースまたはコミッショニングインターフェースは使用できません。 複数インスタンス DB で直接に PID_3Step のパラメータを割り当て、ウォッチテーブル経由でそれをコミッショニングする必要があります。

- **スタートアップ特性**

  Mode パラメータで指定された動作モードも、Reset の立ち下がりエッジおよび CPU コールドリスタート時に開始されます(RunModeByStartup = TRUE の場合)。

- **ENO 特性**

  ENO は動作モードに応じて設定されます。

  State = 0 の場合、ENO = FALSE です。

  State ≠ 0 の場合、ENO = TRUE です。

- **手動モード**

  Manual_UP および Manual_DN 入力パラメータがエッジトリガパラメータとして機能しなくなりました。 エッジトリガの手動モードは、ManualUpInternal および ManualDnInternal タグを使用して引き続き使用可能です。

「エンドストップ信号なしの手動モード」(Mode = 10)では、エンドストップ信号 Actuator_H および Actuator_L は有効になっていても無視されます。

- **PID パラメータの既定値**

  以下の初期設定が変更されています。
  - 比例動作の重み付け (PWeighting)が 0.0 から 1.0 に
  - 微分動作の重み付け(DWeighting)が 0.0 から 1.0 に
  - 微分の遅延係数(TdFiltRatio)が 0.0 から 0.2 に

- **モータ移行時間の制限**

  Config.VirtualActuatorLimit タグで、アクチュエータが一方向に移動するモータ移行時間の最大パーセンテージを設定します。

- **同調時のセットポイント値の指定**

  CancelTuningLevel タグで、同調時のセットポイントの許容変動を設定します。

- **外乱変数オン**

  Disturbance パラメータで外乱変数をオンにすることができます。

- **トラブルシューティング**

  エンドストップ信号が有効になっていない場合(ActuatorEndStopOn = FALSE)、ScaledFeedback は Actuator_H または Actuator_L なしで決定されます。

PID_3Step V1.1

- **CPU スタートアップ時の手動モード**

  CPU の開始時に ManualEnable = TRUE の場合、PID_3Step が手動モードで開始されます。 ManualEnable の立ち上がりエッジは必要ありません。

- **エラーに対する応答**

  ActivateRecoverMode タグが手動モードでもはや有効ではありません。

- **トラブルシューティング**

  Progress タグは、同調または移行時間の測定が正常に行われるとリセットされます。

CPU および FW との互換性

下の表に、PID_3Step のどのバージョンがどの CPU で使用できるかを示します。

CPU	FW	PID_3Step
S7-1200	≥ V4.x	V2.2 V1.1
S7-1200	≥ V3.x	V1.1 V1.0
S7-1200	≥ V2.x	V1.1 V1.0
S7-1200	≥ V1.x	-
S7-1500	≥ V1.5	V2.2 V2.1 V2.0
S7-1500	≥ V1.1	V2.1 V2.0
S7-1500	≥ V1.0	V2.0

PID_3Step V2.x の CPU 処理時間およびメモリ要件

CPU 処理時間

バージョン V2.0 以降の PID_3Step テクノロジーオブジェクトの通常の CPU 処理時間(CPU タイプにより異なる)。

CPU	代表値 PID_3Step V2.x の CPU 処理時間
CPU 1211C ≥ V4.0	410 µs
CPU 1215C ≥ V4.0	410 µs
CPU 1217C ≥ V4.0	410 µs
CPU 1505S ≥ V1.0	50 µs
CPU 1510SP-1 PN ≥ V1.6	120 µs
CPU 1511-1 PN ≥ V1.5	120 µs
CPU 1512SP-1 PN ≥ V1.6	120 µs
CPU 1516-3 PN/DP ≥ V1.5	65 µs
CPU 1518-4 PN/DP ≥ V1.5	5 µs

メモリ要件

バージョン V2.0 以降の PID_3Step テクノロジーオブジェクトのインスタンス DB のメモリ要件。

	PID_3Step V2.x のインスタンス DB のメモリ要件
ロードメモリ要件	約 15000 バイト
合計ワークメモリ要件	1040 バイト
保持ワークメモリ要件	60 バイト

PID_3Step V2

この章には下記に関する情報が記載されています：

PID_3Step V2 の説明

説明

PID_3Step 命令を使用して、積分動作を行うバルブまたはアクチュエータ用の自己同調機能を使用して PID コントローラを設定します。

次の動作モードが使用可能です。

- 無効
- プリチューニング
- 微調整
- 自動モード
- 手動モード
- 代替出力値アプローチ
- 移行時間の測定
- エラーモニタリング
- エラーモニタリング付きの代替出力値アプローチ
- エンドストップ信号なしの手動モード

動作モードの詳細については、State パラメータを参照してください。

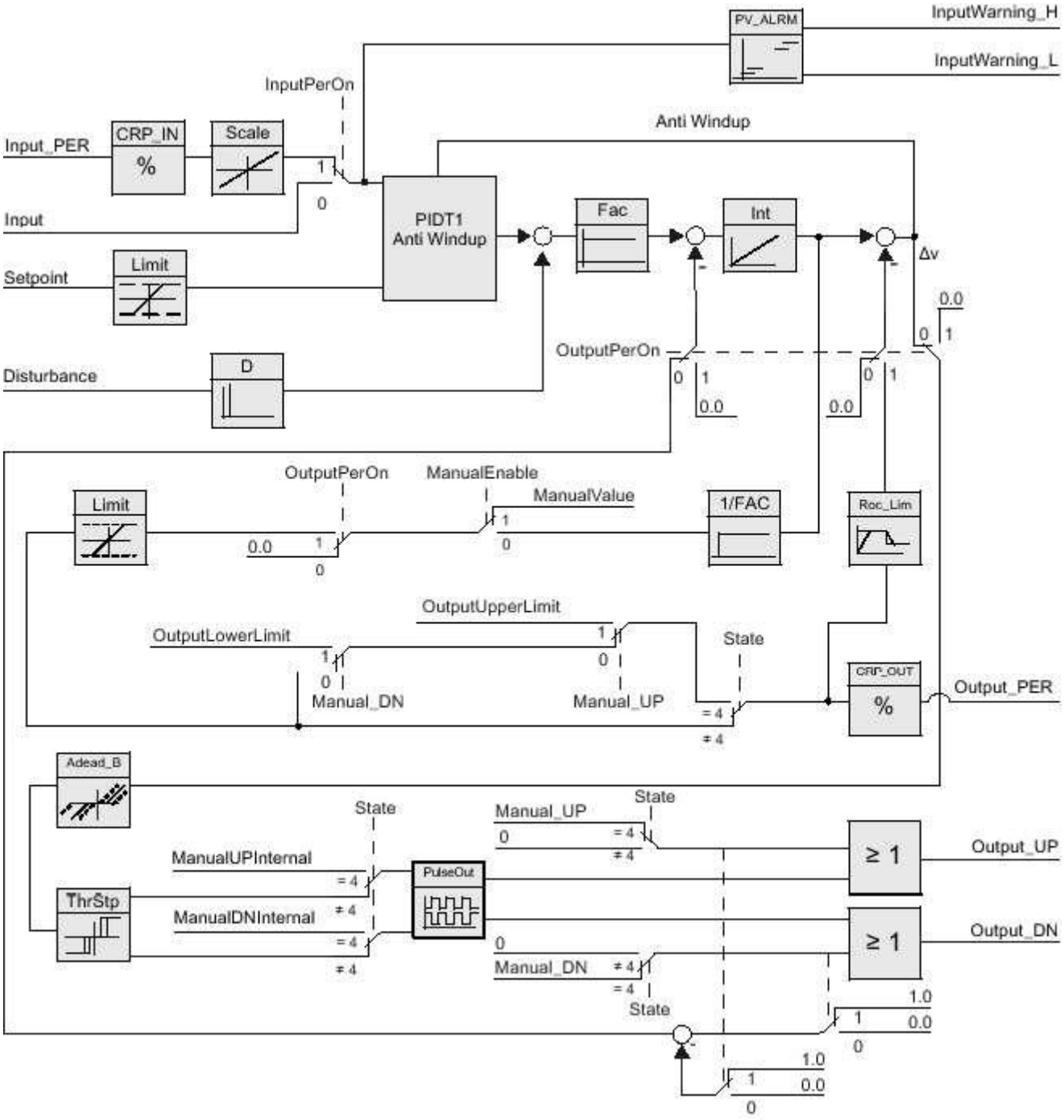
PID アルゴリズム

PID_3Step は、比例および微分動作のワインドアップ防止および重み付け付きの PIDT1 コントローラです。 PID アルゴリズムは、以下の等式に従って動作します。

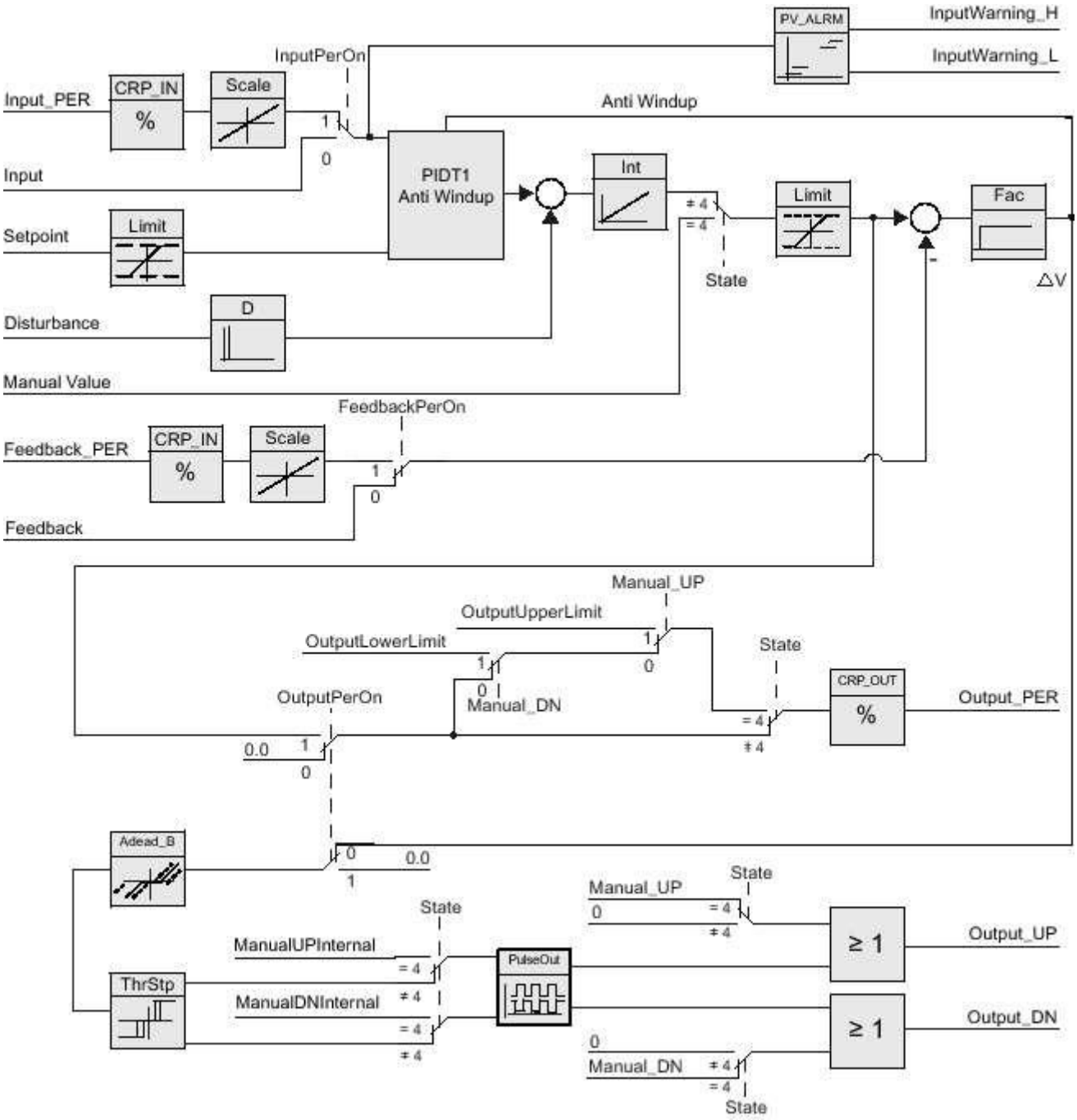
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

シンボル	説明
Δy	PID アルゴリズムの出力値
K_p	比例ゲイン
s	ラプラス演算子
B	比例動作の重み付け
w	セットポイント
x	プロセス値
T_I	積分動作時間
T_D	微分動作時間
A	微分の遅延係数(微分の遅延 T1 = a × T_D)
C	微分動作の重み付け

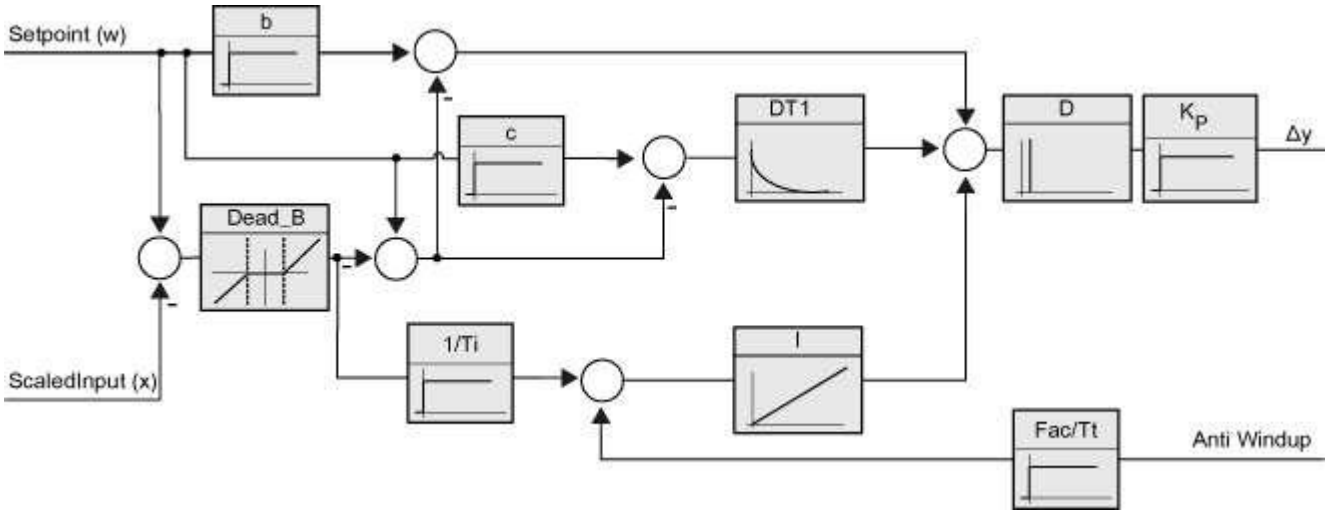
位置フィードバックなしのブロックダイアグラム



位置フィードバック付きのブロックダイアグラム



ワインドアップ防止付きの PIDT1 のブロックダイアグラム



呼び出し

PID_3Step は、周期割り込み DB の一定のタイムスケールで呼び出されます。

マルチインスタンス DB として PID_3Step を呼び出すと、テクノロジーオブジェクトは作成されません。パラメータ割り当てインターフェースおよびコミッショニングインターフェースは使用できません。 複数インスタンス DB で直接に PID_3Step のパラメータを割り当て、ウォッチテーブル経由でそれをコミッショニングする必要があります。

デバイスへのダウンロード

保持タグの現在値は、PID_3Step を完全にダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

スタートアップ

CPU のスタートアップ時に、PID_3Step が ModeIN/OUT パラメータに保存された動作モードで開始されます。 PID_3Step を「無効」モードのままにするには、RunModeByStartup = FALSE を設定してください。

エラーに対する応答

自動モード時およびコミッショニング時は、エラーに対する応答は ErrorBehaviour および ActivateRecoverMode タグによって決まります。 手動モードでは、応答は ErrorBehaviour および ActivateRecoverMode には依存しません。 ActivateRecoverMode = TRUE の場合、応答はさらに発生したエラーによっても異なります。

ErrorBehaviour	ActivateRecoverMode	コフィグレーションエディタ > アクチュエータの設定 > Output を以下に設定	応答
FALSE	FALSE	現在の出力値	「無効」モード(State = 0)に切り替え アクチュエータは現在の位置にとどまります。
FALSE	TRUE	エラー保留中の現在の出力値	「エラーモニタ」モード(State = 7)に切り替え

			アクチュエータはエラーが保留されている間、現在の位置にとどまります。
TRUE	FALSE	代替出力値	「代替出力値アプローチ」モード(State = 5)に切り替え アクチュエータが設定された代替出力値に移行します。 「無効」モード(State = 0)に切り替え アクチュエータは現在の位置にとどまります。
TRUE	TRUE	エラーが保留中の間、代替出力値	「エラーモニタ付きの代替出力値アプローチ」モード(State = 8)に切り替え アクチュエータが設定された代替出力値に移行します。 「エラーモニタ」モード(State = 7)に切り替え

手動モードでは、以下のエラーが発生しない限り、PID_3Step は ManualValue を出力値として使用します。

- 2000h: 「Feedback_PER」パラメータの値が無効です。
- 4000h: 「Feedback」パラメータの値が無効です。
- 8000h: デジタル位置フィードバック中のエラーです。

アクチュエータの位置は、Manual_UP および Manual_DN でのみ変更できます。ManualValue では変更できません。

Error パラメータは、このサイクルでエラーが発生したかどうかを示します。 ErrorBits パラメータは、どのエラーが発生したかを示します。ErrorBits は、Reset または ErrorAck の立ち上がりエッジでリセットされます。

PID_3Step V2 の動作モード

プロセス値制限値のモニタリング

Config.InputUpperLimit および Config.InputLowerLimit 変数で、プロセス値の上限値と下限値を指定します。 プロセス値がこの制限値を超えている場合、エラーが発生します(ErrorBits = 0001h)。

Config.InputUpperWarning および Config.InputLowerWarning 変数で、プロセス値の警告上限値と警告下限値を指定します。 プロセス値がこの警告制限値を超えている場合、警告が発生し(Warning = 0040h)、InputWarning_H または InputWarning_L 出力パラメータが TRUE に変わります。

セットポイントの制限

Config.SetpointUpperLimit および Config.SetpointLowerLimit 変数で、セットポイントの上限値と下限値を指定します。PID_3Step は自動的にセットポイントをプロセス値の制限値に制限します。 セットポイントを小さい範囲に制限することができます。PID_3Step は、この範囲がプロセス制限値の範囲内であるかどうかをチェックします。 セットポイントがこの制限値を超えると、上限値または下限値がセットポイントとして使用され、出力パラメータ SetpointLimit_H または SetpointLimit_L が TRUE に設定されます。

セットポイントは、すべての動作モードで、制限されます。

出力値の制限

Config.OutputUpperLimit および Config.OutputLowerLimit 変数で、出力値の上限値と下限値を指定します。 出力制限値は、「エンドストップ下限」および「エンドストップ上限」内であることが必要です。

- エンドストップ上限: Config.FeedbackScaling.UpperPointOut
- エンドストップ下限: Config.FeedbackScaling.LowerPointOut

ルール:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

「エンドストップ上限」および「エンドストップ下限」に有効な値は、次の条件によって決まります。

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	設定不可(0.0%)	設定不可(100.0%)
FALSE	TRUE	FALSE	-100.0%または 0.0%	0.0%または+100.0%
FALSE	TRUE	TRUE	-100.0%または 0.0%	0.0%または+100.0%
TRUE	FALSE	FALSE	設定不可(0.0%)	設定不可(100.0%)
TRUE	TRUE	FALSE	-100.0%または 0.0%	0.0%または+100.0%
TRUE	TRUE	TRUE	-100.0%または 0.0%	0.0%または+100.0%

OutputPerOn = FALSE および FeedbackOn = FALSE の場合、出力値を制限することはできません。したがって、Output_UP および Output_DN は Actuator_H = TRUE または Actuator_L = TRUE でリセ

ットされます。 エンドストップ信号も存在していない場合、Output_UP および Output_DN は Config.VirtualActuatorLimit × Retain.TransitTime/100 の移動時間後にリセットされます。

出力値は、100%で 27648 で、-100%で-27648 です。 PID_3Step は、バルブを完全に閉じることができる必要があります。

代替出力値

エラーが発生した場合、PID_3Step は代替出力値を出力し、アクチュエータを SavePosition タグで指定された安全な位置に移動することができます。 代替出力値は、出力制限値内であることが必要です。

信号妥当性のモニタリング

以下のパラメータの値は、使用時に有効性をモニタされます。

- Setpoint
- Input
- Input_PER
- Input_PER
- Feedback
- Feedback_PER
- Disturbance
- ManualValue
- SavePosition
- Output_PER

PID_3Step サンプリング時間のモニタ

理想的には、サンプリング時間は呼び出し OB のサイクルタイムと等値です。PID_3Step 命令は、2 つの呼び出し間の時間間隔を測定します。 これは、現在のサンプル時間です。 動作モードの変更ごと、また、初期スタートアップ時に、最初の 10 個のサンプリング時間からサンプリング時間の平均値が求められます。 現在のサンプリング時間とこの平均値の相違が大きすぎると、エラー(ErrorBits = 0800h)がトリガされます。

以下の場合、同調時にエラーが発生します。

- 新しい平均値 >= 1.1 x 古い平均値
- 新しい平均値 <= 0.9 x 古い平均値

以下の場合、自動モードでエラーが発生します。

- 新しい平均値 >= 1.5 x 古い平均値
- 新しい平均値 <= 0.5 x 古い平均値

サンプリング時間のモニタを無効にすると(CycleTime.EnMonitoring = FALSE)、PID_3Step を OB1 で呼び出すこともできます。 この場合、サンプリング時間のずれによる制御品質の低下を受け入れる必要があります。

PID アルゴリズムのサンプリング時間

コントロールされたシステムは、出力値の変更に応答するために、一定量の時間を必要とします。 このため、すべてのサイクルごとに出力値を計算することはお奨めしません。 PID アルゴリズムのサンプリング時間は、出力の 2 つの計算の間の時間を表します。 それは同調時に計算され、サイクルタイ

ムの倍数に四捨五入されます。 PID_3Step の他のすべてのファンクションは、呼び出しの都度実行されます。

モータ移行時間の測定

モータ移行時間は、モータがアクチュエータを閉状態から開状態に移行させるために必要な時間(秒)です。 アクチュエータは、最大時間 Config.VirtualActuatorLimit × Retain.TransitTime/100 の間、一方向に移動します。 PID_3Step は、最適なコントローラ結果を得るために、可能な限り正確なモータ移行時間を必要とします。 アクチュエータのマニュアルに記載されているデータは、このアクチュエータタイプの平均的な値です。 使用するアクチュエータによって、この値は異なる可能性があります。 モータ移行時間は、コミッショニング時に測定できます。 出力制限値は、モータ移行時間の測定中は考慮されません。 アクチュエータはエンドストップ上限または下限に移動できます。

制御論理

出力値を上げるのは通常、プロセス値を上げるために行われます。 これは、標準制御ロジックと呼ばれます。冷却および放電制御システムでは、制御ロジックを反転させる必要があります。PID_3Step は、負の比例ゲインでは機能しません。 InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。 制御ロジックは、プレチューニングと微調整時にも考慮されます。

PID_3Step V2 インターフェースの変更

下の表に、PID_3Step 命令インターフェースで何が変更されたかを示します。

PID_3Step V1	PID_3Step V2	変更
Input_PER	Input_PER	データタイプがワードから整数に
Feedback_PER	Feedback_PER	データタイプがワードから整数に
	Disturbance	新規作成
Manual_UP	Manual_UP	ファンクション
Manual_DN	Manual_DN	ファンクション
	ErrorAck	新規作成
	ModeActivate	新規作成
Output_PER	Output_PER	データタイプがワードから整数に
	ManualUPInternal	新規作成
	ManualDNInternal	新規作成
	CancelTuningLevel	新規作成
	VirtualActuatorLImit	新規作成
Config.Loadbackup	Loadbackup	名前の変更
Config.TransitTime	Retain.TransitTime	名前の変更と保持性の追加
GetTransit-Time.Start		Mode および ModeActivate で置換
SUT.CalculateSUT-Params	SUT.CalculateParams	名前の変更
SUT.TuneRuleSUT	SUT.TuneRule	名前の変更
TIR.CalculateTIR-Params	TIR.CalculateParams	名前の変更
TIR.TuneRuleTIR	TIR.TuneRule	名前の変更
Retain.Mode	Mode	ファンクション IN-OUT パラメータの静的性の宣言

PID_3Step V2 の入力パラメータ

パラメータ	データタイプ	既定	説明
Setpoint	REAL	0.0	自動モードでの PID コントローラのセットポイント
Input	REAL	0.0	ユーザープログラムのタグが、プロセス値のソースとして使用されます。 Input パラメータを使用する場合は、Config.InputPerOn = FALSE を設定する必要があります。
Input_PER	INT	0	アナログ入力が、プロセス値のソースとして使用されます。 Input_PER パラメータを使用する場合は、Config.InputPerOn = TRUE を設定する必要があります。
Actuator_H	BOOL	FALSE	エンドストップ上限用のバルブのデジタル位置フィードバック。 Actuator_H = TRUE の場合、バルブはエンドストップ上限の位置にあり、この方向には既に動きません。
Actuator_L	BOOL	FALSE	エンドストップ下限用のバルブのデジタル位置フィードバック。 Actuator_L = TRUE の場合、バルブはエンドストップ下限の位置にあり、この方向には既に動きません。
Feedback	REAL	0.0	バルブの位置フィードバック。 Feedback パラメータを使用する場合は、Config.FeedbackPerOn = FALSE を設定する必要があります。
Feedback_PER	INT	0	位置のアナログ位置フィードバック Feedback_PER パラメータを使用する場合は、Config.FeedbackPerOn = TRUE を設定する必要があります。 Feedback_PER は、以下のタグに基づいてスケーリングされます。 • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
Disturbance	REAL	0.0	外乱タグまたは事前制御値
ManualEnable	BOOL	FALSE	• FALSE -> TRUE エッジでは「手動モード」が有効になり、State = 4、Mode はそのまま変わりません。

			ManualEnable = TRUE である限り、ModeActivate の立ち上がりエッジで動作モードを変更したり、コミッショニングダイアログを使用することはできません。 • TRUE -> FALSE エッジにより、Mode で指定された動作モードが有効になります。 ModeActivate を使用した動作モードの変更のみをお勧めします。
ManualValue	REAL	0.0	手動モードでは、バルブの絶対位置を指定します。ManualValue は、 Output_PER を使用するか、**または**位置フィードバックが使用できる場合のみ、評価されます。
Manual_UP	BOOL	FALSE	• Manual_UP = TRUE Output_PER または位置フィードバックを使用している場合でも、バルブは開きます。 エンドストップ上限に達している場合、バルブはそれ以上移動されません。 関連項目: Config.VirtualActuatorLimit • Manual_UP = FALSE Output_PER または位置フィードバックを使用している場合、バルブは ManualValue に移動します。 それ以外の場合、バルブはそれ以上移動しません。 Manual_UP および Manual_DN が同時に TRUE に設定されている場合、バルブは移動しません。
Manual_DN	BOOL	FALSE	• Manual_DN = TRUE Output_PER または位置フィードバックを使用している場合でも、バルブは閉じます。 エンドストップ下限に達している場合、バルブはそれ以上移動されません。 関連項目: Config.VirtualActuatorLimit • Manual_DN = FALSE Output_PER または位置フィードバックを使用している場合、バルブは ManualValue に移動します。 それ以外の場合、バルブはそれ以上移動しません。
ErrorAck	BOOL	FALSE	• FALSE -> TRUE エッジ ErrorBits および Warning がリセットされます。
Reset	BOOL	FALSE	コントローラを再起動します。 • FALSE -> TRUE エッジ ○「無効」モードに切り替え ○ ErrorBits および Warning がリセットされます。 ○ 積分動作がクリアされます。 (PID パラメータは保持されます) • Reset = TRUE である限り、PID_3Step は「無効」モードのままです(State = 0)。 • TRUE -> FALSE エッジ

			PID_3Step が Mode パラメータに保存された動作モードに切り替わります。
ModeActivate	BOOL	FALSE	• FALSE -> TRUE エッジ PID_3Step が Mode パラメータに保存された動作モードに切り替わります。

PID_3Step V2 の出力パラメータ

パラメータ	データタイプ	デフォルト	説明
ScaledInput	REAL	0.0	スケーリングされたプロセス値
ScaledFeedback	REAL	0.0	スケーリングされた位置フィードバック 位置フィードバックなしのアクチュエータの場合、ScaledFeedback で示されたアクチュエータの位置は極めて不正確です。この場合、ScaledFeedback は現在の位置のおおまかな評価にしか使用できません。
Output_UP	BOOL	FALSE	バルブを開くためのデジタル出力値。 Config.OutputPerOn = FALSE の場合、Output_UP パラメータが使用されます。
Output_DN	BOOL	FALSE	バルブを閉じるためのデジタル出力値。 Config.OutputPerOn = FALSE の場合、Output_DN パラメータが使用されます。
Output_PER	INT	0	アナログ出力値 Config.OutputPerOn = TRUE の場合、Output_PER が使用されます。
SetpointLimit_H	BOOL	FALSE	SetpointLimit_H = TRUE の場合、セットポイント絶対上限値に達します(Setpoint ≥ Config.SetpointUpperLimit)。 セットポイントは Config.SetpointUpperLimit に制限されます。
SetpointLimit_L	BOOL	FALSE	SetpointLimit_L = TRUE の場合、セットポイント絶対下限値に達します(Setpoint ≤ Config.SetpointLowerLimit)。 セットポイントは Config.SetpointLowerLimit に制限されます。
InputWarning_H	BOOL	FALSE	InputWarning_H = TRUE の場合、プロセス値が警告上限値に到達済みか、警告上限値を超えています。
InputWarning_L	BOOL	FALSE	InputWarning_L = TRUE の場合、プロセス値が警告下限値に到達済みか、警告下限値未満になっています。
State	INT	0	State パラメータは、PID コントローラの現在の動作モードを 示します。 入力パラメータ Mode および ModeActivate の立ち上がりエッジを使用して、動作モードを変更することができます。 • State = 0: 無効 • State = 1: プリチューニング

			• State = 2: 微調整 • State = 3: 自動モード • State = 4: 手動モード • State = 5: 代替出力値アプローチ • State = 6: 移行時間の測定 • State = 7: エラーモニタリング • State = 8: エラーモニタリング付きの代替出力値アプローチ • State = 10: エンドストップ信号なしの手動モード
Error	BOOL	FALSE	Error = TRUE の場合、少なくとも 1 つのエラーがこのサイクルで保留中です。
ErrorBits	DWORD	DW#16#0	ErrorBits パラメータは、どのエラーメッセージが保留中かを示します。ErrorBits は保持され、Reset または ErrorAck の立ち上がりエッジでリセットされます。

PID_3Step V2 の IN-OUT パラメータ

パラメータ	データタイプ	デフォルト	説明
Mode	INT	4	Mode パラメータで、PID_3Step の切り替え先である動作モードを指定します。オプションを以下に示します。 • Mode = 0: 無効 • Mode = 1: プリチューニング • Mode = 2: 微調整 • Mode = 3: 自動モード • Mode = 4: 手動モード • Mode = 6: 移行時間の測定 • Mode = 10: エンドストップ信号なしの手動モード 動作モードは以下によって有効になります。 • ModeActivate の立ち上がりエッジ • Reset の立ち下がりエッジ • ManualEnable の立ち下がりエッジ • RunModeByStartup = TRUE の場合は CPU のコールドリスタート Mode は保持されます。 動作モードの詳細については、パラメータ State と Mode V2 を参照してください。

PID_3Step V2 の静的タグ

リストにないタグを変更してはいけません。 これらは、システム内部の目的のためだけに使用されます。

タグ	データタイプ	既定	説明
ManualUpInternal	BOOL	FALSE	手動モードでは、すべての立ち上がりエッジごとに、全制御範囲の 5%のみバルブが開くか、最小モータ移行時間の間のみバルブが開きます。ManualUpInternal は、Output_PER または位置フィードバックを使用しない場合のみ、評価されます。 このタグは、コミッショニングダイアログで使用します。
ManualDnInternal	BOOL	FALSE	手動モードでは、すべての立ち上がりエッジごとに、全制御範囲の 5%のみバルブが閉じるか、または最小モータ移行時間のみバルブが閉じます。ManualDnInternal は、Output_PER または位置フィードバックを使用しない場合のみ評価されます。 このタグは、コミッショニングダイアログで使用します。
ActivateRecoverMode	BOOL	TRUE	ActivateRecoverMode V2 タグは、エラーに対する応答を決定します。
RunModeByStartup	BOOL	TRUE	CPU のリスタート後に、Mode パラメータの動作モードを有効にします。 RunModeByStartup = TRUE の場合、PID_3Step は CPU のリスタート後に、Mode パラメータに保存されている動作モードで開始されます。 RunModeByStartup = FALSE の場合、PID_3Step は CPU のリスタート後も「無効」モードのままです。
LoadBackUp	BOOL	FALSE	LoadBackUp = TRUE の場合、最後の PID パラメータセットが再読み込みされます。 このセットは、最後の同調前に保存されたものです。LoadBackUp は自動的に FALSE に再セットされます。
PhysicalUnit	INT	0	プロセス値およびセットアポイントの測定単位(たとえば、℃または ºF)
PhysicalQuantity	INT	0	プロセス値およびセットポイントの物理量(たとえば、温度)
ErrorBehaviour	BOOL	FALSE	ErrorBehaviour = FALSE でエラーが発生すると、バルブはその現在位置に留まり、コントローラは「無効」モードまたは「エラーモニタ」モードに直接に切り替わります。 ErrorBehaviour = TRUE でエラーが発生すると、アクチュエータが代替出力値まで移動した後にのみ、「無効」モードまたは「エラーモニタ」モードに切り替わります。 以下のエラーが発生すると、バルブを設定した代替出力値まで移動できません。

			• 2000h: 「Feedback_PER」パラメータの値が無効です。 • 4000h: 「Feedback」パラメータの値が無効です。 • 8000h: デジタル位置フィードバック中のエラーです。 • 20000h: SavePosition タグの値が無効です。
Warning	DWORD	DW#16#0	警告タグは Reset = TRUE または ErrorAck =TRUE 以降の警告を示します。Warning は保持されます。 周期的警告(たとえば、プロセス値警告)が、警告の原因が取り除かれるまで、表示されます。 警告は、その原因がなくなると、すぐに自動的に削除されます。 周期的でない警告(たとえば、変曲点が見つからないなど)はそのままで、エラーと同様に削除されます。
SavePosition	REAL	0.0	代替出力値 ErrorBehaviour = TRUE の場合、アクチュエータはエラーが発生した場合にプラントにとって安全な位置に移動します。 代替出力値に達すると同時に、PID_3Step は ActivateRecoverMode に従った動作モードに切り替わります。
CurrentSetpoint	REAL	0.0	現在アクティブなセットポイント。 この値は同調開始時に固定されます。
CancelTuningLevel	REAL	10.0	同調中のセットポイントの許容変動。 同調は以下の状態になるまでキャンセルされません。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel
Progress	REAL	0.0	同調の進捗状況(パーセンテージ(0.0～100.0)で表示)
Config.InputPerOn	BOOL	TRUE	InputPerOn = TRUE の場合、Input_PER パラメータが使用されます。 InputPerOn = FALSE の場合、Input パラメータが使用されます。
Config.OutputPerOn	BOOL	FALSE	OutputPerOn = TRUE の場合、Output_PER パラメータが使用されます。 OutputPerOn = FALSE の場合、Ouput_UP および Output_DN パラメータが使用されます。
Config.InvertControl	BOOL	FALSE	反転制御ロジック InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。
Config.FeedbackOn	BOOL	FALSE	FeedbackOn = FALSE の場合、位置フィードバックがシミュレートされます。 位置フィードバックは通常、FeedbackOn = TRUE の場合に有効になります。
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn は、FeedbackOn = TRUE のときのみ有効です。 FeedbackPerOn = TRUE の場合、アナログ入力が位置フィードバック(Feedback_PER パラメータ)として使用されます。 FeedbackPerOn = FALSE の場合、Feedback パラメータが位置フィードバックとして使用されます。

Config.ActuatorEndStopOn	BOOL	FALSE	ActuatorEndStopOn = TRUE の場合、位置フィードバック Actuator_L および Actuator_H が考慮されます。
Config.InputUpperLimit	REAL	120.0	プロセス値の上限値 この制限値が守られていることを確認するために、Input と Input_PER をモニタします。 I/O 入力で、プロセス値が標準範囲よりも最大 18%大きくなる場合があります(範囲オーバー)。「プロセス値の上限値」オーバーのエラーは通知されません。 断線および短絡のみが認識され、PID_3Step が設定されたエラーに対する応答に従って応答します。 InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	プロセス値の下限値 InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	プロセス値の警告上限値 プロセス値制限範囲外の InputUpperWarning を設定すると、設定されたプロセス値の上限絶対値が警告上限値として使用されます。 プロセス値制限範囲内の InputUpperWarning を設定すると、この値が警告上限値として使用されます。 InputUpperWarning > InputLowerWarning InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	プロセス値の警告下限値 プロセス値制限範囲外の InputLowerWarning を設定すると、設定されたプロセス値の下限絶対値が警告下限値として使用されます。 プロセス値制限範囲内の InputLowerWarning を設定すると、この値が警告下限値として使用されます。 InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	出力値の上限値 詳細は、OutputLowerLimit を参照してください。
Config.OutputLowerLimit	REAL	0.0	出力値の下限値 OutputPerOn = TRUE または FeedbackOn = TRUE の場合、-100% ~ +100%の値の範囲(0 を含めて)が有効です。-100%では Output = -27648、+100%では Output = 27648 です。 OutputPerOn = FALSE の場合、0% ~ 100%の値の範囲が有効です。 バルブは 0%で完全に閉じ、100%で完全に開きます。
Config.SetpointUpperLimit	REAL	+3.402822e+38	セットポイントの上限値 プロセス値制限範囲外の SetpointUpperLimit を設定すると、設定されたプロセス値の上限絶対値があらかじめセットポイントの上限値として割り当てられます。

			プロセス値制限範囲内の SetpointUpperLimit を設定すると、この値がセットポイントの上限値として使用されます。
Config.SetpointLowerLimit	REAL	-3.402822e+38	セットポイントの下限値 プロセス値制限範囲外の SetpointLowerLimit を設定すると、設定されたプロセス値の下限絶対値があらかじめセットポイントの下限値として割り当てられます。 プロセス値制限範囲内の SetpointLowerLimit を設定すると、この値がセットポイントの下限値として使用されます。
Config.MinimumOnTime	REAL	0.0	最小 ON 時間 サーボドライブをスイッチオンにしていなければならない最小時間(秒単位)。
Config.MinimumOffTime	REAL	0.0	最小 OFF 時間 サーボドライブをスイッチオフにしていなければならない最小時間(秒単位)。
Config.VirtualActuatorLimit	REAL	150.0	以下のすべての条件が満たされている場合、アクチュエータは VirtualActuatorLimit × Retain.TransitTime/100 の最大時間の間、1 つの方向に移動し、警告 2000h が出力されます。 • Config.OutputPerOn = FALSE • Config.ActuatorEndStopOn = FALSE • Config.FeedbackOn = FALSE Config.OutputPerOn = FALSE および Config.ActuatorEndStopOn = TRUE、または Config.FeedbackOn = TRUE の場合、警告 2000h が出力されます。Config.OutputPerOn = TRUE の場合、VirtualActuatorLimit は考慮されません。
Config.InputScaling.UpperPointIn	REAL	27648.0	Input_PER (High)のスケーリング Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.LowerPointIn	REAL	0.0	Input_PER (Low)のスケーリング Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.UpperPointOut	REAL	100.0	スケーリングされた High プロセス値 Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.LowerPointOut	REAL	0.0	スケーリングされた Low プロセス値 Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	Feedback_PER (High)のスケーリング

			Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.LowerPointIn	REAL	0.0	Feedback_PER (Low)のスケーリング Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.UpperPointOut	REAL	100.0	エンドストップ上限 Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.LowerPointOut	REAL	0.0	エンドストップ下限 Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
GetTransitTime.InvertDirection	BOOL	FALSE	InvertDirection = FALSE の場合、バルブ移行時間を求めるために、バルブが完全に開き、閉じた後、再度開きます。 InvertDirection = TRUE の場合、バルブが完全に閉じ、開いた後、再度閉じます。
GetTransitTime.SelectFeedback	BOOL	FALSE	SelectFeedback = TRUE の場合、Feedback_PER となるか、または Feedback が移行時間の測定で考慮されます。 SelectFeedback = FALSE の場合、Actuator_H および Actuator_L が移行時間の測定で考慮されます。
GetTransitTime.State	INT	0	移行時間測定の現在のフェーズ • State = 0: 無効 • State = 1: バルブを完全に開きます • State = 2: バルブを完全に閉じます • State = 3: バルブをターゲット位置に移動します (NewOutput) • State = 4: 移行時間の測定が正常に終了しました • State = 5: 移行時間の測定がキャンセルされました
GetTransitTime.NewOutput	REAL	0.0	位置フィードバック付きの移行時間測定のターゲット位置 ターゲット位置は、「エンドストップ上限」と「エンドストップ下限」の間であることが必要です。 NewOutput と ScaledFeedback の差は、許容制御範囲の少なくとも 50%であることが必要です。
CycleTime.StartEstimation	BOOL	TRUE	StartEstimation = TRUE の場合、PID_3Step サンプリング時間の測定が開始されます。CycleTime.StartEstimation = FALSE の場合はすぐに測定が完了します。

CycleTime.EnEstimation	BOOL	TRUE	EnEstimation = TRUE の場合、PID_3Step サンプリング時間が計算されます。 CycleTime.EnEstimation = FALSE の場合、PID_3Step サンプリング時間は計算されず、CycleTime.Value の設定を手動で修正する必要があります。
CycleTime.EnMonitoring	BOOL	TRUE	EnMonitoring = TRUE の場合、PID_3Step サンプリング時間がモニタされます。 サンプリング時間以内に PID_3Step を実行できない場合は、エラー 0800h が出力され、動作モードが変更されます。 ActivateRecoverMode および ErrorBehaviour が、切り替え先動作モードを決定します。 EnMonitoring = FALSE の場合、PID_3Step サンプリング時間はモニタされず、エラー 0800h は出力されず、動作モードは変更されません。
CycleTime.Value	REAL	0.1	PID_3Step サンプリング時間(秒単位) CycleTime.Value が自動的に測定されます。通常、呼び出し OB のサイクルタイムと同じ値です。
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Retain.CtrlParams.SetByUser の保存された値。 LoadBackUp = TRUE で、CtrlParamsBackUp 構造体から値を再読み込みできます。
CtrlParamsBackUp.Gain	REAL	1.0	比例ゲインの保存エリア
CtrlParamsBackUp.Ti	REAL	20.0	積分動作時間(秒単位)の保存エリア
CtrlParamsBackUp.Td	REAL	0.0	微分動作時間(秒単位)の保存エリア
CtrlParamsBackUp.TdFiltRatio	REAL	0.2	微分の遅延係数の保存エリア
CtrlParamsBackUp.PWeighting	REAL	1.0	比例動作重みの保存エリア
CtrlParamsBackUp.DWeighting	REAL	1.0	微分動作重みの保存エリア
CtrlParamsBackUp.Cycle	REAL	1.0	PID アルゴリズムのサンプリング時間(秒単位)の保存エリア
CtrlParamsBackUp.InputDeadBand	REAL	0.0	制御偏差のデッドバンド幅の保存エリア
PIDSelfTune.SUT.CalculateParams	BOOL	FALSE	制御システムのプロパティは同調中に保存されます。CalculateParams = TRUE の場合、PID パラメータがこれらのプロパティに基づいて再計算されます。TuneRule で設定された方法を使用して、PID パラメータが計算されます。計算の後に、CalculateParams が FALSE に設定されます。
PIDSelfTune.SUT.TuneRule	INT	1	プレチューニング時のパラメータの計算に使用される方法: • SUT.TuneRule = 0: PID 高速 I • SUT.TuneRule = 1: PID 低速 I • SUT.TuneRule = 2: Chien、Hrones、および Reswick PID • SUT.TuneRule = 3: Chien、Hrones、Reswick PI • SUT.TuneRule = 4: PID 高速 II

			• SUT.TuneRule = 5: PID 低速 II
PIDSelfTune.SUT.State	INT	0	SUT.State タグは、プレチューニングの現在のフェーズを示します。 • State = 0 プレチューニングの初期化 • State = 50: 位置フィードバックなしでの開始位置の決定 • State = 100: 標準偏差の計算 • State = 200 変曲点の決定 • State = 300: 立ち上がり時間の決定 • State = 9900 プレチューニングが成功 • State = 1 プレチューニングが失敗
PIDSelfTune.TIR.RunIn	BOOL	FALSE	RunIn タグで、微調整をプレチューニングなしで実行することも指定できます。 • RunIn = FALSE 無効なモードまたは手動モードで微調整が開始されると、プレチューニングが開始されます。 自動モードで微調整が開始されると、システムは既存の PID パラメータを使用してセットポイントを制御します。 この場合のみ、微調整が開始します。 プレチューニングが行えない場合、PID_3Step は同調が開始されたモードに切り替わります。 • RunIn = TRUE プレチューニングはスキップされます。 PID_3Step は、最小または最大出力値でセットポイントに到達することを試みます。 この結果、オーバーシュートが増大する可能性があります。 この場合のみ、微調整が開始します。 微調整後、RunIn が FALSE に設定されます。
PIDSelfTune.TIR.CalculateParams	BOOL	FALSE	制御システムのプロパティは同調中に保存されます。CalculateParams = TRUE の場合､PID パラメータがこれらのプロパティに基づいて再計算されます。TuneRule で設定された方法を使用して、PID パラメータが計算されます。計算の後に、CalculateParams が FALSE に設定されます。
PIDSelfTune.TIR.TuneRule	INT	0	微調整時のパラメータの計算に使用される方法: • TIR.TuneRule = 0: PID (自動) • TIR.TuneRule = 1: PID (高速) • TIR.TuneRule = 2: PID (低速) • TIR.TuneRule = 3: Ziegler-Nichols PID • TIR.TuneRule = 4: Ziegler-Nichols PI • TIR.TuneRule = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	TIR.State タグは、微調整の現在のフェーズを示します。 • State = -100 微調整が実行可能ではありません。 プレチューニングが最初に実行されます。

			• State = 0: 微調整の初期化 • State = 200: 標準偏差の計算 • State = 300 は、最大または最小出力値でセットポイントに達することを試みます。 • State = 400: 既存の PID パラメータによるセットポイントへの到達の試み (プレチューニングが正常に行われた場合) • State = 500: 振動の決定とパラメータの計算 • State = 9900: 微調整が成功 • State = 1: 微調整が成功
Retain.TransitTime	REAL	30.0	モータ移行時間(秒単位) 作動ドライブがバルブを閉状態から開状態に遷移させるために必要な時間(秒単位)。 TransitTime は保持されます。
Retain.CtrlParams.SetByUser	BOOL	FALSE	SetByUser = FALSE の場合、PID パラメータが自動的に決定され、PID_3Step は出力値でのデッドバンドありで動作します。 デッドバンド幅は、同調時に出力値の標準偏差に基づいて計算され、Retain.CtrlParams.OutputDeadBand に保存されます。 SetByUser = TRUE の場合、PID パラメータが手動で入力され、PID_3 Step は出力値でのデッドバンドなしで動作します。 Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser は保持されます。
Retain.CtrlParams.Gain	REAL	1.0	有効な比例ゲイン 制御ロジックを反転するには、Config.InvertControl タグを使用します。 Gain の負の値も制御ロジックを反転させます。 制御ロジックの設定には、InvertControl だけを使用することをお勧めします。 制御ロジックは、InvertControl = TRUE および Gain < 0.0 の場合にも反転します。 Gain は保持されます。
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: 有効な積分動作時間(秒単位) • Ti = 0.0: 積分動作が無効です Ti は保持されます。
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: 有効な微分動作時間(秒単位) • Td = 0.0: 微分動作が無効です Td は保持されます。
Retain.CtrlParams.TdFiltRatio	REAL	0.2	動作中の微分の遅延係数 微分の遅延係数によって、微分動作の影響を遅らせます。 微分の遅延 = 微分動作時間 × 微分の遅延係数 • 0.0: 微分動作は 1 サイクルに限って有効であり、ほとんど影響はありません。 • 0.5: この値は、主要な時定数が 1 つの制御システムで実際に有効であることが実証されています。

			• > 1.0: この係数が大きいほど、微分動作の影響が遅くなります。 TdFiltRatio は保持されます。
Retain.CtrlParams.PWeighting	REAL	1.0	有効な比例動作重み 比例動作はセットポイントの変更とともに弱くなることがあります。 0.0 ~ 1.0 の値を適用できます。 • 1.0: セットポイントの変更に対する比例動作が完全に有効です • 0.0: セットポイントの変更に対する比例動作が有効でありません プロセス値が変更になった場合は、比例動作は常に完全に有効です。 PWeighting は保持されます。
Retain.CtrlParams.DWeighting	REAL	1.0	有効な微分動作重み 微分動作はセットポイントの変更とともに弱くなることがあります。 0.0 ~ 1.0 の値を適用できます。 • 1.0: セットポイントの変更時に微分動作が完全に有効です • 0.0: セットポイントの変更時に微分動作が有効でありません プロセス値が変更になった場合は、微分動作は常に完全に有効です。 DWeighting は保持されます。
Retain.CtrlParams.Cycle	REAL	1.0	有効な PID アルゴリズムのサンプリング時間(秒単位)。呼び出し OB のサイクルタイムの整数倍に丸められます。 Cycle は保持されます。
Retain.CtrlParams.InputDeadBand	REAL	0.0	制御偏差のデッドバンド幅 InputDeadBand は保持されます。

注記
PID コントローラの誤動作を防止するには、「無効」モードでこの表に記載されているタグを変更します。

パラメータ State と Mode V2

パラメータの相互関係

State パラメータは、PID コントローラの現在の動作モードを示します。 State パラメータを変更することはできません。

ModeActivate の立ち上がりエッジで、PID_3Step が ModeIN-OUT パラメータに保存された動作モードに切り替わります。

CPU がオンになるか、Stop から RUN モードに切り替わると、PID_3Step が Mode パラメータに保存された動作モードで開始されます。PID_3Step を「無効」モードのままにするには、RunModeByStartup = FALSE を設定してください。

値の意味

State	動作モードの説明
0	無効 コントローラはオフになり、バルブ位置の変更をもう行いません。
1	プレチューニング プレチューニングでは、出力値のパルスに対するプロセス応答が決定され、変曲点が検索されます。 PID パラメータは、最大立ち上がり速度と制御システムのデッドタイムから計算されます。 プレチューニングと微調整を実行すると最良の PID パラメータを取得できます。 プレチューニングの必要条件 • モータ移行時間が設定または測定されていること。 • 無効(State = 0)、手動モード(State = 4)、または自動モード(State = 3)であること。 • ManualEnable = FALSE • Reset = FALSE • セットポイントとプロセス値が、設定済みの制限範囲内であること。 プロセス値が安定しているほど、PID パラメータの計算が簡単になり、結果が正確になります。 プロセス値の上昇率がノイズと比べて大幅に高ければ、プロセス値のノイズは許容できます。 これは、動作モード「無効」および「手動モード」で最も起こりやすくなります。 セットポイントが、CurrentSetpoint タグで固定されます。 同調は以下の場合にキャンセルされます。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel PID パラメータは再計算される前にバックアップされ、LoadBackUp で再度有効になります。 プレチューニングが正常に終了すると、コントローラが自動モードに切り替わります。プレチューニングが失敗した場合、動作モードの切り替えは ActivateRecoverMode および ErrorBehaviour に依存します。

	プレチューニングフェーズは、SUT.State タグで示されます。
2	微調整 微調整は、プロセス値の一定の制限された振動を生成します。 PID パラメータは、この振動の振幅と周波数に基づいて再計算されます。 通常は微調整に基づく PID パラメータの方が、プレチューニングに基づく PID パラメータよりも優れたマスタコントロールと外乱特性を実現します。 プレチューニングと微調整を実行すると最良の PID パラメータを取得できます。 PID_3Step は、プロセス値のノイズよりも大きな振動の生成を自動的に試行します。 プロセス値の安定性によって微調整が受ける影響は最小限です。 セットポイントが、CurrentSetpoint タグで固定されます。 同調は以下の場合にキャンセルされます。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel PID パラメータは、微調整の前に、バックアップされます。 PID パラメータは、Load-BackUp で再度有効にすることができます。 微調整の必要条件 • モータ移行時間が設定または測定されていること。 • セットポイントとプロセス値が、設定済みの制限範囲内であること。 • ManualEnable = FALSE • Reset = FALSE • 自動(State = 3)、無効(State = 0)、または手動(State = 4)モードであること。 微調整は、開始されたときのモードに従って以下のように実行されます。 • 自動モード(State = 3) 同調を使用して既存の PID パラメータを改善する場合は、自動モードで微調整を開始します。 PID_3Step は、制御ループが安定し、微調整の必要条件が満たされるまでは、既存の PID パラメータを使用してシステムを制御します。 制御ループが安定して微調整の必要条件が満たされた場合のみ、微調整が開始します。 • 無効モード(State = 0)または手動モード(State = 4) プレチューニングの必要条件が満たされると、プレチューニングが開始されます。 制御ループが安定し、微調整の必要条件が満たされるまでは、決定された PID パラメータが制御に使用されます。 PIDSelfTune.TIR.RunIn = TRUE の場合、プレチューニングがスキップされ、最小または最大出力値によるセットポイントへの到達が試みられます。 この結果、オーバーシュートが増大する可能性があります。 微調整が自動的に開始します。 微調整が正常に終了すると、コントローラが自動モードに切り替わります。 微調整が失敗した場合、動作モードの切り替えは ActivateRecoverMode および ErrorBehaviour に依存します。 微調整フェーズは、TIR.State タグで示されます。
3	自動モード 自動モードでは、PID_3Step は、指定されたパラメータに従って、コントロールされるシステムを制御します。 以下の必要条件の 1 つが満たされた場合、コントローラは自動モードに切り替わります。

	• プレチューニングの正常終了 • 微調整の正常終了 • ModeIN-OUT パラメータの値 3 への変更と ModeActivate の立ち上がりエッジ 手動モードから自動モードへの切り替えは、コミッショニングエディタで実行した場合にのみバンプレスです。 ActivateRecoverMode タグは、自動モードで考慮されます。
4	手動モード 手動モードでは、Manual_UP および Manual_DN パラメータまたは ManualValue パラメータで、手動出力値を指定します。 エラー発生時にアクチュエータを出力値まで移動できるかどうかは、ErrorBits パラメータに記載されています。 この動作モードは、ManualEnable = TRUE を使用して有効にすることもできます。 Mode および ModeActivate のみを使用して動作モードを変更することをお勧めします。 手動モードから自動モードへの切り替えはバンプレスです。 手動モードは、エラーが保留中でも使用可能です。
5	代替出力値アプローチ この動作モードは、Errorbehaviour = TRUE および ActivateRecoverMode = FALSE.の場合、エラーが発生すると有効になります。 PID_3Step は、アクチュエータを代替出力値まで移動した後、「無効」モードに切り替わります。
6	移行時間の測定 バルブを閉じた状態から完全に開くためにモータが必要とする時間が特定されます。 この動作モードは、Mode = 6 および ModeActivate = TRUE が設定されたときに有効になります。 移行時間の測定にエンドストップ信号を使用すると、バルブは、現在の位置から完全に開き、完全に閉じ、再度完全に開きます。 GetTransitTime.InvertDirection = TRUE の場合、この動作は逆転します。 移行時間の測定に位置フィードバックを使用すると、アクチュエータが、現在の位置からターゲット位置へ移動されます。 出力制限値は、移行時間の測定中は考慮されません。 アクチュエータはエンドストップ上限または下限に移動できます。
7	エラーモニタリング コントロールアルゴリズムはオフになり、バルブ位置の移動をもう行いません。 この動作モードは、エラー発生時、「無効」モードの代わりに有効になります。 以下のすべての条件を満たす必要があります。 • 自動モード(Mode = 3) • Errorbehaviour = FALSE • ActivateRecoverMode = TRUE • ActivateRecoverMode が有効になる 1 つまたは複数のエラーが発生しました。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。
8	エラーモニタリング付きの代替出力値アプローチ この動作モードは、エラーが発生した時に「代替出力値アプローチ」モードの代わりに有効になります。PID_3Step は、アクチュエータを代替出力値まで移動し、「エラーのモニタ」モードに切り替わります。

	以下のすべての条件を満たす必要があります。 • 自動モード(Mode = 3) • Errorbehaviour = TRUE • ActivateRecoverMode = TRUE • ActivateRecoverMode が有効になる 1 つまたは複数のエラーが発生しました。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。
10	エンドストップ信号なしの手動モード Config.ActuatorEndStopOn = TRUE の場合でも、エンドストップ信号は考慮**されません**。 出力制限値は、移行時間の測定中は考慮されません。 それ以外は、PID_3Step は手動モードの場合と同様に動作します。

ENO 特性

State = 0 の場合、ENO = FALSE です。

State ≠ 0 の場合、ENO = TRUE です。

コミッショニング時の動作モードの自動切り替え

プレチューニングまたは微調整が正常に行われると、自動モードが有効になります。 下の表に、プレチューニングが正常に行われた場合に Mode および State がどのように切り替わるかを示します。

サイクル番号	Mode	State	操作
0	4	4	セット Mode = 1
1	1	4	セット ModeActivate = TRUE
1	4	1	State の値が Mode パラメータに保存されます。 プレチューニングが開始されます。
n	4	1	プレチューニングの正常終了
n	3	3	自動モードが開始されます。

PID_3Step は、エラー発生時に自動的に動作モードを切り替えます。 下の表に、プレチューニングでエラーが発生した場合 Mode および State がにどのように切り替わるかを示します。

サイクル番号	Mode	State	操作
0	4	4	セット Mode = 1
1	1	4	セット ModeActivate = TRUE
1	4	1	State の値が Mode パラメータに保存されます。 プレチューニングが開始されます。
n	4	1	プレチューニングがキャンセルされました
n	4	4	手動モードが開始されます。

ActivateRecoverMode = TRUE の場合、Mode パラメータに保存された動作モードが有効になります。 移行時間の測定、プレチューニング、または微調整の開始時に、PID_3Step が State の値を

ModeIN-OUT パラメータに保存しました。そのため、PID_3Step は移行時間の測定または同調が開始された動作モードに切り替わります。

ActivateRecoverMode = FALSE の場合、「無効」モードまたは「代替出力値アプローチ」モードが有効になります。

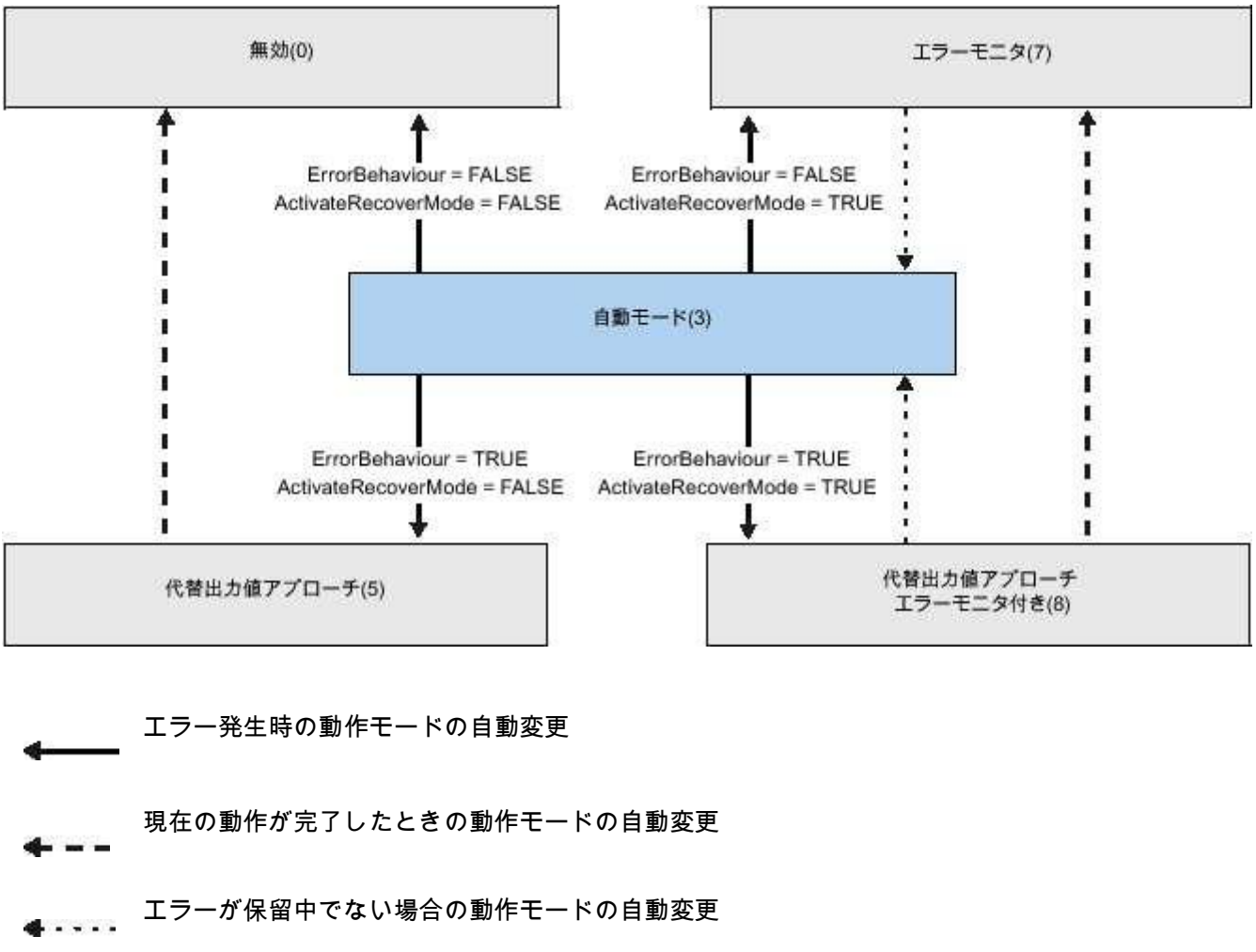
移行時間の測定後の動作モードの自動切り替え

ActivateRecoverMode = TRUE の場合、移行時間の測定が正常に行われた後で、Mode パラメータに保存された動作モードが有効になります。

ActivateRecoverMode = FALSE の場合、移行時間の測定が正常の行われた後で、システムは「無効」動作モードに切り替わります。

自動モードでの動作モードの自動切り替え

PID_3Step は、エラー発生時に自動的に動作モードを切り替えます。 次の図は、動作モードの変更に対する ErrorBehaviour および ActivateRecoverMode の影響を示します。



エラー発生時の動作モードの自動変更

現在の動作が完了したときの動作モードの自動変更

エラーが保留中でない場合の動作モードの自動変更

パラメータ ErrorBits V2

複数のエラーが同時に保留中の場合、ErrorBits の値がバイナリ加算で表示されます。 たとえば、ErrorBits = 0003h の表示はエラー 0001h および 0002h が同時に保留中であることを示します。

位置フィードバックがある場合、PID_3Step は手動モードで ManualValue を出力値として使用します。 例外は Errorbits = 10000h です。

ErrorBits (DW#16#...)	説明
0000	エラーはありません。
0001	「Input」パラメータがプロセス値制限範囲外です。 • Input > Config.InputUpperLimit または • Input < Config.InputLowerLimit エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_3Step は自動モードのままです。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
0002	「Input_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。 エラーが発生する前に自動モードが有効で ActivateRecoverMode = TRUE の場合、PID_3Step は「エラーモニタリング付きの代替出力値アプローチ」または「エラーモニタリング」モードに切り替わります。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
0004	微調整中のエラー プロセス値の振幅を維持できませんでした。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_3Step は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0010	セットポイントが、同調時に変更されました。 CancelTuningLevel タグで、セットポイントの許容変動を設定できます。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_3Step は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0020	プレチューニングを微調整中に行うことはできません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_3Step は微調整モードのままです。
0080	プレチューニング中のエラー 出力制限値の設定が不正です。 出力値の制限値が正しく設定されていて、制御ロジックと一致しているかどうかをチェックします。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_3Step は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。

0100	微調整時のエラーによって、パラメータが無効になりました。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_3Step は同調をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0200	「Input」パラメータの値が無効です。 値が無効な番号書式です。 エラーが発生する前に自動モードが有効で ActivateRecoverMode = TRUE の場合、PID_3Step は「エラーモニタリング付きの代替出力値アプローチ」または「エラーモニタリング」モードに切り替わります。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
0400	出力値の計算が失敗しました。 PID パラメータをチェックしてください。 エラーが発生する前に自動モードが有効で ActivateRecoverMode = TRUE の場合、PID_3Step は「エラーモニタリング付きの代替出力値アプローチ」または「エラーモニタリング」モードに切り替わります。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
0800	サンプリング時間エラー: PID_3Step が、サイクル割り込み OB のサンプリング時間内に呼び出されていません。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_3Step は自動モードのままです。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
1000	「Setpoint」パラメータの値が無効です。 値が無効な番号書式です。 エラーが発生する前に自動モードが有効で ActivateRecoverMode = TRUE の場合、PID_3Step は「エラーモニタリング付きの代替出力値アプローチ」または「エラーモニタリング」モードに切り替わります。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
2000	「Feedback_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。 アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。 手動モードでは、アクチュエータの位置は、Manual_UP および Manual_DN でのみ変更できます。ManualValue では変更できません。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
4000	「Feedback」パラメータの値が無効です。 値が無効な番号書式です。

	アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。 手動モードでは、アクチュエータの位置は、Manual_UP および Manual_DN でのみ変更できます。ManualValue では変更できません。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
8000	デジタル位置フィードバック中のエラーです。 Actuator_H = TRUE および Actuator_L = TRUE です。 アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。 この状態では、手動モードは可能ではありません。 アクチュエータをこの状態から移行するには、[アクチュエータのエンドストップ]を無効にするか(Config.ActuatorEndStopOn = FALSE)、エンドストップ信号なしの手動モードに切り替える必要があります(Mode = 10)。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。 プレチューニングモード、微調整モード、または移行時間の測定モード、および ActivateRecoverMode = TRUE がエラー発生前に有効になっていた場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。
10000	「ManualValue」パラメータの値が無効です。 値が無効な番号書式です。 アクチュエータを手動値まで移動できず、アクチュエータは現在位置のままです。 ManualValue で有効な値を指定するか、Manual_UP および Manual_DN でアクチュエータを手動モードで移動します。
20000	SavePosition タグの値が無効です。 値が無効な番号書式です。 アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。
40000	「Disturbance」パラメータの値が無効です。値が無効な番号書式です。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、Disturbance は 0 に設定されます。PID_3Step は自動モードのままです。 プレチューニングまたは微調整がエラー発生前に有効になっていて、 ActivateRecoverMode = TRUE の場合、PID_3Step は Mode パラメータに保存されている動作モードに切り替わります。 現在のフェーズの Disturbance が出力値に影響しない場合、同調はキャンセルされません。 エラーは、移行時間の測定中は影響しません。

タグ ActivateRecoverMode V2

ActivateRecoverMode タグは、エラーに対する応答を決定します。 Error パラメータは、エラーが保留中かどうかを示します。 エラーが保留中でない場合、Error = FALSE となります。 ErrorBits パラメータは、どのエラーが発生したかを示します。

通知
システムが損傷することがあります。 ActivateRecoverMode = TRUE の場合、PID_3Step はプロセス制限値を越えていても自動モードのままです。 そのため、システムが損傷することがあります。 システムを損傷から守るために、エラーの場合に制御システムがどのように応答するかを設定する必要があります。

自動モード

ActivateRecoverMode	説明
FALSE	エラー発生時には、PID_3Step は、「無効」または「代替出力値アプローチ」モードに切り替わります。 コントローラは、Reset の立ち下りエッジまたは ModeActivate の立ち上がりエッジでのみ有効になります。
TRUE	自動モードでエラーが頻繁に発生する場合、この設定は制御応答にマイナスの影響を及ぼします。エラーが発生するたびに、PID_3Step が計算された出力値と代替出力値の間で切り替わるからです。 この場合は、ErrorBits パラメータをチェックして、エラーの原因を取り除いてください。 以下のエラーの 1 つまたは複数が発生しても、PID_3Step は自動モードのままです。 • 0001h:「Input」パラメータがプロセス値制限範囲外です。 • 0800h: サンプリング時間のエラー • 40000h: Disturbance パラメータの値が無効です。 1 つまたは複数の以下のエラーが発生する場合、PID_3Step は、「エラーモニタ付き代替出力値アプローチ」または「エラーモニタ」モードに切り替わります。 • 0002h: 「Input_PER」パラメータの値が無効です。 • 0200h: 「Input」パラメータの値が無効です。 • 0400h: 出力値の計算が失敗しました。 • 1000h: 「Setpoint」パラメータの値が無効です。 以下の 1 つまたは複数のエラーが発生した場合、PID_3Step はアクチュエータを移動できなくなります。 • 2000h: 「Feedback_PER」パラメータの値が無効です。 • 4000h: 「Feedback」パラメータの値が無効です。 • 8000h: デジタル位置フィードバック中のエラーです。 • 20000h: SavePosition タグの値が無効です。 値が無効な番号フォーマットです。 この特性は ErrorBehaviour とは無関係です。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。

プリチューニング、微調整、および移行時間の測定

ActivateRecoverMode	説明
FALSE	エラー発生時には、PID_3Step は、「無効」または「代替出力値アプローチ」モードに切り替わります。 コントローラは、Reset の立ち下りエッジまたは ModeActivate の立ち上がりエッジでのみ有効になります。 コントローラは、移行時間の測定が正常に行われた後で「無効」モードに切り替わります。
TRUE	以下のエラーが発生した場合、PID_3Step はアクティブモードのままです。 • 0020h: プリチューニングを微調整中に行うことはできません。 以下のエラーは無視されます。 • 10000h: 「ManualValue」パラメータの値が無効です。 • 20000h: SavePosition タグの値が無効です。 これ以外のエラーが発生した場合、PID_3Step は同調をキャンセルして同調が開始されたモードに切り替わります。

手動モード

ActivateRecoverMode は手動モードでは無効です。

タグ Warning V2

複数の警告が同時に保留中の場合、その値がバイナリ加算で表示されます。たとえば、警告 0005h の表示は警告 0001h および 0004h が同時に保留中であることを示します。

Warning (DW#16#...)	説明
0000	保留中の警告がありません。
0001	プレチューニング時に変曲点が見つかりませんでした。
0004	セットポイントが、設定済みの制限値に制限されました。
0008	選択された計算方法で、コントロールされるシステムの必要なプロパティの中に定義されなかったものがあります。 代わりに、PID パラメータが TIR.TuneRule = 3 メソッドによって計算されました。
0010	Reset = TRUE または ManualEnable = TRUE.のために、動作モードを変更できませんでした。
0020	呼び出し OB のサイクルタイムが、PID アルゴリズムのサンプリング時間を制限します。 さらに短い OB サイクルタイムを使用して、結果を改善してください。
0040	プロセス値が、その警告制限値の 1 つを超えました。
0080	Mode での値が不正です。 動作モードは変更されません。
0100	手動値が、コントローラ出力の制限値に制限されました。
0200	調整の指定されたルールはサポートされていません。 PID パラメータは計算されません。
0400	アクチュエータの設定が選択された測定方法と適合しないため、移行時間を測定できません。
0800	現在の位置と新しい出力値の間の相違が、移行時間測定には小さすぎます。 これにより、不正な結果を生じる場合があります。 現在の出力値と新しい出力値の相違は、制御範囲全体の少なくとも 50%であることが必要です。
1000	代替出力値が出力制限値を超えているため、代替出力値に達することができません。
2000	アクチュエータが Config.VirtualActuatorLimit × Retain.TransitTime よりも長く一方向に移動しました。 アクチュエータがエンドストップ信号に達したかどうかをチェックします。

以下の警告は、原因が取り除かれるとすぐに削除されます。

- 0001h
- 0004h
- 0008h
- 0040h
- 0100h
- 2000h

他のすべての警告は、Reset または ErrorAck の立ち上がりエッジで解除されます。

PID_3Step V1

この章には下記に関する情報が記載されています：

PID_3Step V1 の説明

説明

PID_3Step 命令を使用して、積分動作を行うバルブまたはアクチュエータ用の自己同調機能を使用して PID コントローラを設定します。

次の動作モードが使用可能です。

- 無効
- プリチューニング
- 微調整
- 自動モード
- 手動モード
- 代替出力値アプローチ
- 移行時間の測定
- エラーモニタリング付きの代替出力値アプローチ
- エラーモニタリング

動作モードの詳細については、State パラメータを参照してください。

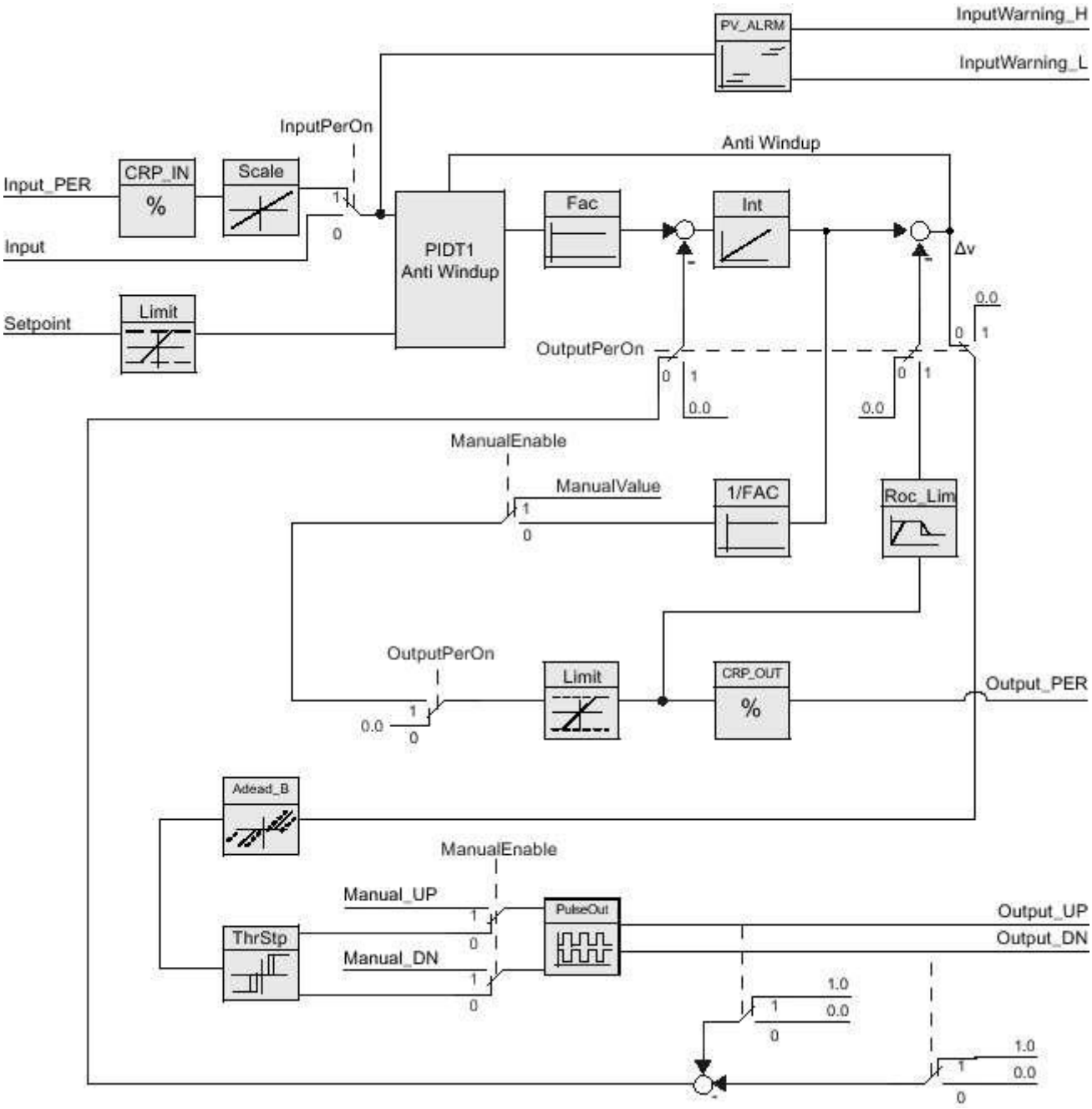
PID アルゴリズム

PID_3Step は、比例および微分動作のワインドアップ防止および重み付け付きの PIDT1 コントローラです。 出力値の計算には、次の数式が使用されます。

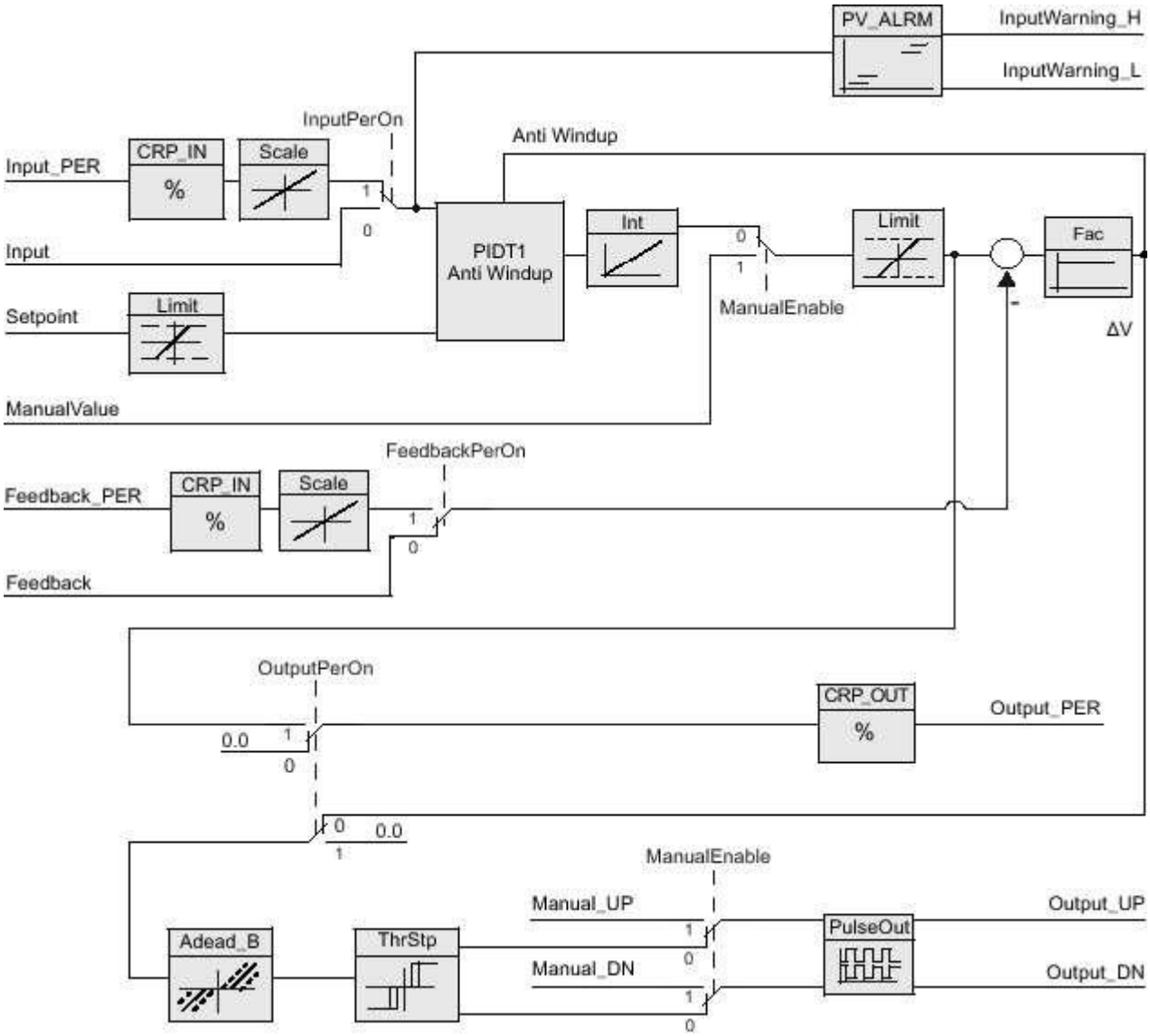
$$\Delta y = K_p \cdot s \cdot \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

シンボル	説明
y	出力値
K_p	比例ゲイン
s	ラプラス演算子
B	比例動作の重み付け
w	セットポイント
x	プロセス値
T_I	積分動作時間
A	微分の遅延係数($T1 = a \times T_D$)
T_D	微分動作時間
C	微分動作の重み付け

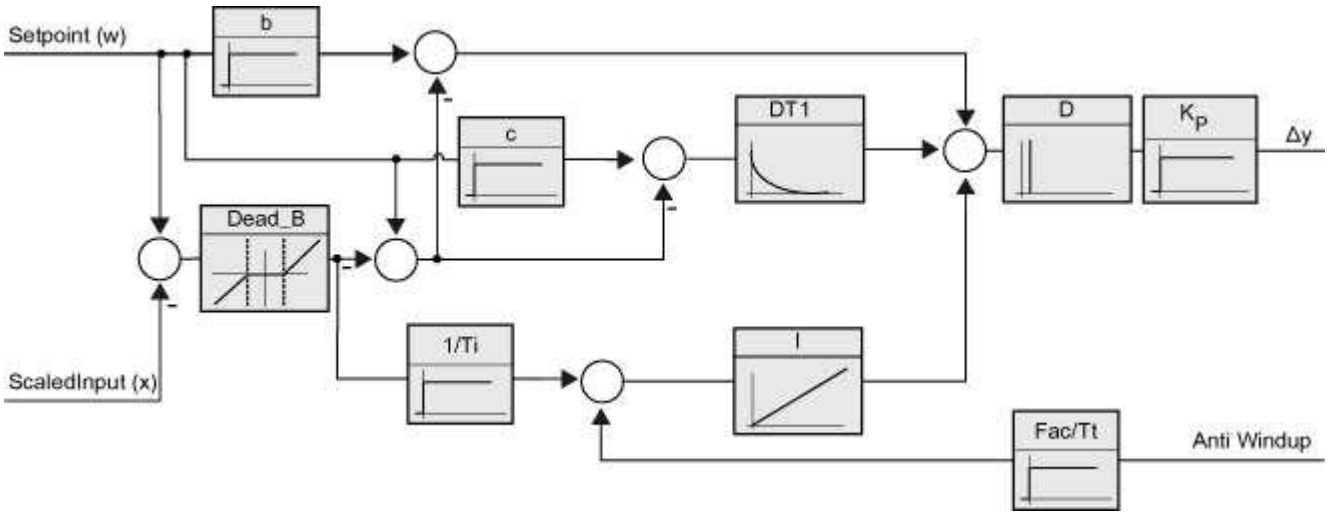
位置フィードバックなしのブロックダイアグラム



位置フィードバック付きのブロックダイアグラム



ワインドアップ防止付きの PIDT1 のブロックダイアグラム



呼び出し

PID_3Step は、一定の時間間隔(呼び出し OB(可能な場合、周期割り込み OB)のサイクルタイム)で呼び出されます。

デバイスへのダウンロード

保持タグの現在値は、PID_3Step を完全にダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

スタートアップ

CPU のスタートアップ時には、PID_3Step は、最後に有効であった動作モードで開始されます。PID_3Step を「無効」モードのままにするには、RunModeByStartup = FALSE を設定してください。

エラーに対する応答

エラーが発生すると、エラーは Error パラメータに出力されます。 PID_3Step の応答を ErrorBehaviour および ActivateRecoverMode タグを使用して設定します。

ErrorBehaviour	ActivateRecoverMode	アクチュエータ設定 Output を以下の値に設定	応答
0	FALSE	現在の出力値	「無効」モード(Mode = 0)に切り替え
0	TRUE	エラー保留中の現在の出力値	「エラーモニタ」モード(Mode = 7)に切り替え
1	FALSE	代替出力値	「代替出力値アプローチ」モード(Mode = 5)に切り替え 「無効」モード(Mode = 0)に切り替え
1	TRUE	エラーが保留中の間、代替出力値	「エラーモニタ付きの代替出力値アプローチ」モード(Mode = 8)に切り替え 「エラーモニタ」モード(Mode = 7)に切り替え

ErrorBits パラメータは、どのエラーが発生したかを示します。

PID_3Step V1 の動作原理

プロセス値制限値のモニタリング

Config.InputUpperLimit および Config.InputLowerLimit タグで、プロセス値の上限値と下限値を指定します。 プロセス値がこの制限値を超えている場合、エラーが発生します(ErrorBits = 0001hex)。

Config.InputUpperWarning および Config.InputLowerWarning タグで、プロセス値の警告上限値と警告下限値を指定します。 プロセス値がこの警告制限値を超えている場合、警告が発生し(Warnings = 0040hex)、InputWarning_H または InputWarning_L 出力パラメータが TRUE に変わります。

セットポイントの制限

Config.SetpointUpperLimit および Config.SetpointLowerLimit タグで、セットポイントの上限値と下限値を指定します。PID_3Step は自動的にセットポイントをプロセス値の制限値に制限します。 セットポイントを小さい範囲に制限することができます。PID_3Step は、この範囲がプロセス制限値の範囲内であるかどうかをチェックします。 セットポイントがこの制限値を越えると、上限値または下限値がセットポイントとして使用され、出力パラメータ SetpointLimit_H または SetpointLimit_L が TRUE に設定されます。

セットポイントは、すべての動作モードで、制限されます。

出力値の制限

Config.OutputUpperLimit および Config.OutputLowerLimit タグで、出力値の上限値と下限値を指定します。 出力制限値は、「エンドストップ下限」および「エンドストップ上限」内でなければなりません。

- エンドストップ上限: Config.FeedbackScaling.UpperPointOut
- エンドストップ下限: Config.FeedbackScaling.LowerPointOut

ルール:

UpperPointOut ≥ OutputUpperLimit > OutputLowerLimit ≥ LowerPointOut

「エンドストップ上限」および「エンドストップ下限」に有効な値は、次の条件によって決まります。

- FeedbackOn
- FeedbackPerOn
- OutputPerOn

OutputPerOn	FeedbackOn	FeedbackPerOn	LowerPointOut	UpperPointOut
FALSE	FALSE	FALSE	設定不可(0.0%)	設定不可(100.0%)
FALSE	TRUE	FALSE	-100.0%または 0.0%	0.0%または+100.0%
FALSE	TRUE	TRUE	-100.0%または 0.0%	0.0%または+100.0%
TRUE	FALSE	FALSE	設定不可(100.0%)	設定不可(100.0%)
TRUE	TRUE	FALSE	-100.0%または 0.0%	0.0%または+100.0%
TRUE	TRUE	TRUE	-100.0%または 0.0%	0.0%または+100.0%

OutputPerOn = FALSE および FeedbackOn = FALSE の場合は、出力値を制限できません。 デジタル出力は、Actuator_H = TRUE または Actuator_L = TRUE でリセットされるか、あるいは移動時間がモータ移行時間の 110%に達するとリセットされます。

出力値は、100%で 27648 で、-100%で-27648 です。 PID_3Step は、バルブを完全に閉じることができなくてはいけません。 このため、ゼロを出力値制限範囲に含める必要があります。

代替出力値

エラーが発生した場合、PID_3Step は代替出力値を出力し、アクチュエータを SavePosition タグで指定された安全な位置に移動することができます。 代替出力値は、出力制限値内でなければなりません。

信号妥当性のモニタリング

以下のパラメータの値は、有効性をモニタされます。

- Setpoint
- Input
- Input_PER
- Feedback
- Feedback_PER
- Output

PID_3Step サンプリング時間のモニタ

理想的には、サンプリング時間は呼び出し OB のサイクルタイムと等値です。PID_3Step 命令は、2 つの呼び出し間の時間間隔を測定します。 これは、現在のサンプル時間です。 動作モードの変更ごと、また、初期スタートアップ時に、最初の 10 個のサンプリング時間からサンプリング時間の平均値が求められます。 現在のサンプリング時間とこの平均値の相違が大きすぎると、エラー(Error-Bits = 0800 hex)がトリガされます。

以下の条件が発生すると、PID_3Step は同調時に「無効」モードに設定されます。

- 新しい平均値 >= 1.1 x 古い平均値
- 新しい平均値 <= 0.9 x 古い平均値

自動モードでは、以下の条件が発生すると、PID_3Step は「無効」モードに設定されます。

- 新しい平均値 >= 1.5 x 古い平均値
- 新しい平均値 <= 0.5 x 古い平均値

PID アルゴリズムのサンプリング時間

コントロールされたシステムは、出力値の変更に応答するために、一定量の時間を必要とします。 このため、すべてのサイクルごとに出力値を計算することはお奨めしません。 PID アルゴリズムのサンプリング時間は、出力の 2 つの計算の間の時間を表わします。 それは同調時に計算され、サイクルタイムの倍数に四捨五入されます。 PID_3Step の他のすべてのファンクションは、呼び出しの都度実行されます。

モータ移行時間の測定

モータ移行時間は、モータがアクチュエータを閉状態から開状態に移行させるために必要な時間(秒)です。 アクチュエータが一方向に移動する最大時間は、モータ移行時間の 110%です。 PID_3Step は、最適なコントローラ結果を得るために、可能な限り正確なモータ移行時間を必要とします。 アクチュエータのマニュアルに記載されているデータは、このアクチュエータタイプの平均的な値です。 使用するアクチュエータによって、この値は異なる可能性があります。 モータ移行時間は、コミッショニング時に測定できます。 出力制限値は、モータ移行時間の測定中は考慮されません。 アクチュエータはエンドストップ上限または下限に移動できます。

制御論理

出力値を上げるのは通常、プロセス値を上げるために行われます。 これは、標準制御ロジックと呼ばれます。 冷却および放電制御システムでは、制御ロジックを反転させる必要があります。PID_3Step は、負の比例ゲインでは機能しません。 InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。 制御ロジックは、プリチューニングと微調整時にも考慮されます。

PID_3Step V1 の入力パラメータ

パラメータ	データタイプ	デフォルト	説明
Setpoint	REAL	0.0	自動モードでの PID コントローラのセットポイント
Input	REAL	0.0	ユーザープログラムの変数が、プロセス値のソースとして使用されます。 パラメータ Input を使用する場合は、Config.Input-PerOn = FALSE を設定する必要があります。
Input_PER	WORD	W#16#0	アナログ入力が、プロセス値のソースとして使用されます。 パラメータ Input_PER を使用する場合は、Con-fig.InputPerOn = TRUE を設定する必要があります。
Actuator_H	BOOL	FALSE	High エンドストップ用のバルブのデジタル位置フィードバック。 Actuator_H = TRUE の場合、バルブは High エンドストップの位置にあり、この方向にはもう動きません。
Actuator_L	BOOL	FALSE	Low エンドストップ用のバルブのデジタル位置フィードバック。 Actuator_L = TRUE の場合、バルブは Low エンドストップの位置にあり、この方向にはもう動きません。
Feedback	REAL	0.0	バルブの位置フィードバック。 パラメータ Feedback を使用する場合は、Con-fig.FeedbackPerOn = FALSE を設定する必要があります。
Feedback_PER	WORD	W#16#0	バルブ位置のアナログフィードバック。 パラメータ Feedback_PER を使用する場合は、Config.FeedbackPerOn = TRUE を設定する必要があります。 Feedback_PER は、以下の変数に基づいてスケーリングされます。 • Config.FeedbackScaling.LowerPointIn • Config.FeedbackScaling.UpperPointIn • Config.FeedbackScaling.LowerPointOut • Config.FeedbackScaling.UpperPointOut
ManualEnable	BOOL	FALSE	• FALSE -> TRUE エッジでは「手動モード」が選択され、State = 4, Retain.Mode はそのまま変わりません。

			• TRUE -> FALSE エッジによって、最も最近にアクティブだった動作モードが選択されます。 Retain.Mode の変更は、ManualEnable = TRUE 時には効果を持ちません。 Retain.Mode の変更は、ManualEnable での TRUE -> FALSE エッジ時に考慮されます。 **PID_3Step V1.1**: CPU の開始時に ManualEnable = TRUE の場合、PID_3Step が手動モードで開始されます。 ManualEnable での立ち上がりエッジ (FALSE -> TRUE)は不要です。 **PID_3Step V1.0** CPU の開始時に、ManualEnable での立ち上がりエッジ(FALSE->TRUE)の場合のみ、PID_3Step が手動モードに切り替わります。 立ち上がりエッジがない場合は、ManualEnable が FALSE であった最後の動作モードで PID_3Step が開始されます。
ManualValue	REAL	0.0	手動モードでは、バルブの絶対位置を指定します。ManualValue は、 OutputPer を使用するか、**または**、位置フィードバックが使用できる場合のみ、評価されます。
Manual_UP	BOOL	FALSE	手動モードでは、すべての立ち上がりエッジごとに、全コントロール範囲の 5%のみバルブが開くか、最小モータ遷移時間の間のみバルブが開きます。Manual_UP は、Output_PER を使用しないで、かつ、位置フィードバックが使用できない場合のみ、評価されます。
Manual_DN	BOOL	FALSE	手動モードでは、すべての立ち上がりエッジごとに、全コントロール範囲の 5%のみバルブが閉じるか、最小モータ遷移時間の間のみバルブが閉じます。Manual_DN は、Output_PER を使用しないで、かつ、位置フィードバックが使用できない場合のみ、評価されます。
Reset	BOOL	FALSE	コントローラを再起動します。 • FALSE -> TRUE エッジ ○ [無効]モードに切り替わります。 ○ コントローラの中間値がリセットされます。 (PID パラメータは保持されます) • TRUE -> FALSE エッジ 直近の有効なモードに切り替わります。

PID_3Step V1 の出力パラメータ

パラメータ	データタイプ	デフォルト	説明
ScaledInput	REAL	0.0	スケーリングされたプロセス値
ScaledFeedback	REAL	0.0	スケーリングされた位置フィードバック 位置フィードバックなしのアクチュエータの場合、ScaledFeedback で示されたアクチュエータの位置は極めて不正確です。この場合、ScaledFeedback は現在の位置のおおまかな評価にしか使用できません。
Output_UP	BOOL	FALSE	バルブを開くためのデジタル出力値。 Config.OutputPerOn = FALSE の場合、Output_UP パラメータが使用されます。
Output_DN	BOOL	FALSE	バルブを閉じるためのデジタル出力値。 Config.OutputPerOn = FALSE の場合、Output_DN パラメータが使用されます。
Output_PER	WORD	W#16#0	アナログ出力値 Config.OutputPerOn = TRUE の場合、Output_PER が使用されます。
SetpointLimit_H	BOOL	FALSE	SetpointLimit_H = TRUE の場合、セットポイント絶対上限値に達します。 CPU では、セットポイントは設定済みの絶対セットポイント上限値に制限されています。 設定済みのプロセス値絶対上限値は、セットポイント上限値のデフォルト値です。 Config.SetpointUpperLimit をプロセス制限範囲内の値に設定すると、この値がセットポイント上限値として使用されます。
SetpointLimit_L	BOOL	FALSE	SetpointLimit_L = TRUE の場合、セットポイント絶対下限値に達します。 CPU では、セットポイントは設定済みの絶対セットポイント下限値に制限されています。 設定済みのプロセス値絶対下限値は、セットポイント下限値の初期設定です。 Config.SetpointLowerLimit をプロセス制限範囲内の値に設定すると、この値がセットポイント下限値として使用されます。
InputWarning_H	BOOL	FALSE	InputWarning_H = TRUE の場合、プロセス値が警告上限値に到達済みか、警告上限値を超えています。
InputWarning_L	BOOL	FALSE	InputWarning_L = TRUE の場合、プロセス値が警告下限値に到達済みか、警告下限値未満になっています。
State	INT	0	State パラメータは、PID コントローラの現在の動作モードを

			示します。 Retain.Mode タグで動作モードを変更します。 • State = 0: 無効 • State = 1: プリチューニング • State = 2: 微調整 • State = 3: 自動モード • State = 4: 手動モード • State = 5: 代替出力値アプローチ • State = 6: 移行時間の測定 • State = 7: エラーモニタリング • State = 8: エラーモニタリング付きの代替出力値アプローチ
Error	BOOL	FALSE	Error = TRUE の場合、少なくとも 1 つのエラーが保留中です。
ErrorBits	DWORD	DW#16#0	ErrorBits パラメータは、エラーメッセージを示します。

PID_3Step V1 の静的タグ

リストにないタグを変更してはいけません。 これらは、システム内部の目的のためだけに使用されます。

タグ	データタイプ	デフォルト	説明
ActivateRecoverMode	BOOL	TRUE	ActivateRecoverMode タグは、エラーに対する応答を決定します。
RunModeByStartup	BOOL	TRUE	CPU のリスタート後にモードを有効化 RunModeByStartup = TRUE の場合、コントローラは CPU のリスタート後に最後に有効だった動作モードを返します。 RunModeByStartup = FALSE の場合、コントローラは CPU のリスタート後も非アクティブのままです。
PhysicalUnit	INT	0	プロセス値およびセットアポイントの測定単位(たとえば、℃または ºF)
PhysicalQuantity	INT	0	プロセス値およびセットポイントの物理量(たとえば、温度)
ErrorBehaviour	INT	0	ErrorBehaviour = 0 でエラーが発生すると、バルブはその現在位置に留まり、コントローラは「無効」モードまたは「エラーモニタ」モードに直接に切り替わります。 ErrorBehaviour = 1 でエラーが発生すると、アクチュエータが代替出力値まで移動した後にのみ、「無効」モードまたは「エラーモニタ」モードに切り替わります。 以下のエラーが発生すると、バルブを設定した代替出力値まで移動できません。 • 2000h: 「Feedback_PER」パラメータの値が無効です。 • 4000h: 「Feedback」パラメータの値が無効です。 • 8000h: デジタル位置フィードバック中のエラーです。
Warning	DWORD	DW#16#0	警告タグ には、リセット後または動作モードの最後の変更後に生成された警告が表示されます。 周期的警告(たとえば、プロセス値警告)が、警告の原因が取り除かれるまで、表示されます。 警告は、その原因がなくなると、すぐに自動的に削除されます。 周期的でない警告(たとえば、変曲点が見つからないなど)はそのままで、エラーと同様に削除されます。
SavePosition	REAL	0.0	代替出力値 ErrorBehaviour = 1 でエラーが発生すると、アクチュエータがプラントの安全な位置まで移動した後にのみ、「無効」モードに切り替わります。

CurrentSetpoint	REAL	0.0	現在アクティブなセットポイント。 この値は同調開始時に固定されます。
Progress	REAL	0.0	同調の進捗状況(パーセンテージ(0.0～100.0)で表示)
Config.InputPerOn	BOOL	TRUE	InputPerOn = TRUE の場合、Input_PER パラメータが使用されます。 InputPerOn = FALSE の場合、Input パラメータが使用されます。
Config.OutputPerOn	BOOL	FALSE	OutputPerOn = TRUE の場合、Output_PER パラメータが使用されます。 OutputPerOn = FALSE の場合、Ouput_UP および Output_DN パラメータが使用されます。
Config.LoadBackUp	BOOL	FALSE	LoadBackUp = TRUE の場合、最後の PID パラメータセットが再読み込みされます。 このセットは、最後の同調動作前に保存されたものです。LoadBackUp は自動的に FALSE にリセットされます。
Config.InvertControl	BOOL	FALSE	反転制御ロジック InvertControl = TRUE の場合、制御偏差が大きくなると出力値が低下します。
Config.FeedbackOn	BOOL	FALSE	FeedbackOn = FALSE の場合、位置フィードバックがシミュレートされます。 位置フィードバックは通常、FeedbackOn = TRUE の場合に有効になります。
Config.FeedbackPerOn	BOOL	FALSE	FeedbackPerOn は、FeedbackOn = TRUE のときのみ有効です。 FeedbackPerOn = TRUE の場合、アナログ入力が位置フィードバック(Feedback_PER パラメータ)として使用されます。 FeedbackPerOn = FALSE の場合、Feedback パラメータが位置フィードバックとして使用されます。
Config.ActuatorEndStopOn	BOOL	FALSE	ActuatorEndStopOn = TRUE の場合、位置フィードバック Actuator_L および Actuator_H が考慮されます。
Config.InputUpperLimit	REAL	120.0	プロセス値の上限値 I/O 入力で、プロセス値が標準範囲よりも最大 18%大きくなる場合があります(範囲オーバー)。「プロセス値の上限値」オーバーのエラーは通知されません。 断線および短絡のみが認識され、PID_3Step が設定されたエラーに対する応答に従って応答します。 InputUpperLimit > InputLowerLimit
Config.InputLowerLimit	REAL	0.0	プロセス値の下限値 InputLowerLimit < InputUpperLimit
Config.InputUpperWarning	REAL	+3.402822e+38	プロセス値の警告上限値 プロセス値制限範囲外の InputUpperWarning を設定すると、設定されたプロセス値絶対上限値が警告上限値として使用されます。 プロセス値制限範囲内の InputUpperWarning を設定すると、この値が警告上限値として使用されます。 InputUpperWarning > InputLowerWarning

			InputUpperWarning ≤ InputUpperLimit
Config.InputLowerWarning	REAL	-3.402822e+38	プロセス値の警告下限値 プロセス値制限範囲外の InputLowerWarning を設定すると、設定されたプロセス値絶対下限値が警告下限値として使用されます。 プロセス値制限範囲内の InputLowerWarning を設定すると、この値が警告下限値として使用されます。 InputLowerWarning < InputUpperWarning InputLowerWarning ≥ InputLowerLimit
Config.OutputUpperLimit	REAL	100.0	出力値の上限値 詳細は、OutputLowerLimit を参照してください。
Config.OutputLowerLimit	REAL	0.0	出力値の下限値 OutputPerOn = TRUE または FeedbackOn = TRUE の場合、-100% ~ +100%の値の範囲(0 を含めて)が有効です。 -100%では Output = -27648、+100%では Output = 27648 です。 OutputPerOn = FALSE の場合、0% ~ 100%の値の範囲が有効です。 バルブは 0%で完全に閉じ、100%で完全に開きます。
Config.SetpointUpperLimit	REAL	+3.402822e+38	セットポイントの上限値 プロセス値制限範囲外の SetpointUpperLimit を設定すると、設定されたプロセス値絶対上限値がセットポイント上限値として事前割り当てされます。 プロセス値制限範囲内の SetpointUpperLimit を設定すると、この値がセットポイント上限値として使用されます。
Config.SetpointLowerLimit	REAL	-3.402822e+38	セットポイントの下限値 プロセス値制限範囲外の SetpointLowerLimit を設定すると、設定されたプロセス値絶対下限値がセットポイント下限値として事前割り当てされます。 プロセス値制限範囲内の SetpointLowerLimit を設定すると、この値がセットポイント下限値として使用されます。
Config.MinimumOnTime	REAL	0.0	最小 ON 時間 サーボドライブをスイッチオンにしていなければならない最小時間(秒単位)。
Config.MinimumOffTime	REAL	0.0	最小 OFF 時間 サーボドライブをスイッチオフにしていなければならない最小時間(秒単位)。
Config.TransitTime	REAL	30.0	モータ移行時間 作動ドライブがバルブを閉状態から開状態に遷移させるために必要な時間(秒単位)。
Config.InputScaling.UpperPointIn	REAL	27648.0	高 Input_PER のスケーリング

			Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.LowerPointIn	REAL	0.0	低 Input_PER のスケーリング Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.UpperPointOut	REAL	100.0	スケーリングされた High プロセス値 Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.InputScaling.LowerPointOut	REAL	0.0	スケーリングされた Low プロセス値 Input_PER は、InputScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.UpperPointIn	REAL	27648.0	高 Feedback_PER のスケーリング Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.LowerPointIn	REAL	0.0	低 Feedback_PER のスケーリング Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.UpperPointOut	REAL	100.0	エンドストップ上限 Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
Config.FeedbackScaling.LowerPointOut	REAL	0.0	エンドストップ下限 Feedback_PER は、FeedbackScaling 構造体の 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてパーセントに変換されます。
GetTransitTime.InvertDirection	BOOL	FALSE	InvertDirection = FALSE の場合、バルブ移行時間を求めるために、バルブが完全に開き、閉じた後、再度開きます。 InvertDirection = TRUE の場合、バルブが完全に閉じ、開いた後、再度閉じます。
GetTransitTime.SelectFeedback	BOOL	FALSE	SelectFeedback = TRUE の場合、Feedback_PER となるか、または Feedback が移行時間の測定で考慮されます。 SelectFeedback = FALSE の場合、Actuator_H および Actuator_L が移行時間の測定で考慮されます。
GetTransitTime.Start	BOOL	FALSE	Start = TRUE の場合、移行時間の測定が開始されます。

GetTransitTime.State	INT	0	移行時間測定の現在のフェーズ • State = 0: 無効 • State = 1: バルブを完全に開きます • State = 2: バルブを完全に閉じます • State = 3: バルブをターゲット位置に移動します (NewOutput) • State = 4: 移行時間の測定が正常に終了しました • State = 5: 移行時間の測定がキャンセルされました
GetTransitTime.New-Output	REAL	0.0	位置フィードバック付きの移行時間測定のターゲット位置 ターゲット位置は、「エンドストップ上限」と「エンドストップ下限」の間でなければなりません。 NewOutput と ScaledFeedback の差は、許容制御範囲の少なくとも 50%でなければなりません。
CycleTime.StartEstimation	BOOL	TRUE	StartEstimation = TRUE の場合、PID_3Step サンプリング時間の測定が開始されます。CycleTime.StartEstimation = FALSE の場合はすぐに測定が完了します。
CycleTime.EnEstimation	BOOL	TRUE	EnEstimation = TRUE の場合、PID_3Step サンプリング時間が計算されます。
CycleTime.EnMonitoring	BOOL	TRUE	EnMonitoring = TRUE の場合、PID_3Step サンプリング時間がモニタされます。 サンプリング時間以内に PID_3Step を実行できない場合は、エラー 0800h が出力され、動作モードが変更されます。 ActivateRecoverMode および ErrorBehaviour が、切り替え先動作モードを決定します。 EnMonitoring = FALSE の場合、PID_3Step サンプリング時間はモニタされず、エラー 0800h は出力されず、動作モードは変更されません。
CycleTime.Value	REAL	0.1	PID_3Step サンプリング時間(秒単位) CycleTime.Value が自動的に測定されます。通常、呼び出し OB のサイクルタイムと同じ値です。
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Retain.CtrlParams.SetByUser の保存された値。 Config.LoadBackUp = TRUE で、CtrlParamsBackUp 構造体から値を再読み込みできます。
CtrlParamsBackUp.Gain	REAL	1.0	比例ゲインの保存エリア
CtrlParamsBackUp.Ti	REAL	20.0	積分動作時間の保存エリア
CtrlParamsBackUp.Td	REAL	0.0	微分動作時間の保存エリア
CtrlParamsBackUp.TdFiltRatio	REAL	0.0	微分の遅延係数の保存エリア
CtrlParamsBackUp.PWeighting	REAL	0.0	比例動作重みの保存エリア
CtrlParamsBackUp.DWeighting	REAL	0.0	微分動作重みの保存エリア
CtrlParamsBackUp.Cycle	REAL	1.0	PID アルゴリズムのサンプリング時間の保存エリア

CtrlParamsBackUp.InputDeadBand	REAL	0.0	制御偏差のデッドバンド幅の保存エリア
PIDSelfTune.SUT.CalculateSUTParams	BOOL	FALSE	制御システムのプロパティは同調中に保存されます。CalculateSUTParams = TRUE の場合、PID パラメータがこれらのプロパティに基づいて再計算されます。TuneRuleSUT で設定された方法を使用して、PID パラメータが計算されます。計算の後に、CalculateSUTParams が FALSE に設定されます。
PIDSelfTune.SUT.TuneRuleSUT	INT	1	プリチューニング時のパラメータの計算に使用される方法: • TuneRuleSUT = 0: PID 高速 I • TuneRuleSUT = 1: PID 低速 I • TuneRuleSUT = 2: Chien、Hrones、および Reswick PID • TuneRuleSUT = 3TuneRuleSUT = 3: Chien、Hrones、Reswick PI • TuneRuleSUT = 4: PID 高速 II • TuneRuleSUT = 5: PID 低速 II
PIDSelfTune.SUT.State	INT	0	SUT.State タグは、プリチューニングの現在のフェーズを示します。
PIDSelfTune.TIR.RunIn	BOOL	FALSE	• RunIn = FALSE 無効なモードまたは手動モードで微調整が開始されると、プリチューニングが開始されます。 自動モードで微調整が開始されると、システムは既存の PID パラメータを使用してセットポイントを制御します。 この場合のみ、微調整が開始されます。 プリチューニングが可能でない場合、PID_3Step は「無効」モードに切り替わります。 • RunIn = TRUE プリチューニングはスキップされます。 PID_3Step は、最小または最大出力値でセットポイントに到達することを試みます。この結果、オーバーシュートが増大する可能性があります。 この場合のみ、微調整が開始されます。 微調整後、RunIn が FALSE に設定されます。
PIDSelfTune.TIR.CalculateTIRParams	BOOL	FALSE	制御システムのプロパティは同調中に保存されます。CalculateTIRParams = TRUE の場合、PID パラメータがこれらのプロパティに基づいて再計算されます。TuneRuleTIR で設定された方法を使用して、PID パラメータが計算されます。計算の後に、CalculateTIRParams が FALSE に設定されます。
PIDSelfTune.TIR.TuneRuleTIR	INT	0	微調整時のパラメータの計算に使用される方法: • TuneRuleTIR = 0: PID (自動) • TuneRuleTIR = 1: PID (高速) • TuneRuleTIR = 2: PID (低速) • TuneRuleTIR = 3: Ziegler-Nichols PID

			• TuneRuleTIR = 4: Ziegler-Nichols PI • TuneRuleTIR = 5: Ziegler-Nichols P
PIDSelfTune.TIR.State	INT	0	TIR.State タグは、「微調整」の現在のフェーズを示します。
Retain.Mode	INT	0	Retain.Mode の値の変更は、別の動作モードへの切り替えをトリガします。 Mode を以下に変更すると、以下の動作モードが有効になります。 • Mode = 0: 無効 • Mode = 1: プリチューニング • Mode = 2: 微調整 • Mode = 3: 自動モード • Mode = 4: 手動モード • Mode = 5: 代替出力値アプローチ • Mode = 6: 移行時間の測定 • Mode = 7Mode = 7: エラーモニタリング • Mode = 8: エラーモニタリング付きの代替出力値アプローチ Mode は保持されます。
Retain.CtrlParams.SetByUser	BOOL	FALSE	SetByUser = FALSE の場合、PID パラメータが自動的に決定され、PID_3Step は出力値でのデッドバンドありで動作します。 デッドバンド幅は、同調時に出力値の標準偏差に基づいて計算され、Retain.CtrlParams.OutputDeadBand に保存されます。 SetByUser = TRUE の場合、PID パラメータは手動で入力され、PID_3 Step は出力値でのデッドバンドなしで動作します。Retain.CtrlParams.OutputDeadBand = 0.0 SetByUser は保持されます。
Retain.CtrlParams.Gain	REAL	1.0	有効な比例ゲイン Gain は保持されます。
Retain.CtrlParams.Ti	REAL	20.0	• Ti > 0.0: 有効な積分動作時間 • Ti = 0.0: 積分動作が無効です Ti は保持されます。
Retain.CtrlParams.Td	REAL	0.0	• Td > 0.0: 有効な微分動作時間 • Td = 0.0: 微分動作が無効です Td は保持されます。
Retain.CtrlParams.TdFiltRatio	REAL	0.0	動作中の微分の遅延係数 TdFiltRatio は保持されます。
Retain.CtrlParams.PWeighting	REAL	0.0	有効な比例動作重み PWeighting は保持されます。
Retain.CtrlParams.DWeighting	REAL	0.0	有効な微分動作重み DWeighting は保持されます。

Retain.CtrlParams.Cycle	REAL	1.0	有効な PID アルゴリズムのサンプリング時間(秒単位)。呼び出し OB のサイクルタイムの整数倍に丸められます。 Cycle は保持されます。
Retain.CtrlParams.Input-DeadBand	REAL	0.0	制御偏差のデッドバンド幅 InputDeadBand は保持されます。

注記
PID コントローラの誤動作を防止するには、「無効」モードでこの表に記載されているタグを変更します。「Retain.Mode」タグを「0」に設定すると、「無効」モードが強制されます。

パラメータ State と Retain.Mode V1

パラメータの相互関係

State パラメータは、PID コントローラの現在の動作モードを示します。 State パラメータを変更することはできません。

動作モードを切り替えるには、Retain.Mode タグを変更する必要があります。 これは、新しい動作モードの値が Retain.Mode 内に存在するときにも適用されます。 たとえば、まず Retain.Mode = 0 を設定してから Retain.Mode = 3 を設定します。 コントローラの現在の動作モードでこの変更が行える場合、State は、Retain.Mode の値に設定されます。

PID_3Step が自動的に動作モードを切り替えるときは、以下が適用されます。 State != Retain.Mode.

例:

- 成功したプリチューニング後
  State = 3 および Retain.Mode = 1
- エラーが発生した場合
  State = 0 および Retain.Mode は前の値、たとえば Retain.Mode = 3 にとどまります。
- ManualEnalbe = TRUE
  State = 4 および Retain.Mode は前の値、たとえば Retain.Mode = 3 にとどまります。

注記
たとえば、Mode = 0 で自動モードを終了しないで、成功した微調整を繰り返すこともできます。

1 つのサイクルについて、Retain.Mode を 9999 などの不正な値に設定すると、State に対して効果を持ちません。 次のサイクルで、Mode = 2 を設定します。 このように、最初に「無効」モードに切り替えなくても、Retain.Mode を変更することができます。

値の意味

State / Retain.Mode	説明
0	無効 コントローラはオフになり、バルブ位置の変更をもう行いません。
1	プリチューニング プリチューニングでは、出力値のパルスに対するプロセス応答が決定され、変曲点が検索されます。 最適化 PID パラメータは、コントロールされるシステムの最大上昇率とデッドタイムの関数として計算されます。 プリチューニングの必要条件 • State = 0 または State = 4 • ManualEnable = FALSE • モータ移行時間が設定または測定されていること。 • セットポイントとプロセス値が、設定済みの制限範囲内であること。

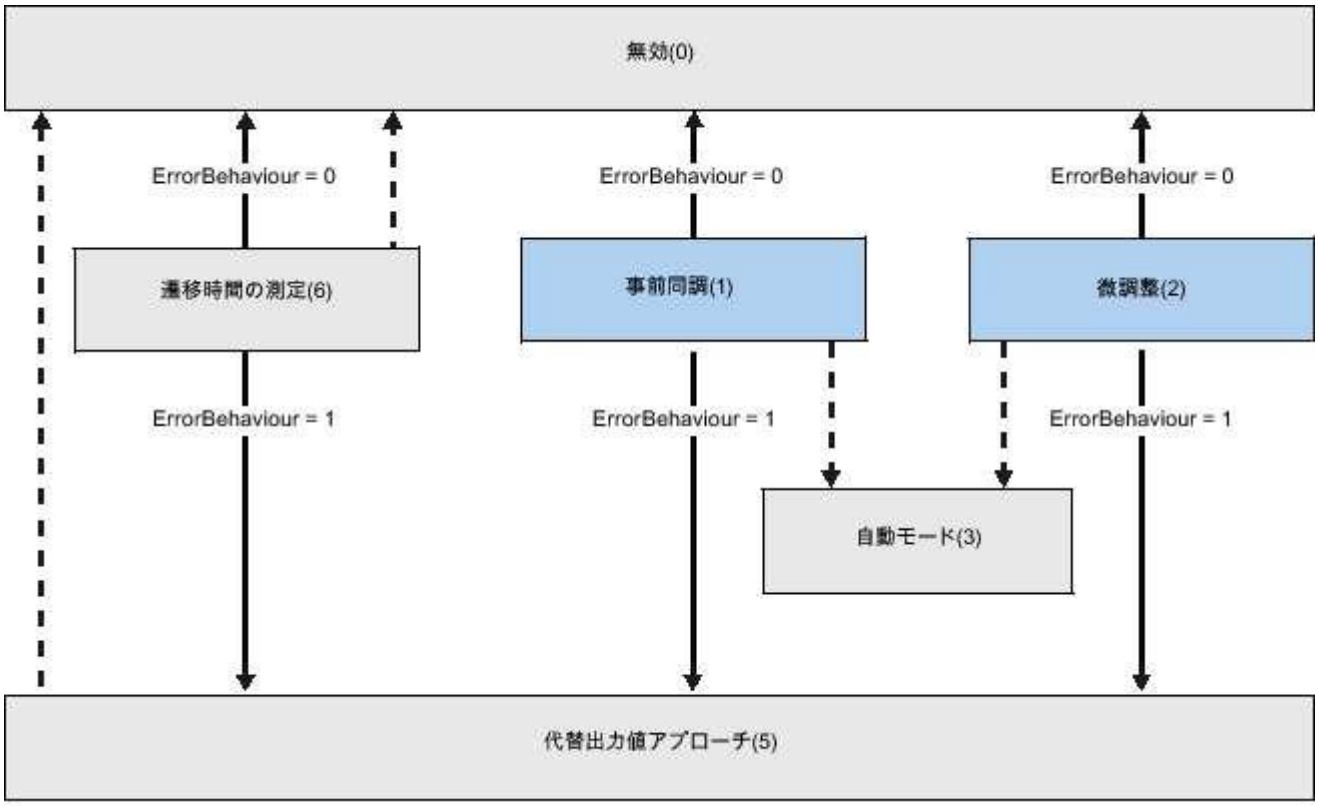
	プロセス値が安定しているほど、PID パラメータの計算が簡単になり、結果が正確になります。 プロセス値の上昇率がノイズと比べて大幅に高ければ、プロセス値のノイズは許容できます。 PID パラメータは再計算される前にバックアップされ、Config.LoadBackUp で再度有効になります。 セットポイントが、CurrentSetpoint タグで固定されます。 コントローラはプリチューニングが成功すると自動モードに切り替わり、プリチューニングが失敗すると「無効」モードに切り替わります。 プリチューニングフェーズは、SUT.State タグで示されます。
2	微調整 微調整は、プロセス値の一定の制限された振動を生成します。 PID パラメータは、この振動の振幅と周波数に基づいて同調されます。 プリチューニング時と微調整時のプロセス応答間の相違が分析されます。 すべての PID パラメータはこの結果から計算されます。 通常は微調整に基づく PID パラメータの方が、プリチューニングに基づく PID パラメータよりも優れたマスタコントロールと外乱特性を実現します。 PID_3Step は、プロセス値のノイズよりも大きな振動の生成を自動的に試行します。 プロセス値の安定性によって微調整が受ける影響は最小限です。 PID パラメータは、微調整の前に、バックアップされます。 PID パラメータは、Config.LoadBackUp で再度有効にすることができます。 セットポイントが、CurrentSetpoint タグで固定されます。 微調整の必要条件 • モータ移行時間が設定または測定されていること。 • セットポイントとプロセス値が、設定済みの制限範囲内であること。 • ManualEnable = FALSE • 自動(State = 3)、無効(State = 0)、または手動(State = 4)モードであること。 微調整は、開始されたときのモードに従って以下のように実行されます。 • 自動モード(State = 3) 同調を使用して既存の PID パラメータを改善する場合は、自動モードで微調整を開始します。 PID_3Step は、制御ループが安定し、微調整の必要条件が満たされるまでは、既存の PID パラメータを使用してシステムを制御します。 制御ループが安定して微調整の必要条件が満たされた場合のみ、微調整が開始します。 • 無効モード(State = 0)または手動モード(State = 4) プリチューニングは、常に最初に開始されます。 制御ループが安定し、微調整の必要条件が満たされるまでは、決定された PID パラメータが制御に使用されます。 PIDSelfTune.TIR.RunIn = TRUE の場合、プリチューニングがスキップされ、最小または最大出力値によるセットポイントへの到達が試みられます。 この結果、オーバーシュートが増大する可能性があります。 微調整が自動的に開始します。 微調整が正常に終了すると、コントローラが自動モードに切り替わります。 微調整が失敗した場合、コントローラは「無効」モードに切り替わります。 微調整フェーズは、TIR.State タグで示されます。
3	自動モード 自動モードでは、PID_3Step は、指定されたパラメータに従って、コントロールされるシステムを制御します。

	以下の必要条件の 1 つが満たされた場合、コントローラは自動モードに切り替わります。 • プリチューニングの正常終了 • 微調整の正常終了 • Retain.Mode タグを値 3 に変更 CPU がオンになるか、Stop モードから RUN モードに切り替わった場合、PID_3Step は直近の有効な動作モードで開始されます。 PID_3Step を「無効」モードのままにするには、RunModeByStartup = FALSE を設定してください。 ActivateRecoverMode タグは、自動モードで考慮されます。
4	手動モード 手動モードでは、Manual_UP および Manual_DN パラメータまたは ManualValue パラメータで、手動出力値を指定します。 エラー発生時にアクチュエータを出力値まで移動できるかどうかは、ErrorBits パラメータに記載されています。 この動作モードは Retain.Mode = 4 の場合、または ManualEnable の立ち上がりエッジで有効になります。 ManualEnable が TRUE に変わると、 State だけが変更されます。Retain.Mode は現在の値を保持します。 ManualEnable の立ち下りエッジでは、PID_3Step は直前の動作モードに戻ります。 この自動モードへの変更はバンプなしです。 **PID_3Step V1.1** エラー発生時は、手動モードは常に可能です。 **PID_3Step V1.0** エラー発生時は、手動モードは ActivateRecoverMode タグに依存します。
5	代替出力値アプローチ この動作モードは、エラーが発生した場合、または Reset = TRUE 時に Errorbehaviour = 1 および ActivateRecoverMode = FALSE.の場合、有効になります。 PID_3Step は、アクチュエータを代替出力値まで移動した後、「無効」モードに切り替わります。
6	移行時間の測定 バルブを閉じた状態から完全に開くためにモータが必要とする時間が特定されます。 この動作モードは、GetTransitTime.Start = TRUE が設定されたときに有効になります。 移行時間の測定にエンドストップ信号を使用すると、バルブは、現在の位置から完全に開き、完全に閉じ、再度完全に開きます。 GetTransitTime.InvertDirection = TRUE の場合、この動作は逆転します。 移行時間の測定に位置フィードバックを使用すると、アクチュエータが、現在の位置からターゲット位置へ移動されます。 出力制限値は、移行時間の測定中は考慮されません。 アクチュエータはエンドストップ上限または下限に移動できます。
7	エラーモニタリング コントロールアルゴリズムはオフになり、バルブ位置の移動をもう行いません。 この動作モードは、エラー発生時、「無効」モードの代わりに有効になります。 以下のすべての条件を満たさなければなりません。

	• Mode = 3 (自動モード) • Errorbehaviour = 0 • ActivateRecoverMode = TRUE • ActivateRecoverMode が有効になる 1 つまたは複数のエラーが発生しました。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。
8	エラーモニタリング付きの代替出力値アプローチ この動作モードは、エラーが発生した時に「代替出力値アプローチ」モードの代わりに有効になります。PID_3Step は、アクチュエータを代替出力値まで移動し、「エラーのモニタ」モードに切り替わります。 以下のすべての条件を満たさなければなりません。 • Mode = 3 (自動モード) • Errorbehaviour = 1 • ActivateRecoverMode = TRUE • ActivateRecoverMode が有効になる 1 つまたは複数のエラーが発生しました。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。

コミッショニング時の動作モードの自動切り替え

PID_3Step は、エラー発生時に自動的に動作モードを切り替えます。 次の図は、移行時間測定、プリチューニング、および微調整モードからの動作モードの切り替えに対する ErrorBehaviour の影響を示します。

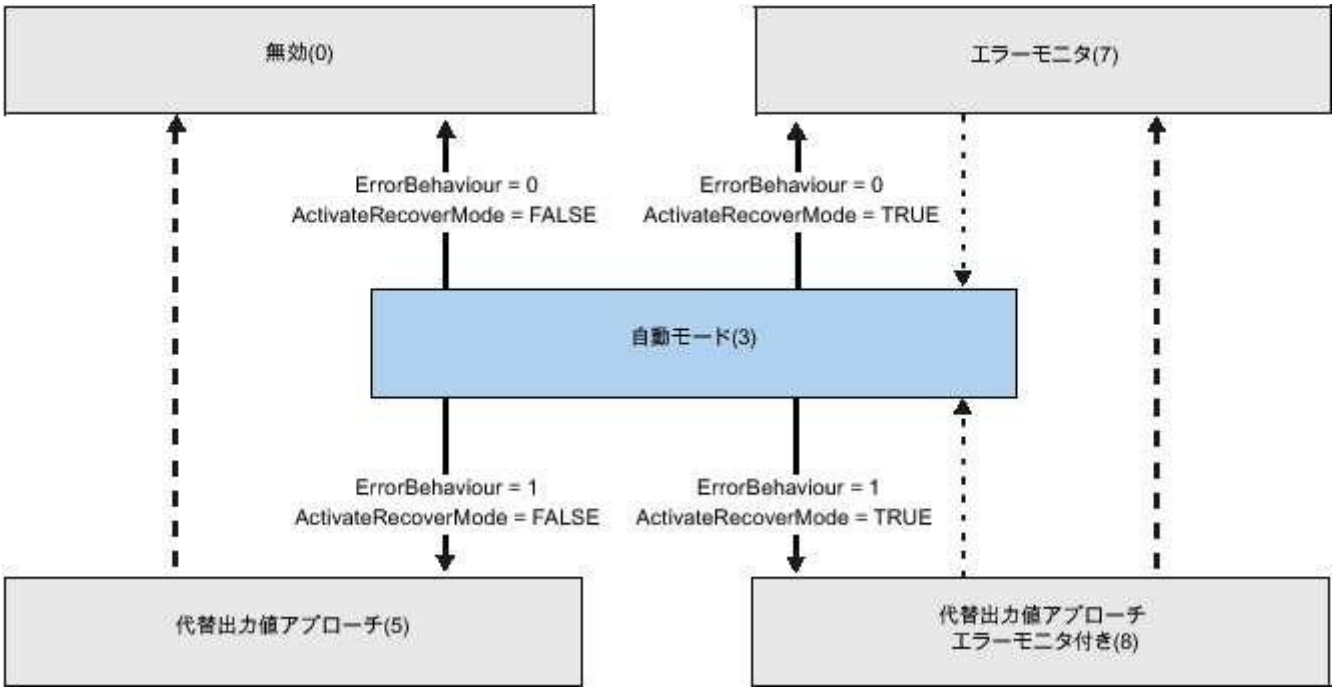


エラー発生時の動作モードの自動変更

現在の動作が完了したときの動作モードの自動変更

自動モードでの動作モードの自動切り替え(PID_3Step V1.1)

PID_3Step は、エラー発生時に自動的に動作モードを切り替えます。 次の図は、動作モードの変更に対する ErrorBehaviour および ActivateRecoverMode の影響を示します。

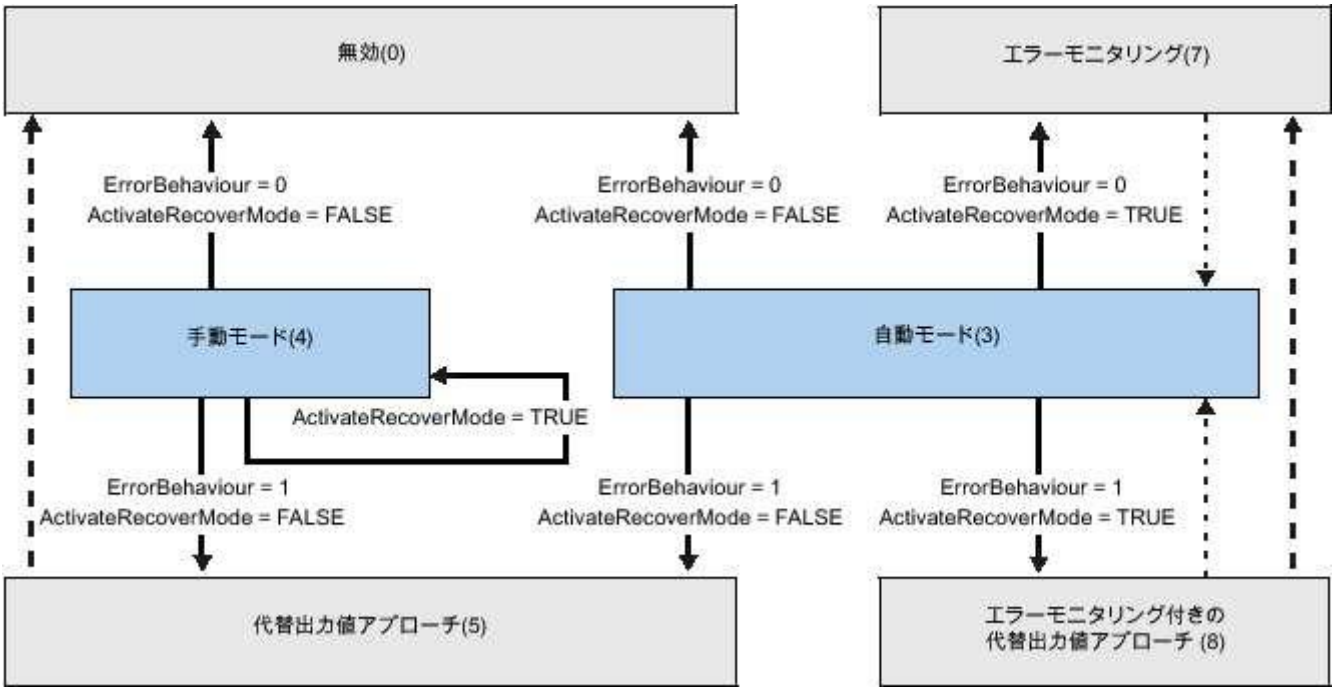


エラー発生時の動作モードの自動変更

現在の動作が完了したときの動作モードの自動変更

自動モードおよび手動モードでの動作モードの自動切り替え(PID_3Step V1.0)

PID_3Step は、エラー発生時に自動的に動作モードを切り替えます。 次の図は、動作モードの変更に対する ErrorBehaviour および ActivateRecoverMode の影響を示します。



エラー発生時の動作モードの自動変更

現在の動作が完了したときの動作モードの自動変更

エラーが保留中でない場合の動作モードの自動変更

パラメータ ErrorBits V1

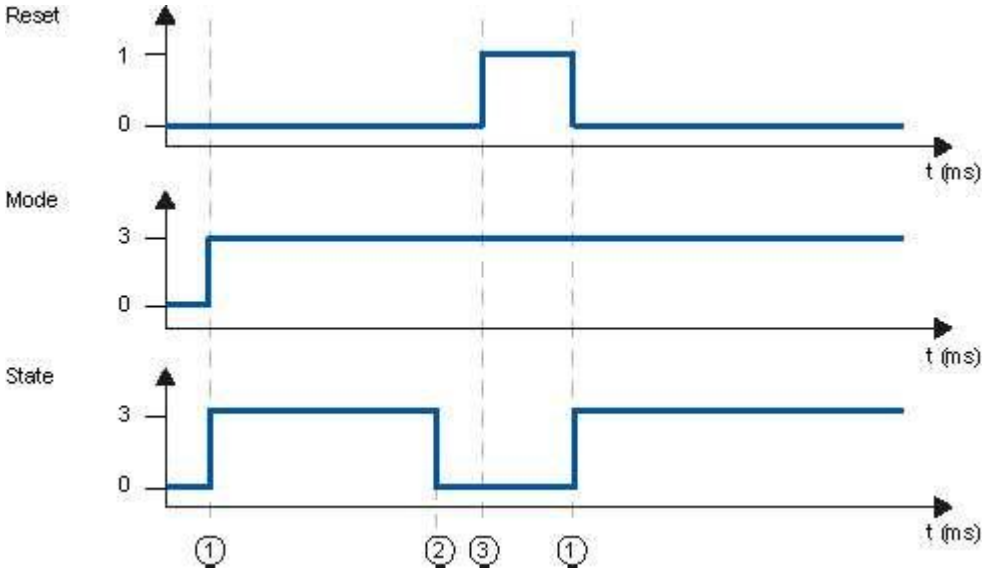
複数のエラーが同時に保留中の場合、エラーコードの値がバイナリ加算で表示されます。 たとえば、エラーコード 0003 の表示は、エラー 0001 と 0002 が同時に保留中であることを示します。

ErrorBits (DW#16#...)	説明
0000	エラーはありません。
0001	「Input」パラメータがプロセス値制限範囲外です。 • Input > Config.InputUpperLimit または • Input < Config.InputLowerLimit ActivateRecoverMode = TRUE および ErrorBehaviour = 1 の場合、アクチュエータは代替出力値まで移動します。 ActivateRecoverMode = TRUE および ErrorBehaviour = 0 の場合、アクチュエータはその現在位置で停止します。 ActivateRecoverMode = FALSE の場合、アクチュエータはその現在位置で停止します。 PID_3Step V1.1 アクチュエータは手動モードで移動することができます。 PID_3Step V1.0 この状態では、手動モードは可能ではありません。 エラーを取り除くまでは、アクチュエータを移動できません。
0002	「Input_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。
0004	微調整中のエラー プロセス値の振幅を維持できませんでした。
0020	プリチューニングを自動モードで、または微調整中に行うことはできません。
0080	プリチューニング中のエラー 出力制限値の設定が不正です。 出力値の制限値が正しく設定されていて、制御ロジックと一致しているかどうかをチェックします。
0100	微調整時のエラーによって、パラメータが無効になりました。
0200	「Input」パラメータの値が無効です。 値が無効な番号フォーマットです。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。
0400	出力値の計算が失敗しました。 PID パラメータをチェックしてください。
0800	サンプリング時間エラー: PID_3Step が、サイクル割り込み OB のサンプリング時間内に呼び出されていません。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。
1000	「Setpoint」パラメータの値が無効です。 値が無効な番号フォーマットです。

	エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。
2000	「Feedback_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。 アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。 この状態では、手動モードは可能ではありません。 アクチュエータをこの状態から解除するには、位置フィードバックを無効にする必要があります(Config. FeedbackOn = FALSE)。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。
4000	「Feedback」パラメータの値が無効です。 値が無効な番号フォーマットです。 アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。この状態では、手動モードは可能ではありません。 アクチュエータをこの状態から解除するには、位置フィードバックを無効にする必要があります(Config. FeedbackOn = FALSE)。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。
8000	デジタル位置フィードバック中のエラーです。 Actuator_H = TRUE および Actuator_L = TRUE です。 アクチュエータを代替出力値まで移動できず、アクチュエータは現在位置のままです。 この状態では、手動モードは可能ではありません。 アクチュエータをこの状態から解除するには、[アクチュエータのエンドストップ]を無効にする必要があります(Config.ActuatorEndStopOn = FALSE)。 エラーが発生する前に自動モードが有効で、ActivateRecoverMode = TRUE で、かつ、エラーが保留中でない場合、PID_3Step は自動モードに戻ります。

パラメータ Reset V1

Reset での立ち上がりはエラーおよび警告をリセットし、積分動作を解除します。 Reset での立ち下がりは、直近の有効な動作モードへの変更をトリガします。



① 有効化

② Error

③ Reset

タグ ActivateRecoverMode V1

ActivateRecoverMode 変数の効果は、PID_3Step のバージョンに応じて異なります。

バージョン 1.1 の動作

ActivateRecoverMode 変数は、自動モードでのエラー発生時の動作を決定します。ActivateRecoverMode は、事前同調、微調整、および遷移時間測定時には無効です。

ActivateRecoverMode	説明
FALSE	エラー発生時には、PID_3Step は、「無効」または「代替出力値アプローチ」動作モードに切り替わります。 コントローラは、Retain.Mode のリセットまたは変更によって有効になります。
TRUE	自動モードでエラーが頻繁に発生すると、この設定はコントロール応答に対してマイナスの効果を持ちます。 この場合は、ErrorBits パラメータをチェックして、エラーの原因を取り除いてください。 1 つまたは複数のエラーが発生する場合、PID_3Step は、「エラーモニタ付き代替出力値アプローチ」または「エラーモニタ」モードに切り替わります。 • 0002h:パラメータ「Input_PER」の値が無効です。 • 0200h:パラメータ「Input」の値が無効です。 • 0800h:サンプリング時間のエラー • 1000h:パラメータ「Setpoint」の値が無効です。 • 2000h:パラメータ「Feedback_PER」の値が無効です。 • 4000h:パラメータ「Feedback」の値が無効です。 • 8000h: デジタル位置フィードバックのエラーです。 エラー 2000h、4000h、および 8000h の場合、PID_3Step は、設定された代替出力値にアプローチ**できません**。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。

バージョン 1.0 の動作

ActivateRecoverMode 変数は、自動および手動モードでのエラー発生時の動作を決定します。ActivateRecoverMode は、事前同調、微調整、および遷移時間測定時には無効です。

ActivateRecoverMode	説明
FALSE	エラー発生時には、PID_3Step は、「無効」または「代替出力値アプローチ」動作モードに切り替わります。 コントローラは、Retain.Mode のリセットまたは変更によって有効になります。
TRUE	**自動モードでのエラー** 自動モードでエラーが頻繁に発生すると、この設定はコントロール応答に対してマイナスの効果を持ちます。 この場合は、ErrorBits パラメータをチェックして、エラーの原因を取り除いてください。

	1 つまたは複数のエラーが発生する場合、PID_3Step は、「エラーモニタ付き代替出力値アプローチ」または「エラーモニタ」モードに切り替わります。 • 0002h:パラメータ「Input_PER」の値が無効です。 • 0200h:パラメータ「Input」の値が無効です。 • 0800h:サンプリング時間のエラー • 1000h:パラメータ「Setpoint」の値が無効です。 • 2000h:パラメータ「Feedback_PER」の値が無効です。 • 4000h:パラメータ「Feedback」の値が無効です。 • 8000h: デジタル位置フィードバックのエラーです。 エラー 2000h、4000h、および 8000h の場合、PID_3Step は、設定された代替出力値にアプローチ**できません**。 保留中のエラーがなくなると、PID_3Step は直ちに自動モードに戻ります。 **手動モードでのエラー** 以下のエラーの 1 つまたは複数が発生しても、PID_3Step は手動モードのままです。 • 0002h:パラメータ「Input_PER」の値が無効です。 • 0200h:パラメータ「Input」の値が無効です。 • 0800h:サンプリング時間のエラー • 1000h:パラメータ「Setpoint」の値が無効です。 • 2000h:パラメータ「Feedback_PER」の値が無効です。 • 4000h:パラメータ「Feedback」の値が無効です。 • 8000h: デジタル位置フィードバックのエラーです。 エラー 2000h、4000h、および 8000h の場合、バルブを適切な位置まで移動**できません**。

タグ Warning V1

複数の警告が同時に保留中の場合、その値がバイナリ加算で表示されます。 警告 0003 の表示は、たとえば、警告 0001 と警告 0002 が同時に保留中であることを示します。

Warning (DW#16#...)	説明
0000	保留中の警告がありません。
0001	プリチューニング時に変曲点が見つかりませんでした。
0002	微調整時に振動が増加しました。
0004	セットポイントが、設定済みの制限値に制限されました。
0008	選択された計算方法で、コントロールされるシステムの必要なプロパティの中に定義されなかったものがあります。 代わりに、「TuneRuleTIR = 3」方法を使用して、PID パラメータが起算されました。
0010	ManualEnable = TRUE のために、動作モードを変更できませんでした。
0020	呼び出し OB のサイクルタイムが、PID アルゴリズムのサンプリング時間を制限します。 さらに短い OB サイクルタイムを使用して、結果を改善してください。
0040	プロセス値が、その警告制限値の 1 つを超えました。
0080	Retain.Mode での値が不正です。 動作モードは変更されません。
0100	手動値が、コントローラ出力の制限値に制限されました。
0200	同調で使用されるルールが、不正な結果をもたらすか、サポートされていません。
0400	移行時間測定で選択された方法が、アクチュエータに適合しません。 アクチュエータの設定が選択された測定方法と適合しないため、移行時間を測定できません。
0800	現在の位置と新しい出力値の間の相違が、移行時間測定には小さすぎます。 これにより、不正な結果を生じる場合があります。 現在の出力値と新しい出力値の相違は、制御範囲全体の少なくとも 50%でなければなりません。
1000	代替出力値が出力制限値を超えているため、代替出力値に達することができません。

以下の警告は、原因が取り除かれるとすぐに削除されます。

- 0004
- 0020
- 0040
- 0100

他のすべての警告は、Reset での立ち上がりエッジで解除されます。

タグ SUT.State V1

SUT.State	名前	説明
0	SUT_INIT	事前同調の初期化
50	SUT_TPDN	位置フィードバックなしでの開始位置の決定
100	SUT_STDABW	標準偏差の計算
200	SUT_GET_POI	インフレクションポイントの検索
300	SUT_GET_RISETM	立ち上がり時間の決定
9900	SUT_IO	事前同調が成功
1	SUT_NIO	事前同調が失敗

タグ TIR.State V1

TIR.State	名前	説明
-100	TIR_FIRST_SUT	微調整が実行可能ではありません。 事前同調が最初に実行されます。
0	TIR_INIT	微調整の初期化
200	TIR_STDABW	標準偏差の計算
300	TIR_RUN_IN	最大または最小出力値によるセットポイントへの到達の試み
400	TIR_CTRLN	既存の PID パラメータを使用した、セットポイントへの到達の試み (事前同調が成功した場合)
500	TIR_OSZIL	振動の決定とパラメータの計算
9900	TIR_IO	微調整が成功
1	TIR_NIO	微調整が失敗

PID_Temp

この章には下記に関する情報が記載されています：

- CPU および FW との互換性 (S7-1200, S7-1500)
- PID_Temp V1 の CPU 処理時間およびメモリ要件 (S7-1200, S7-1500)
- PID_Temp (S7-1200, S7-1500)

CPU および FW との互換性

下の表に、PID_Temp のどのバージョンがどの CPU で使用できるかを示します。

CPU	FW	PID_Temp
S7-1200	≥ V4.1	V1.0
S7-1500	≥ V1.7	V1.0

PID_Temp V1 の CPU 処理時間およびメモリ要件

CPU 処理時間

バージョン 1.0 以降の PID_Temp テクノロジーオブジェクトの通常の CPU 処理時間(CPU タイプにより異なる)。

CPU	代表値 PID_Temp V1 の CPU 処理時間
CPU 1211C ≥ V4.1	580 µs
CPU 1215C ≥ V4.1	580 µs
CPU 1217C ≥ V4.1	580 µs
CPU 1505S ≥ V1.0	50 µs
CPU 1510SP-1 PN ≥ V1.7	130 µs
CPU 1511-1 PN ≥ V1.7	130 µs
CPU 1512SP-1 PN ≥ V1.7	130 µs
CPU 1516-3 PN/DP ≥ V1.7	75 µs
CPU 1518-4 PN/DP ≥ V1.7	6 µs

メモリ要件

バージョン V1.0 以降の PID_Temp テクノロジーオブジェクトのインスタンス DB のメモリ要件。

	PID_Temp V1 のインスタンス DB のメモリ要件
ロードメモリ要件	約 17000 バイト
合計ワークメモリ要件	1280 バイト
保持ワークメモリ要件	100 バイト

PID_Temp

この章には下記に関する情報が記載されています：

PID_Temp の説明

説明

PID_Temp 命令は、PID コントローラに、温度プロセス用の統合された調整機能を提供します。PID_Temp は、純粋な加熱または加熱/冷却の用途に使用できます。

次の動作モードが使用可能です。

- 無効
- プレチューニング
- 微調整
- 自動モード
- 手動モード
- エラーモニタリング付きの代替出力値

動作モードの詳細については、State パラメータを参照してください。

PID アルゴリズム

PID_Temp は、比例および微分動作のワインドアップ防止および重み付け付きの PIDT1 コントローラです。 PID アルゴリズムは、以下の等式に従って動作します(制御ゾーンおよびデッドバンド無効):

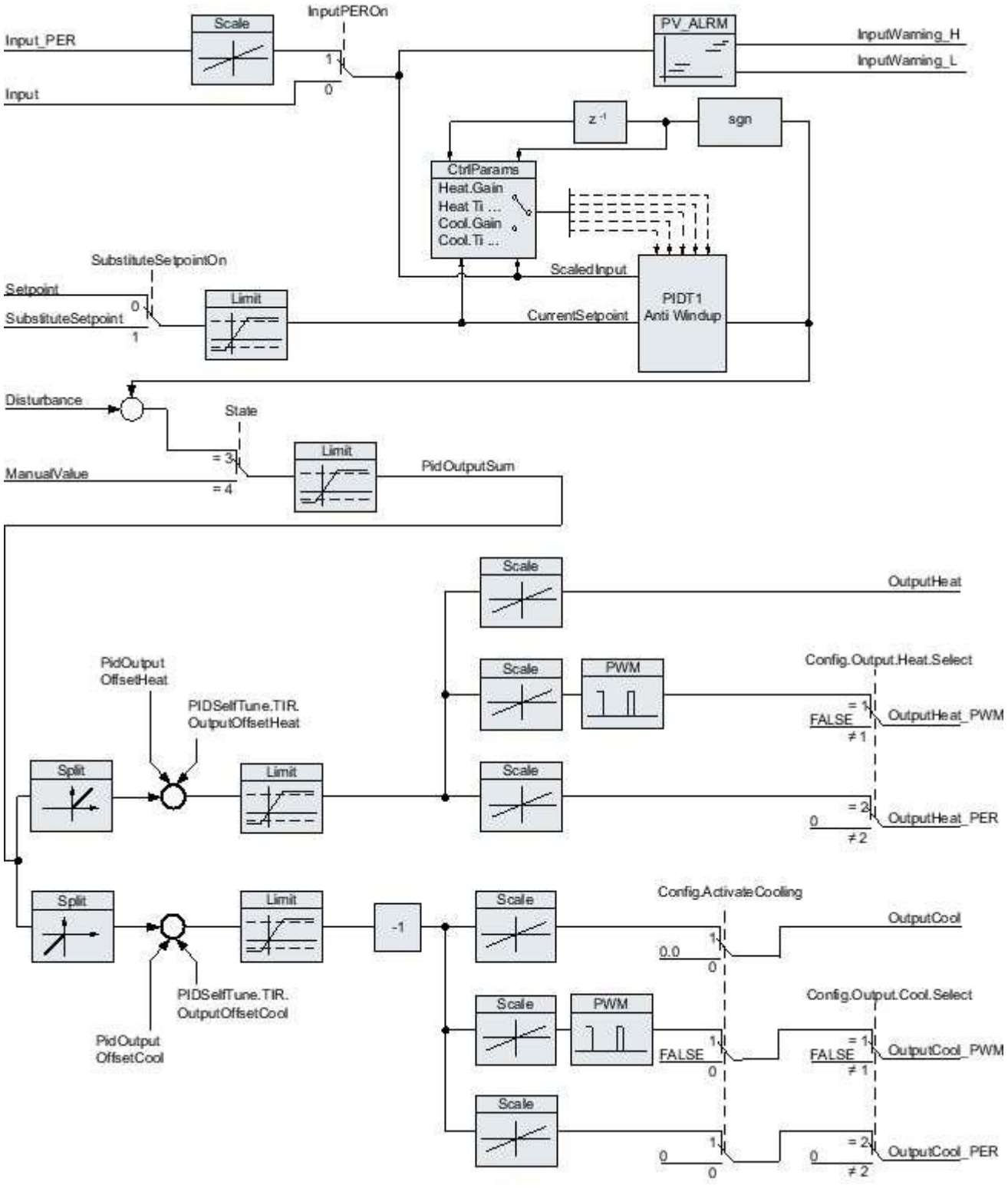
$$y = K_p \left[(b \cdot w - x) + \frac{1}{T_I \cdot s} (w - x) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1} (c \cdot w - x) \right]$$

次の表に、等式と以降の図で使用するアイコンの意味を示します。

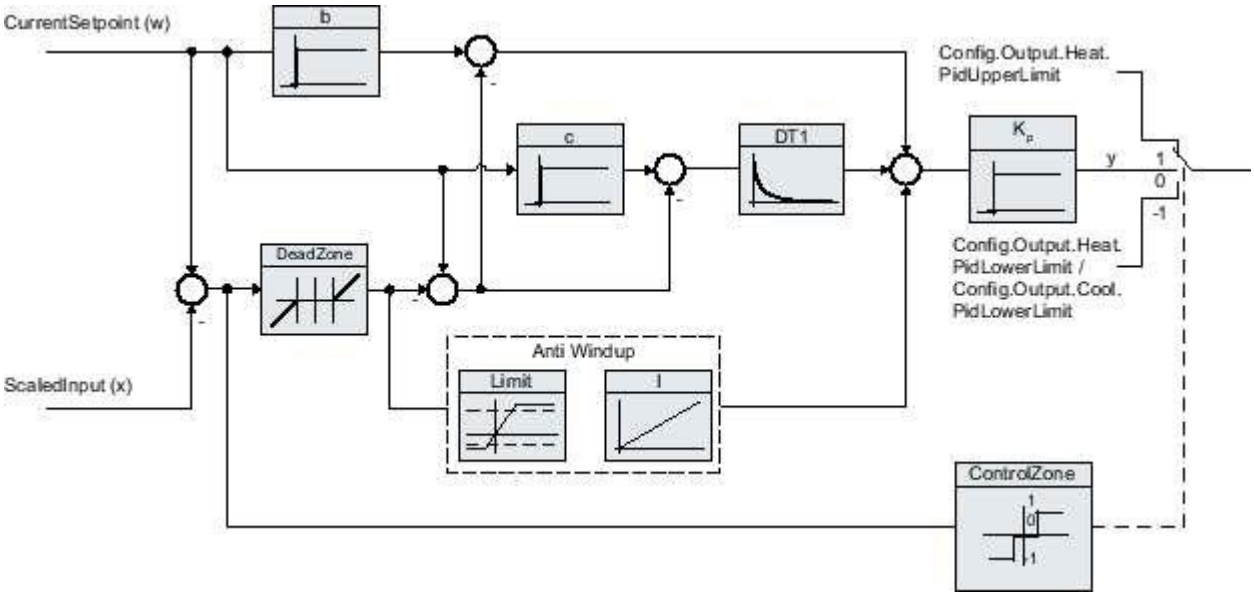
アイコン	説明	PID_Temp 命令の関連パラメータ
y	PID アルゴリズムの出力値	-
K_p	比例ゲイン	Retain.CtrlParams.Heat.Gain Retain.CtrlParams.Cool.Gain CoolFactor
s	ラプラス演算子	-
B	比例動作の重み付け	Retain.CtrlParams.Heat.PWeighting Retain.CtrlParams.Cool.PWeighting
w	セットポイント	CurrentSetpoint
x	プロセス値	ScaledInput
T_I	積分動作時間	Retain.CtrlParams.Heat.Ti Retain.CtrlParams.Cool.Ti
T_D	微分動作時間	Retain.CtrlParams.Heat.Td Retain.CtrlParams.Cool.Td

A	微分の遅延係数(微分の遅延 T1 = a × T_D)	Retain.CtrlParams.Heat.TdFiltRatio Retain.CtrlParams.Cool.TdFiltRatio
C	微分動作の重み付け	Retain.CtrlParams.Heat.DWeighting Retain.CtrlParams.Cool.DWeighting
DeadZone	デッドバンド幅	Retain.CtrlParams.Heat.DeadZone Retain.CtrlParams.Cool.DeadZone
ControlZone	コントロールゾーン幅	Retain.CtrlParams.Heat.ControlZone Retain.CtrlParams.Cool.ControlZone

PID_Temp のブロックダイアグラム



ワインドアップ防止付きの PIDT1 のブロックダイアグラム



呼び出し

PID_Temp は、サイクリック割り込み DB の一定のタイムスケールで呼び出されます。

マルチインスタンス DB として PID_Temp を呼び出すと、テクノロジーオブジェクトは作成されません。 パラメータ割り当てインターフェースおよびコミッショニングインターフェースは使用できません。 複数インスタンス DB で直接に PID_Temp のパラメータを割り当て、ウォッチテーブル経由でそれをコミッショニングする必要があります。

デバイスへのダウンロード

保持変数のプロセス値は、PID_Temp 全体をダウンロードしたときにのみ更新されます。

デバイスへのテクノロジーオブジェクトのダウンロード

スタートアップ

CPU のスタートアップ時に、PID_Temp が ModeIN/OUT パラメータに保存された動作モードで開始されます。 スタートアップ中に「無効」動作モードに切り替えるには、RunModeByStartup = FALSE を設定します。

エラーに対する応答

エラーが発生した場合の動作は、タグ SetSubstituteOutput および ActivateRecoverMode によって決まります。 ActivateRecoverMode = TRUE の場合、動作は発生したエラーによっても異なります。

SetSubstitu-teOutput	ActivateR-ecover-Mode	構成エディタ > 出力の基本設定 > PidOutputSum を以下に設定	応答
関連なし	FALSE	ゼロ(無効)	「無効」(State = 0)モードに切り替え PID アルゴリズムの出力値と加熱および冷却のすべての出力が 0 に設定されます。加熱および冷却のスケーリングは非アクティブです。

FALSE	TRUE	エラーが保留中の間、エラーの現在値	「エラーモニタ付きの代替出力値」モード (State = 5)に切り替え 現在の出力値が、エラーの保留中にアクチュエータに転送されます。
TRUE	TRUE	エラーが保留中の間、代替出力値	「エラーモニタ付きの代替出力値」モード (State = 5)に切り替え SubstituteOutput の値がエラーの保留中にアクチュエータに転送されます。

手動モードでは、PID_Temp は ManualValue を出力値として使用します(ManualValue が無効でない場合)。

- ManualValue が無効の場合、SubstituteOutput が使用されます。
- ManualValue および SubstituteOutput が無効の場合、Config.Output.Heat.PidLowerLimit が使用されます。

Error パラメータは、エラーが保留中かどうかを示します。 エラーが保留中でない場合、Error = FALSE となります。 ErrorBits パラメータは、どのエラーが発生したかを示します。ErrorBits は、Reset または ErrorAck の立ち上がりエッジでリセットされます。

PID_Temp の機能説明

プロセス値限界値のモニタリング

Config.InputUpperLimit および Config.InputLowerLimit タグで、プロセス値の上限値と下限値を指定します。プロセス値がこの限界値を超えている場合、エラーが発生します(ErrorBits = 0000001h)。

Config.InputUpperWarning および Config.InputLowerWarning タグで、プロセス値の警告上限値と警告下限値を指定します。プロセス値がこの警告限界値を超えている場合、警告が発生し(Warning = 0000040h)、InputWarning_H または InputWarning_L 出力パラメータが TRUE に変わります。

セットポイントの制限

Config.SetpointUpperLimit および Config.SetpointLowerLimit タグで、セットポイントの上限値と下限値を指定します。 PID_Temp は、セットポイントを自動的にプロセス値の限界値に制限します。 セットポイントをさらに小さな領域に制限できます。 PID_Temp は、この領域がプロセス値の限界値内にあるかどうかをチェックします。 セットポイントがこの限界値を超えると、上限値または下限値がセットポイントとして使用され、出力パラメータ SetpointLimit_H または SetpointLimit_L が TRUE に設定されます。

セットポイントは、すべての動作モードで、制限されます。

代替セットポイント

SubstituteSetpoint タグで代替セットポイントを指定し、SubstituteSetpointOn = TRUE で有効にできます。 このようにして、たとえば、ユーザープログラムを変更する必要なく、カスケード内のスレーブコントローラについて、セットポイントを一時的に直接指定できます。 セットポイントの限界値設定は、代替セットポイントにも適用されます。

加熱および冷却

既定の設定で、PID_Temp は加熱の出力のみを使用します(OutputHeat, OutputHeat_PWM, OutputHeat_PER)。 PID アルゴリズムの出力値(PidOutputSum)はスケーリングされ、加熱の出力で出力されます。 OutputHeat_PWM または OutputHeat_PER が計算される場合、Config.Output.Heat.Select で指定します。OutputHeat が常に計算されます。

Config.ActivateCooling = TRUE を使用して、冷却の出力を有効にすることもできます(OutputCool, OutputCool_PWM, OutputCool_PER)。 PID アルゴリズムの正の出力値(PidOutputSum)はスケーリングされ、加熱の出力で出力されます。 PID アルゴリズムの負の出力値はスケーリングされ、冷却の出力で出力されます。 OutputCool_PWM または OutputCool_PER が計算される場合、Config.Output.Cool.Select で指定します。OutputCool が常に計算されます。

冷却を有効にして PID 出力値を計算する方法は 2 つあります。

- 冷却係数(Config.AdvancedCooling = FALSE):

  冷却の出力値計算は、設定可能な冷却係数 Config.CoolFactor を考慮し、加熱用の PID パラメータを使用して行います。 この方法は、加熱および冷却アクチュエータの時間応答が類似し、ゲインが異なる場合に適切です。 この方法を選択すると、冷却のプレチューニングと微調整および冷却に設定された PID パラメータが使用できません。 加熱の調整のみを実行できます。

- PID パラメータの切り替え(Config.AdvancedCooling = TRUE):

  冷却の出力値計算は、個別の PID パラメータセットを使用して行います。 計算の出力値および制御偏差に基づいて、PID アルゴリズムが加熱用または冷却用 PID パラメータを使用するかどうかを決定します。 この方法は、加熱および冷却アクチュエータの時間応答が異なり、かつゲインが異なる場合に適切です。 冷却のプレチューニングおよび微調整は、この方法を選択した場合にのみ使用可能です。

出力限界値およびスケーリング

動作モードに基づき、PID 出力値(PidOutputSum)が PID アルゴリズムによって自動的に計算されるか、手動値(ManualValue)または設定済み代替出力値 (SubstituteOutput)によって定義されます。

PID 出力値は、設定によって制限されます。

- 冷却が無効な場合(Config.ActivateCooling = FALSE)、 Config.Output.Heat.PidUpperLimit が上限値であり、Config.Output.Heat.PidLowerLimit が下限値です。
- 冷却が有効な場合(Config.ActivateCooling = TRUE)、Config.Output.Heat.PidUpperLimit が上限値であり、Config.Output.Cool.PidLowerLimit が下限値です。

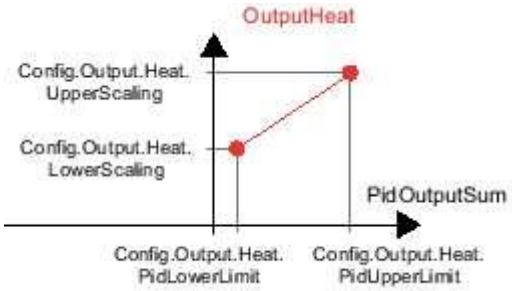
PID 出力値はスケーリングされ、加熱および冷却の出力で出力されます。 スケーリングは各出力について個別に定義することができ、構造体 Config.Output.Heat または Config.Output.Cool 内で、それぞれ 2 つの値ペアを使用して指定します。

出力	値ペア	パラメータ
OutputHeat	値ペア 1	PID 出力値の上限値(加熱) Config.Output.Heat.PidUpperLimit, スケールされた高出力値(加熱)Config.Output.Heat.UpperScaling
	値ペア 2	PID 出力値の下限値(加熱) Config.Output.Heat.PidLowerLimit, スケールされた低 PWM 出力値(加熱)Config.Output.Heat.LowerScaling
OutputHeat_PWM	値ペア 1	PID 出力値の上限値(加熱) Config.Output.Heat.PidUpperLimit, スケールされた高 PWM 出力値(加熱) Config.Output.Heat.PwmUpperScaling
	値ペア 2	PID 出力値の下限値(加熱) Config.Output.Heat.PidLowerLimit, スケールされた低 PWM 出力値(加熱) Config.Output.Heat.PwmLowerScaling
OutputHeat_PER	値ペア 1	PID 出力値の上限値(加熱) Config.Output.Heat.PidUpperLimit, スケールされた高アナログ出力値(加熱) Config.Output.Heat.PerUpperScaling
	値ペア 2	PID 出力値の下限値(加熱) Config.Output.Heat.PidLowerLimit, スケールされた低アナログ出力値(加熱) Config.Output.Heat.PerLowerScaling
OutputCool	値ペア 1	PID 出力値の下限値(冷却) Config.Output.Cool.PidLowerLimit,

		スケールされた高出力値(冷却) Config.Output.Cool.UpperScaling
	値ペア 2	PID 出力値の上限値(冷却) Config.Output.Cool.PidUpperLimit, スケールされた低出力値(冷却) Config.Output.Cool.LowerScaling
OutputCool_PWM	値ペア 1	PID 出力値の下限値(冷却) Config.Output.Cool.PidLowerLimit, スケールされた高 PWM 出力値(冷却) Config.Output.Cool.PwmUpperScaling
	値ペア 2	PID 出力値の上限値(冷却) Config.Output.Cool.PidUpperLimit, スケールされた低 PWM 出力値(冷却) Config.Output.Cool.PwmLowerScaling
OutputCool_PER	値ペア 1	PID 出力値の下限値(冷却) Config.Output.Cool.PidLowerLimit, スケールされた高アナログ出力値(冷却) Config.Output.Cool.PerUpperScaling
	値ペア 2	PID 出力値の上限値(冷却) Config.Output.Cool.PidUpperLimit, スケールされた低アナログ出力値(冷却) Config.Output.Cool.PerLowerScaling
冷却が有効である場合(Config.ActivateCooling = TRUE)、Config.Output.Heat.PidLowerLimit の値は 0.0 でなければなりません。 Config.Output.Cool.PidUpperLimit の値は常に 0.0 でなければなりません。		

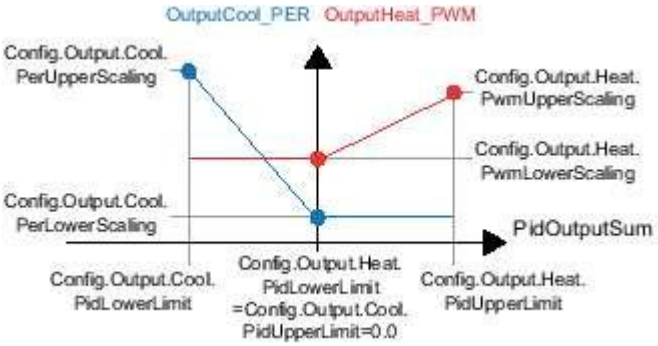
例:

出力 OutputHeat(冷却無効化、Config.Output.Heat.PidLowerLimit は 0.0 に等しくなくてよい)を使用した場合の出力スケーリング:



例:

出力 OutputHeat_PWM および OutputCool_PER(冷却有効化、Config.Output.Heat.PidLowerLimit は 0.0 でなければならない)を使用した場合の出力スケーリング:



「無効」動作モードを除いて、出力値は常にそのスケーリングされた上限出力値と下限出力値の間に存在します。たとえば、OutputHeat の場合、常に Config.Output.Heat.UpperScaling と Config.Output.Heat.LowerScaling の間に存在します。

この値を関連する出力で制限する場合は、これらのスケーリング値も調整する必要があります。

カスケード

PID_Temp はカスケード制御の使用をサポートしています(参照: **プログラムの作成**).

代替出力値

エラーが発生した場合、PID_Temp は SubstituteOutput タグで定義している代替出力値を出力できます。 代替出力値は、PID 出力限界値内であることが必要です。 代替出力値により発生する加熱および冷却の出力値は、設定済み出力スケーリングの結果です。

信号妥当性のモニタリング

以下のパラメータの値は、使用時に有効性をモニタされます。

- Setpoint
- SubstituteSetpoint
- Input
- Input_PER
- Disturbance
- ManualValue
- SubstituteOutput
- 構造体 Retain.CtrlParams.Heat および Retain.CtrlParams.Cool.内の PID パラメータ

PID_Temp サンプリング時間のモニタリング

理想的には、サンプリング時間はサイクリック割り込み OB のサイクルタイムと等値です。PID_Temp 命令は、2 つの呼び出し間の時間間隔を測定します。 これは、現在のサンプル時間です。動作モードの変更ごと、また、初期スタートアップ時に、最初の 10 個のサンプリング時間からサンプリング時間の平均値が求められます。現在のサンプリング時間とこの平均値の相違が大きすぎると、エラー(Error = 0000800h)がトリガされます。

以下の場合、調整時にエラーが発生します。

- 新しい平均値 >= 1.1 x 古い平均値
- 新しい平均値 <= 0.9 x 古い平均値

以下の場合、自動モードでエラーが発生します。

- 新しい平均値 >= 1.5 x 古い平均値
- 新しい平均値 <= 0.5 x 古い平均値

サンプリング時間のモニタを無効にすると(CycleTime.EnMonitoring = FALSE)、PID_Temp を OB1 で呼び出すこともできます。 この場合、サンプリング時間のずれによる制御品質の低下を受け入れる必要があります。

PID アルゴリズムのサンプリング時間

コントロールされたシステムは、出力値の変更に応答するために、一定量の時間を必要とします。 このため、すべてのサイクルごとに出力値を計算することはお奨めしません。 PID アルゴリズムのサンプリング時間は、出力の 2 つの計算の間の時間を表します。 これは調整中に計算され、サイクリック割り込み OB のサイクルタイムの偶数倍に四捨五入されます(サンプリング時間 PID_Temp)。 PID_Temp の他のすべてのファンクションは、呼び出しの都度実行されます。

冷却および PID パラメータの切り替えが有効である場合、PID_Temp は加熱および冷却の PID アルゴリズムに個別のサンプリング時間を使用します。 その他のすべての設定では、加熱用の PID アルゴリズムのサンプリング時間のみが使用されます。

OutputHeat_PWM または OutputCool_PWM を使用する場合、パルス幅変調の時間として PID アルゴリズムのサンプリング時間が使用されます。 出力信号の精度は、OB のサイクルタイムに対する PID アルゴリズムのサンプリング時間の比率によって決まります。サイクルタイムは、PID アルゴリズムのサンプリング時間の 10 分の 1 以下であることが必要です。

OutputHeat_PWM または OutputCool_PWM を使用するとき、PID アルゴリズムサンプリング時間、つまりパルス幅変調の時間が非常に長い場合、Config.Output.Heat.PwmPeriode または Config.Output.Cool.PwmPeriode パラメータで時間のずれを短く設定して、プロセス値の滑らかさを向上させることができます。

制御論理

PID_Temp は、加熱または加熱/冷却の用途に使用でき、常に標準の制御論理で作動します。

PID 出力値の増加は(PidOutputSum)、プロセス値の増加のために行います。 PID 出力値により発生する加熱および冷却の出力値は、設定済み出力スケーリングの結果です。

制御ロジックの反転または負の比例ゲインはサポートされていません。

増加がプロセス値を減少させる目的のみのアプリケーションで出力値が必要な場合(たとえば、排出制御)、制御ロジックの反転によって PID_Compact を使用できます。

PID_Temp の入力パラメータ

パラメータ	データタイプ	既定	説明
Setpoint	REAL	0.0	自動モードでの PID コントローラのセットポイント 値の有効な範囲: Config.SetpointUpperLimit ≥ Setpoint ≥ Config.SetpointLowerLimit Config.InputUpperLimit ≥ Setpoint ≥ Config.InputLowerLimit
Input	REAL	0.0	ユーザープログラムのタグが、プロセス値のソースとして使用されます。 Input パラメータを使用する場合は、Config.InputPerOn = FALSE を設定する必要があります。
Input_PER	INT	0	アナログ入力が、プロセス値のソースとして使用されます。 Input_PER パラメータを使用する場合は、Config.InputPerOn = TRUE を設定する必要があります。
Disturbance	REAL	0.0	外乱変数または前制御値
ManualEnable	BOOL	FALSE	• A FALSE -> TRUE エッジでは「手動モード」が有効になり、State = 4、Mode はそのまま変わりません。 ManualEnable = TRUE である限り、ModeActivate の立ち上がりエッジで動作モードを変更したり、コミッショニングダイアログを使用したりすることはできません。 • TRUE -> FALSE エッジにより、Mode で指定された動作モードが有効になります。 Mode および ModeActivate のみを使用して動作モードを変更することをお勧めします。
ManualValue	REAL	0.0	手動値 この値は、手動モードで、PID 出力値 (PidOutputSum)として使用されます。 この手動値から発生する加熱および冷却の出力値は、設定済み出力スケーリング(構造体 Config.Output.Heat および Config.Output.Cool)の結果です。 冷却出力が有効になったコントローラの場合(Config.ActivateCooling = TRUE)、以下を定義します。 • 加熱用出力で出力する正の手動値 • 冷却用出力で出力する負の手動値 許容範囲は設定によって決まります。 • 無効になった冷却出力(Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Heat.PidLowerLimit • 有効になった冷却出力(Config.ActivateCooling = TRUE):

			Config.Output.Heat.PidUpperLimit ≥ ManualValue ≥ Config.Output.Cool.PidLowerLimit
ErrorAck	BOOL	FALSE	• FALSE -> TRUE エッジ ErrorBits および Warning がリセットされます。
Reset	BOOL	FALSE	コントローラを再起動します。 • FALSE -> TRUE エッジ ○「無効」モードに切り替え ○ ErrorBits および Warning がリセットされます。 ○ 積分動作がクリアされます。 (PID パラメータは保持されます) • Reset = TRUE である限り、 ○ PID_Temp は「無効」モード (State = 0)のままになります。 ○ Mode および ModeActivate または ManualEnable を使用して動作モードを変更できません。 ○ コミッショニングダイアログは使用できません。 • TRUE -> FALSE エッジ PID_Temp が Mode パラメータに保存された動作モードに切り替わります。
ModeActivate	BOOL	FALSE	• FALSE -> TRUE エッジ PID_Temp が Mode 入力に保存されている動作モードに切り替わります。

PID_Temp の出力パラメータ

パラメータ	データタイプ	既定	説明
ScaledInput	REAL	0.0	スケーリングされたプロセス値
OutputHeat	REAL	0.0	REAL フォーマットの出力値(加熱) PID 出力値(PidOutputSum)は 2 つの値ペア Config.Output.Heat.PidUpperLimit, Config.Output.Heat.UpperScaling および Config.Output.Heat.PidLowerLimit, Config.Output.Heat.LowerScaling によってスケーリングされ、OutputHeat で REAL フォーマットで出力されます。 OutputHeat は常に計算されます。
OutputCool	REAL	0.0	REAL フォーマットの出力値(冷却) PID 出力値(PidOutputSum)は 2 つの値ペア Config.Output.Cool.PidUpperLimit, Config.Output.Cool.LowerScaling および Config.Output.Cool.PidLowerLimit, Config.Output.Cool.UpperScaling によってスケーリングされ、OutputCool で REAL フォーマットで出力されます。 OutputCool は、冷却出力が有効な場合のみ計算されます(Config.ActivateCooling = TRUE)。
OutputHeat_PER	INT	0	アナログ出力値(加熱) PID 出力値(PidOutputSum)は 2 つの値ペア Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PerUpperScaling および Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PerLowerScaling によってスケーリングされ、OutputHeat_PER でアナログ値として出力されます。 OutputHeat_PER は、Config.Output.Heat.Select = 2 の場合にのみ計算されます。
OutputCool_PER	INT	0	アナログ出力値(冷却) PID 出力値(PidOutputSum)は 2 つの値ペア Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PerLowerScaling および Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PerUpperScaling によってスケーリングされ、OutputCool_PER でアナログ値として出力されます。 OutputCool_PER は、冷却出力が有効で(Config.ActivateCooling = TRUE)かつ Config.Output.Cool.Select = 2 の場合にのみ計算されます。
OutputHeat_PWM	BOOL	FALSE	パルス幅調整された出力値(加熱) PID 出力値(PidOutputSum)は 2 つの値ペア Config.Output.Heat.PidUpperLimit, Config.Output.Heat.PwmUpperScaling および Config.Output.Heat.PidLowerLimit, Config.Output.Heat.PwmLowerScaling によってスケーリングされ、OutputHeat_PWM でパルス幅調整された出力値(変数スイッチオンおよびスイッチオフ時間)として出力されます。

			OutputHeat_PWM は、Config.Output.Heat.Select = 1 の場合にのみ計算されます。
Output-Cool_PWM	BOOL	FALSE	パルス幅調整された出力値(冷却) PID 出力値(PidOutputSum)は 2 つの値ペア Config.Output.Cool.PidUpperLimit, Config.Output.Cool.PwmLowerScaling および Config.Output.Cool.PidLowerLimit, Config.Output.Cool.PwmUpperScaling によってスケーリングされ、OutputCool_PWM でパルス幅調整された値(変数スイッチオンおよびスイッチオフ時間)として出力されます。 OutputCool_PWM は、冷却出力が有効で(Config.ActivateCooling = TRUE)かつ Config.Output.Cool.Select = 1 の場合にのみ計算されます。
SetpointLimit_H	BOOL	FALSE	SetpointLimit_H = TRUE の場合、セットポイントの上限絶対値(Setpoint ≥ Config.SetpointUpperLimit)または Setpoint ≥ Config.InputUpperLimit に達します。 セットポイントの上限値は Config.SetpointUpperLimit および Config.InputUpperLimit の最小値です。
SetpointLimit_L	BOOL	FALSE	SetpointLimit_L = TRUE の場合、セットポイントの下限絶対値(Setpoint ≤ Config.SetpointLowerLimit)または Setpoint ≤ Config.InputLowerLimit に達します。 セットポイントの下限値は Config.SetpointLowerLimit および Config.InputLowerLimit の最大値です。
InputWarning_H	BOOL	FALSE	InputWarning_H = TRUE の場合、プロセス値が警告上限値に到達済みか、警告上限値を超えています(ScaledInput ≥ Config.InputUpperWarning)。
InputWarning_L	BOOL	FALSE	InputWarning_L = TRUE の場合、プロセス値が警告下限値に到達済みか、警告下限値未満になっています(ScaledInput ≤ Config.InputLowerWarning)。
State	INT	0	PID_Temp の状態およびモードパラメータは、PID コントローラの現在の動作モードを示します。 入力パラメータ Mode および ModeActivate の立ち上がりエッジを使用して、動作モードを変更することができます。 プレチューニングおよび微調整のため、Heat.EnableTuning および Cool.EnableTuning を使用して、加熱または冷却で調整を行うかどうかを指定します。 • State = 0: 無効 • State = 1: プレチューニング • State = 2: 微調整 • State = 3: 自動モード • State = 4: 手動モード • State = 5: エラーモニタリング付きの代替出力値
Error	BOOL	FALSE	Error = TRUE の場合、少なくとも 1 つのエラーがこのサイクルで保留中です。
ErrorBits	DWORD	DW#16#0	PID_Temp ErrorBits パラメータは保留中のエラーメッセージを示します。 ErrorBits は保持され、Reset または ErrorAck の立ち上がりエッジでリセットされます。

PID_Temp の入力/出力パラメータ

パラメータ	データタイプ	既定	説明
Mode	INT	4	Mode で、PID_Temp の切り替え先である動作モードを指定します。オプションを以下に示します。 • Mode = 0: 無効 • Mode = 1: プレチューニング • Mode = 2: 微調整 • Mode = 3: 自動モード • Mode = 4: 手動モード 動作モードは以下によって有効になります。 • ModeActivate の立ち上がりエッジ • Reset の立ち下がりエッジ • ManualEnable の立ち下がりエッジ • RunModeByStartup = TRUE の場合は CPU のコールドリスタート プレチューニングおよび微調整のため、Heat.EnableTuning および Cool.EnableTuning を使用して、加熱または冷却で調整を行うかどうかを指定します。 Mode は保持されます。 動作モードの詳細については、状態およびモードパラメータを参照してください。
Master	DWORD	DW#16#0	カスケード制御のインターフェース この PID_Temp インスタンスをカスケードでスレーブコントローラとして使用する場合(Config.Cascade.IsSlave = TRUE)、命令の呼び出しでマスタコントローラのスレーブパラメータでマスタパラメータを割り当てます。 例: スレーブコントローラ「PID_Temp_2」の SCL 内のマスタコントローラ「PID_Temp_1」による呼び出し: --- `"PID_Temp_2"(Master := "PID_Temp_1".Slave, Setpoint := "PID_Temp_1".OutputHeat);` --- このインターフェースを使用して、動作モード、限界値および代替セットポイントに関するスレーブコントローラ情報をマスタコントローラと交換します。 マスタコントローラの呼び出しは、同じサイクリック割り込み OB でのスレーブコントローラの呼び出しよりも前に発生する必要があることに注意してください。 割り当て:

			• ビット 0 ~ 15: 割り当てなし • ビット 16 ~ 23: – リミットカウンタ: 出力値がこのカウンタの制限インクリメントであるスレーブコントローラ。 設定済みスレーブの数(Config.Cascade.CountSlaves)およびワインドアップ防止モードに応じて(Config.Cascade.AntiWindUpMode)、マスタコントローラは適切に応答します。 • ビット 24 – スレーブコントローラの自動モード: すべてのスレーブコントローラが自動モードの場合、TRUE • ビット 25 – スレーブコントローラの代替セットポイント: スレーブコントローラが代替セットポイントを有効にした場合、TRUE (代替値 SetpointOn = TRUE)
Slave	DWORD	DW#16#0	カスケード制御のインターフェース このインターフェースを使用して、動作モード、限界値および代替セットポイントに関するスレーブコントローラ情報をマスタコントローラと交換します。 マスタパラメータの説明を参照。

PID_Temp 静的タグ

リストにないタグを変更してはいけません。 これらは、システム内部の目的のためだけに使用されます。

タグ	データタイプ	既定	説明
IntegralResetMode	INT	1	IntegralResetMode タグが積分動作 PIDCtrl.IOutputOld の既定設定を決定します。 "動作モード「無効」から「自動モード」に変更するとき、 この設定は 1 サイクルの間だけ有効です。 • IntegralResetMode = 0: まるめ 値は、切り替えがバンプレスとなるように割り当てられます。 • IntegralResetMode = 1: 削除 値は消去されます。 制御偏差によって出力値がジャンプ変動しません。 • IntegralResetMode = 2: 保持 値は変更されません。 ユーザープログラムを使用して新しい値を定義できます。 • IntegralResetMode = 3: 事前に割り当て 値は、最後のサイクルで PidOutputSum = OverwriteInitialOutputValue であるかのように自動的に事前割り当てされます。 この設定は、たとえば、オーバーライドコントローラなどに有用です。
OverwriteInitialOutputValue	REAL	0.0	IntegralResetMode = 3 である場合、PIDCtrl.IOutputOld の値は最後のサイクルで「PidOutputSum」=「OverwriteInitialOutputValue」であるかのように事前割り当てされます。
RunModeByStartup	BOOL	TRUE	CPU のリスタート後に、Mode パラメータの動作モードを有効にします。 • RunModeByStartup = TRUE の場合、PID_Temp は CPU のリスタート後に、Mode パラメータに保存された動作モードで開始されます。 • RunModeByStartup = FALSE の場合、PID_Temp は CPU のリスタート後も「無効」モードのままです。
LoadBackUp	BOOL	FALSE	LoadBackUp = TRUE の場合、PID パラメータの最後のセットは CtrlParamsBackUp 構造体から再ロードされます。 このセットは、最後の調整前に保存されたものです。LoadBackUp は自動的に FALSE に再セットされます。 受信はバンプレスです。
SetSubstituteOutput	BOOL	TRUE	エラーが保留中の出力値の選択(State = 5): • SetSubstituteOutput = TRUE および ActivateRecoverMode = TRUE の場合、エラーが保留になっている間は設定された代替出力値 SubstituteOutput が出力されます。

			• SetSubstituteOutput = FALSE および ActivateRecover-Mode = TRUE の場合、エラーが保留になっている間、アクチュエータは現在の PID 出力値のままです。 • ActivateRecoverMode = FALSE の場合、SetSubstituteOutput は無効です。 • SubstituteOutput が無効の場合(ErrorBits = 0020000h)、代替出力値は出力できません。 この場合、加熱の PID 出力値の下限値(Config.Output.Heat.PidLowerLimit)が PID 出力値として使用されます。
Physica-lUnit	INT	0	プロセス値およびセットポイントの測定単位(たとえば、℃または ºF) このパラメータはエディタの表示に使用され、制御アルゴリズムには影響を与えません。
Physical-Quantity	INT	0	プロセス値およびセットポイントの物理量(たとえば、温度) このパラメータはエディタの表示に使用され、制御アルゴリズムには影響を与えません。
Activa-teReco-verMode	BOOL	TRUE	ActivateRecoverMode タグは、エラーに対する応答を決定します。
Warning	DWORD	0	Warning タグは、Reset = TRUE または ErrorAck =TRUE からの警告を示します。 警告は保持されます。
Progress	REAL	0.0	現在の調整段階の進捗状況(パーセンテージ(0.0 ~ 100.0)で表示)
Current-Setpoint	REAL	0.0	CurrentSetpoint は常に、現在有効なセットポイントを示します。 この値は調整中は固定されます。
Cancel-TuningLe-vel	REAL	10.0	調整中のセットポイントの許容変動。 調整は以下の状態になるまでキャンセルされません。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel
Substitu-teOutput	REAL	0.0	代替出力値は、以下の条件が満たされている限り PID 出力値として使用されます。 • ActivateRecoverMode が有効な自動モードで 1 つまたは複数のエラーが保留中である • SetSubstituteOutput = TRUE • ActivateRecoverMode = TRUE この代替出力値から発生する加熱および冷却の出力値は、設定済み出力スケーリング(構造体 Config.Output.Heat および Con-fig.Output.Cool)の結果です。 冷却出力が有効になったコントローラの場合(Config.Activate-Cooling = TRUE)、以下を定義します。 • 加熱用出力で出力する正の代替出力値 • 冷却用出力で出力する負の代替出力値 許容範囲は設定によって決まります。 • 無効になった冷却出力(Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Con-fig.Output.Heat.PidLowerLimit

			• 有効になった冷却出力(Config.ActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ SubstituteOutput ≥ Config.Output.Cool.PidLowerLimit
PidOutputSum	REAL	0.0	PID 出力値 PidOutputSum は PID アルゴリズムの出力値を示します。 動作モードに応じて、自動的に計算されるか、手動値によって定義されるか、または設定済み代替出力値のいずれかになります。 この PID 出力値から発生する加熱および冷却の出力値は、設定済み出力スケーリング(構造体 Config.Output.Heat および Config.Output.Cool)の結果です。 PidOutputSum は設定で定義されたとおりに制限されます。 • 無効になった冷却出力(Config.ActivateCooling = FALSE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Heat.PidLowerLimit • 有効になった冷却出力(ConfigActivateCooling = TRUE): Config.Output.Heat.PidUpperLimit ≥ PidOutputSum ≥ Config.Output.Cool.PidLowerLimit
PidOutputOffsetHeat	REAL	0.0	加熱の PID 出力値のオフセット PidOutputOffsetHeat が、加熱分岐の PidOutputSum から発生する値に追加されます。 加熱の出力で正のオフセットを受信するには、PidOutputOffsetHeat に正の値を入力します。 加熱の出力で発生する値は、設定済み出力スケーリングの結果です(Config.Output.Heat 構造体)。 このオフセットは、固定最小値を必要とするアクチュエータに使用できます(最小速度のあるファンなど)。
PidOutputOffsetCool	REAL	0.0	冷却の PID 出力値のオフセット PidOutputOffsetCool が、冷却分岐の PidOutputSum から発生する値に追加されます。 冷却の出力で正のオフセットを受信するには、PidOutputOffsetCool に負の値を入力します。 冷却の出力で発生する値は、設定済み出力スケーリングの結果です(Config.Output.Cool 構造体)。 このオフセットは、固定最小値を必要とするアクチュエータに使用できます(最小速度のあるファンなど)。
SubstituteSetpointOn	BOOL	FALSE	代替セットポイントをコントローラセットポイントとして有効にします。 • FALSE = Setpoint パラメータが使用されます。 • TRUE = SubstituteSetpoint パラメータがセットポイントとして使用されます。 SubstituteSetpointOn を使用して、ユーザプログラムに切り替えずにカスケードで直接スレーブコントローラのセットポイントを指定できます。
SubstituteSetpoint	REAL	0.0	代替セットポイント SubstituteSetpointOn = TRUE の場合、SubstituteSetpoint パラメータがセットポイントとして使用されます。 値の有効な範囲:

			Config.SetpointUpperLimit ≥ SubstituteSetpoint ≥ Config.SetpointLowerLimit, Config.InputUpperLimit ≥ SubstituteSetpoint ≥ Config.InputLowerLimit
DisableCooling	BOOL	FALSE	DisableCooling = TRUE は、 自動モードで、PidOutputSum を下限値として 0.0 に設定することによって、加熱/冷却 コントローラの冷却分岐を無効にします(Config.ActivateCooling = TRUE)。 PidOutputOffsetCool および冷却出力の出力スケーリングはアクティブなままになります。 DisableCooling を複数ゾーンアプリケーションの調整に使用して、すべてのコントローラがまだ調整を完了していない限り、冷却分岐を一時的に無効にすることができます。 このパラメータはユーザーによって手動で設定/リセットされ、PID_Temp 命令によって自動的にリセットされません。
AllSlaveAutomaticState	BOOL	FALSE	この PID_Temp インスタンスがカスケードでマスタコントローラとして使用される場合(Config.Cascade.IsMaster = TRUE)、AllSlaveAutomaticState = TRUE はすべてのスレーブコントローラが自動モードであることを示します。 マスタコントローラの調整、手動モードまたは自動モードは、すべてのスレーブコントローラが自動モードである場合にのみ正確に実行できます。 AllSlaveAutomaticState は、マスタコントローラとスレーブコントローラをマスタパラメータとスレーブパラメータを使用して相互接続した場合にのみ判別されます。 詳細については、マスタパラメータを参照してください。
NoSlaveSubstituteSetpoint	BOOL	FALSE	この PID_Temp インスタンスがカスケードでマスタコントローラとして使用される場合(Config.Cascade.IsMaster = TRUE)、NoSlaveSubstituteSetpoint = TRUE はスレーブコントローラが代替セットポイントで有効にされていないことを示します。 マスタコントローラの調整、手動モードまたは自動モードは、スレーブコントローラが代替セットポイントで有効にされていない場合にのみ正確に実行できます。 NoSlaveSubstituteSetpoint は、マスタコントローラとスレーブコントローラをマスタパラメータとスレーブパラメータを使用して相互接続した場合にのみ判別されます。 詳細については、マスタパラメータを参照してください。
Heat.EnableTuning	BOOL	TRUE	加熱の調整の有効化 Heat.EnableTuning を以下の調整について設定する必要があります(同時または前に Mode および ModeActivate で開始する): • 加熱のプレチューニング • 加熱および冷却のプレチューニング • 加熱の微調整 このパラメータは、PID_Temp 命令によって自動的にリセットされません。
Cool.EnableTuning	BOOL	FALSE	冷却の調整の有効化 Cool.EnableTuning を以下の調整について設定する必要があります(同時または前に Mode および ModeActivate で開始する):

			• プレチューニング冷却 • 加熱および冷却のプレチューニング • 冷却の微調整 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(「Config.ActivateCooling」= TRUE かつ「Config.Advanced-Cooling」 = TRUE)。 このパラメータは、PID_Temp 命令によって自動的にリセットされません。
Config.In-putPerOn	BOOL	TRUE	InputPerOn = TRUE の場合、プロセス値を検出するために In-put_PER パラメータが使用されます。 InputPerOn = FALSE の場合、Input パラメータが使用されます。
Config.In-putUpper-Limit	REAL	120.0	プロセス値の上限値 この限界値が守られていることを確認するために、Input と In-put_PER をモニタします。 限界値を超えた場合、エラーが出力され、応答が ActivateRecoverMode によって決定されます。 I/O 入力で、プロセス値が公称範囲よりも最大 18%大きくなる場合があります(範囲オーバー)。 つまり、上限値およびプロセス値のスケーリングのプリセットで I/O 入力を使用する場合、限界値を超えることはできません。 プレチューニングの開始時、プロセス値の上限値と下限値の差がチェックされ、セットポイントとプロセス値間の距離が、必要な要件に一致するかどうかが判定されます。 InputUpperLimit > InputLowerLimit
Config.In-putLower-Limit	REAL	0.0	プロセス値の下限値 この限界値が守られていることを確認するために、Input と In-put_PER をモニタします。 限界値を下回った場合、エラーが出力され、応答が ActivateRecoverMode によって決定されます。 InputLowerLimit < InputUpperLimit
Config.In-putUpper-Warning	REAL	3.402822e+38	プロセス値の警告上限値 この限界値が守られていることを確認するために、Input と In-put_PER をモニタします。 上限値を上回った場合は、Warning パラメータに警告が出力されます。 • プロセス値制限範囲外の InputUpperWarning を設定すると、設定されたプロセス値の上限絶対値が警告上限値として使用されます。 • プロセス値制限範囲内の InputUpperWarning を設定すると、この値が警告上限値として使用されます。 InputUpperWarning > InputLowerWarning
Config.In-putLower-Warning	REAL	-3.402822e+38	プロセス値の警告下限値 この限界値が守られていることを確認するために、Input と In-put_PER をモニタします。 下限値を下回った場合は、Warning パラメータに警告が出力されます。 • プロセス値制限範囲外の InputLowerWarning を設定すると、設定されたプロセス値の下限絶対値が警告下限値として使用されます。 • プロセス値制限範囲内の InputLowerWarning を設定すると、この値が警告下限値として使用されます。

			InputLowerWarning < InputUpperWarning
Config.SetpointUpperLimit	REAL	3.402822e+38	セットポイントの上限値 この限界値が守られていることを確認するために、Setpoint と SubstituteSetpoint をモニタします。 上限値を上回った場合は、Warning パラメータに警告が出力されます。 • プロセス値制限範囲外の SetpointUpperLimit を設定すると、設定されたプロセス値の上限絶対値がセットポイントの上限値として使用されます。 • プロセス値制限範囲内の SetpointUpperLimit を設定すると、この値がセットポイントの上限値として使用されます。 SetpointUpperLimit > SetpointLowerLimit
Config.SetpointLowerLimit	REAL	-3.402822e+38	セットポイントの下限値 この限界値が守られていることを確認するために、Setpoint と SubstituteSetpoint をモニタします。 下限値を下回った場合は、Warning パラメータに警告が出力されます。 • プロセス値制限範囲外の SetpointLowerLimit を設定すると、設定されたプロセス値の下限絶対値がセットポイントの下限値として使用されます。 • プロセス値制限範囲内の SetpointLowerLimit を設定すると、この値がセットポイントの下限値として使用されます。 SetpointLowerLimit < SetpointUpperLimit
Config.ActivateCooling	BOOL	FALSE	冷却出力の有効化 • Config.ActivateCooling = FALSE 加熱の出力のみが使用されます。 • Config.ActivateCooling = TRUE 加熱および冷却の出力が使用されます。 冷却出力を使用している場合、コントローラをマスタコントローラとして設定してはなりません(Config.Cascade.IsMaster は FALSE である必要があります)。
Config.AdvancedCooling	BOOL	TRUE	加熱/冷却の方法 • 冷却係数(Config.AdvancedCooling = FALSE) 冷却の出力値計算は、設定可能な冷却計数 Config.CoolFactor を考慮し、加熱用の PID パラメータ(Retain.CtrlParams.Heat 構造体)を使用して行います。 この方法は、加熱および冷却アクチュエータの時間応答が類似し、ゲインが異なる場合に適切です。 冷却のプレチューニングおよび微調整は、この方法を選択した場合には使用できません。 加熱の調整のみを実行できます。 • PID パラメータの切り替え(Config.AdvancedCooling = TRUE) 冷却の出力値計算は、個別の PID パラメータセットを使用して行います(Retain.CtrlParams.Cool 構造体)。 この方法は、加熱および冷却アクチュエータの時間応答が異なり、かつゲインが異なる場合に適切です。

			冷却のプレチューニングおよび微調整は、この方法を選択した場合にのみ使用可能です(Mode = 1 または 2、Cool.Enable-Tuning = TRUE)。 Config.AdvancedCooling は、冷却出力が有効な場合のみ計算されます(Config.ActivateCooling = TRUE)。
Con-fig.Cool-Factor	REAL	1.0	冷却係数 Config.AdvancedCooling = FALSE の場合、Config.CoolFactor が冷却の出力値の計算で係数として考慮されます。 加熱および冷却アクチュエータの異なるゲインをこのようにして考慮できます。 Config.CoolFactor は自動的に設定されないか、調整中に調整されます。 比率「加熱アクチュエータゲイン/冷却アクチュエータゲイン」を使用して、Config.CoolFactor を手動で正しく設定する必要があります。 例: Config.CoolFactor = 2.0 は、加熱アクチュエータのゲインが、冷却アクチュエータのゲインの 2 倍であることを意味します。 Config.CoolFactor は冷却出力が有効で(Config.ActivateCooling = TRUE)、かつ冷却係数が加熱/冷却の方法として選択されている(Config.AdvancedCooling = FALSE)場合にのみ有効です。 Config.CoolFactor > 0.0
Config.In-putScal-ing.Upper-PointIn	REAL	27648.0	Input_PER (High)のスケーリング Input_PER は 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてスケーリングされます。 Input_PER がプロセス値検出に使用される場合にのみ有効です(Config.InputPerOn = TRUE)。 UpperPointIn > LowerPointIn
Config.In-putScal-ing.Lower-PointIn	REAL	0.0	Input_PER (Low)のスケーリング Input_PER は 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてスケーリングされます。 Input_PER がプロセス値検出に使用される場合にのみ有効です(Config.InputPerOn = TRUE)。 LowerPointIn < UpperPointIn
Config.In-putScal-ing.Upper-PointOut	REAL	100.0	スケーリングされた High プロセス値 Input_PER は 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてスケーリングされます。 Input_PER がプロセス値検出に使用される場合にのみ有効です(Config.InputPerOn = TRUE)。 UpperPointOut > LowerPointOut
Config.In-putScal-ing.Lower-PointOut	REAL	0.0	スケーリングされた Low プロセス値 Input_PER は 2 つの値ペア UpperPointOut、UpperPointIn および LowerPointOut、LowerPointIn に基づいてスケーリングされます。

			Input_PER がプロセス値検出に使用される場合にのみ有効です (Config.InputPerOn = TRUE)。 LowerPointOut < UpperPointOut
Config.Output.Heat.Select	INT	1	加熱の出力値の選択 Config.Output.Heat.Select は加熱に使用する出力を指定します。 • Heat.Select = 0 - OutputHeat が使用されます。 • Heat.Select = 1 - OutputHeat および OutputHeat_PWM が使用されます。 • Heat.Select = 2 -OutputHeat および OutputHeat_PER が使用されます。 使用されない出力は計算されず、既定値のままになります。
Config.Output.Heat.PwmPeriode	REAL	0.0	加熱のパルス幅変調(PWM)の時間(OutputHeat_PWM 出力)(秒単位): • Heat.PwmPeriode = 0.0 加熱の PID アルゴリズムのサンプリング時間(Retain.CtrlParams.Heat.Cycle)が PWM の時間として使用されます。 • Heat.PwmPeriode > 0.0 値は PID_Temp サンプリング時間(CycleTime.Value)の整数倍に四捨五入され、PWM の時間として使用されます。 この設定を使用して、PID アルゴリズムの長いサンプリング時間によってプロセス値のスムージングを向上させることができます。 値は、以下の条件を満たす必要があります。 ○ Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle, ○ Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime ○ Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime
Config.Output.Heat.PidUpperLimit	REAL	100.0	加熱の PID 出力値の上限値 PID 出力値 (PidOutputSum)は上限値に制限されます。 Heat.PidUpperLimit は、加熱の出力に対して、PID 出力値(PidOutputSum)のスケーリングのため、以下のパラメータと一緒に値ペアを形成します。 • Heat.UpperScaling FOR OutputHeat • Heat.PwmUpperScaling FOR OutputHeat_PWM • Heat.PerUpperScaling FOR OutputHeat_PER この値を関連する出力で制限する場合は、これらのスケーリング値も調整する必要があります。 Heat.PidUpperLimit > Heat.PidLowerLimit
Config.Output.Heat.PidLowerLimit	REAL	0.0	加熱の PID 出力値の下限値 冷却出力が**無効**なコントローラの場合(Config.ActivateCooling = FALSE)、PID 出力値(PidOutputSum)はこの下限値に制限されます。

			冷却出力が**有効**なコントローラの場合(Config.ActivateCooling = TRUE)、値は 0.0 である必要があります。 Heat.PidLowerLimit は、加熱の出力に対して、PID 出力値(PidOutputSum)のスケーリングのため、以下のパラメータと一緒に値ペアを形成します。 • Heat.LowerScaling FOR OutputHeat • Heat.PwmLowerScaling FOR OutputHeat_PWM • Heat.PerLowerScaling FOR OutputHeat_PER この値を関連する出力で制限する場合は、これらのスケーリング値も調整する必要があります。 許容範囲は設定によって決まります。 • 無効になった冷却出力(Config.ActivateCooling = FALSE): Heat.PidLowerLimit < Heat.PidUpperLimit • 有効になった冷却出力(Config.ActivateCooling = TRUE): Heat.PidLowerLimit = 0.0
Config.Output.Heat.UpperScaling	REAL	100.0	加熱のスケールされた高出力値 Heat.UpperScaling および Heat.PidUpperLimit は加熱の出力値(OutputHeat)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputHeat 値は、常に Heat.UpperScaling と Heat.LowerScaling の間に存在します。 Heat.UpperScaling ≠ Heat.LowerScaling
Config.Output.Heat.LowerScaling	REAL	0.0	加熱のスケールされた低 PWM 出力値 Heat.LowerScaling および Heat.PidLowerLimit は加熱の出力値(OutputHeat)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputHeat 値は、常に Heat.UpperScaling と Heat.LowerScaling の間に存在します。 Heat.UpperScaling ≠ Heat.LowerScaling
Config.Output.Heat.PwmUpperScaling	REAL	100.0	加熱のスケールされた高 PWM 出力値 Heat.PwmUpperScaling および Heat.PidUpperLimit は加熱のパルス幅調整された出力値(OutputHeat_PWM)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputHeat_PWM 値は、常に Heat.PwmUpperScaling と Heat.PWMLowerScaling の間に存在します。 Heat.PwmUpperScaling は、OutputHeat_PWM が加熱の出力として選択された場合にのみ有効です(Heat.Select = 1)。 100.0 ≥ Heat.PwmUpperScaling ≥ 0.0 Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling
Config.Output.Heat.PwmLo-	REAL	0.0	加熱のスケールされた低 PWM 出力値 Heat.PwmLowerScaling および Heat.PidLowerLimit は加熱のパルス幅調整された出力値(OutputHeat_PWM)に対して、PID 出力

werScaling			値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputHeat_PWM 値は、常に Heat.PwmUpperScaling と Heat.PwmLowerScaling の間に存在します。 Heat.PwmLowerScaling は、OutputHeat_PWM が加熱の出力として選択された場合にのみ有効です(Heat.Select = 1)。 100.0 ≥ Heat.PwmLowerScaling ≥ 0.0 Heat.PwmUpperScaling ≠ Heat.PwmLowerScaling
Config.Output.Heat.PerUpperScaling	REAL	27648.0	加熱のスケールされた高アナログ出力値 Heat.PerUpperScaling および Heat.PidUpperLimit は加熱のアナログ出力値(OutputHeat_PER)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputHeat_PER 値は、常に Heat.PerUpperScaling と Heat.PerLowerScaling の間に存在します。 Heat.PerUpperScaling は、OutputHeat_PER が加熱の出力として選択された場合にのみ有効です(Heat.Select = 2)。 32511.0 ≥ Heat.PerUpperScaling ≥ -32512.0 Heat.PerUpperScaling ≠ Heat.PerLowerScaling
Config.Output.Heat.PerLowerScaling	REAL	0.0	加熱のスケールされた低アナログ出力値 Heat.PerLowerScaling および Heat.PidLowerLimit は加熱のアナログ出力値(OutputHeat_PER)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputHeat_PER 値は、常に Heat.PerUpperScaling と Heat.PerLowerScaling の間に存在します。 Heat.PerLowerScaling は、OutputHeat_PER が加熱の出力として選択された場合にのみ有効です(Heat.Select = 2)。 32511.0 ≥ Heat.PerLowerScaling ≥ -32512.0 Heat.PerUpperScaling ≠ Heat.PerLowerScaling
Config.Output.Heat.MinimumOnTime	REAL	0.0	加熱(OutputHeat_PWM 出力)のパルス幅変調の最小オン時間 PWM パルスはこの値よりも決して短くなりません。 値は以下に四捨五入されます。 Heat.MinimumOnTime = n × CycleTime.Value Heat.MinimumOnTime は、加熱の出力 OutputHeat_PWM が選択された場合にのみ有効です(Heat.Select = 1)"。 100000.0 ≥ Heat.MinimumOnTime ≥ 0.0
Config.Output.Heat.MinimumOffTime	REAL	0.0	加熱(OutputHeat_PWM 出力)のパルス幅変調の最小オフ時間 PWM 一時停止はこの値よりも決して短くなりません。 値は以下に四捨五入されます。 Heat.MinimumOffTime = n × CycleTime.Value Heat.MinimumOffTime は、加熱の出力 OutputHeat_PWM が選択された場合にのみ有効です(Heat.Select = 1)"。

			100000.0 ≥ Heat.MinimumOffTime ≥ 0.0
Config.Output.Cool.Select	INT	1	冷却の出力値の選択 Config.Output.Cool.Select は冷却に使用する出力を指定します。 • Cool.Select = 0 - OutputCool が使用されます。 • Cool.Select = 1 -OutputCool および OutputCool_PWM が使用されます。 • Cool.Select = 2 - OutputCool および OutputCool_PER が使用されます。 使用されない出力は計算されず、既定値のままになります。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。
Config.Output.Cool.PwmPeriode	REAL	0.0	冷却のパルス幅変調の時間(OutputCool_PWM 出力)(秒単位): • Cool.PwmPeriode = 0.0 で、かつ、Config.AdvancedCooling = FALSE: 加熱の PID アルゴリズムのサンプリング時間 (Retain.CtrlParams.Heat.Cycle)が PWM の時間として使用されます。 • Cool.PwmPeriode = 0.0 で、かつ Config.AdvancedCooling = TRUE: 冷却の PID アルゴリズムのサンプリング時間(Retain.CtrlParams.Cool.Cycle)が PWM の時間として使用されます。 • Cool.PwmPeriode > 0.0: 値は PID_Temp サンプリング時間の整数倍に四捨五入され(CycleTime.Value)、PWM の時間として使用されます。 この設定を使用して、PID アルゴリズムの長いサンプリング時間によってプロセス値のスムージングを向上させることができます。 値は、以下の条件を満たす必要があります。 ○ Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle または Retain.CtrlParams.Heat.Cycle ○ Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime ○ Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。
Config.Output.Cool.PidUpperLimit	REAL	0.0	冷却の PID 出力値の上限値 値は 0.0 でなければなりません。 Cool.PidUpperLimit は、冷却の出力に対して、PID 出力値(PidOutputSum)のスケーリングのため、以下のパラメータと一緒に値ペアを形成します。 • Cool.LowerScaling FOR OutputCool • Cool.PwmLowerScaling FOR OutputCool_PWM • Cool.PerLowerScaling FOR OutputCool_PER

			この値を関連する出力で制限する場合は、これらのスケーリング値も調整する必要があります。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。 Cool.PidUpperLimit = 0.0
Config.Output.Cool.PidLowerLimit	REAL	-100.0	冷却の PID 出力値の下限値 冷却出力が有効なコントローラの場合(Config.ActivateCooling = TRUE)、PID 出力値(PidOutputSum)はこの下限値に制限されます。 Cool.PidLowerLimit は、冷却の出力に対して、PID 出力値(PidOutputSum)のスケーリングのため、以下のパラメータと一緒に値ペアを形成します。 • Cool.UpperScaling FOR OutputCool • Cool.PwmUpperScaling FOR OutputCool_PWM • Cool.PerUpperScaling FOR OutputCool_PER この値を関連する出力で制限する場合は、これらのスケーリング値も調整する必要があります。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。 Cool.PidLowerLimit < Cool.PidUpperLimit
Config.Output.Cool.UpperScaling	REAL	100.0	冷却のスケールされた高出力値 Cool.UpperScaling および Cool.PidLowerLimit は冷却の出力値(OutputCool)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputCool 値は、常に Cool.UpperScaling と Cool.LowerScaling の間に存在します。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。 Cool.UpperScaling ≠ Cool.LowerScaling
Config.Output.Cool.LowerScaling	REAL	0.0	冷却のスケールされた低出力値 Cool.LowerScaling および Cool.PidUpperLimit は冷却の出力値(OutputCool)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputCool 値は、常に Cool.UpperScaling と Cool.LowerScaling の間に存在します。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。 Cool.UpperScaling ≠ Cool.LowerScaling
Config.Output.Cool.PwmUpperScaling	REAL	100.0	冷却のスケールされた高 PWM 出力値 Cool.PwmUpperScaling および Cool.PidLowerLimit は冷却のパルス幅調整された出力値(OutputCool_PWM)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。

			OutputCool_PWM 値は、常に Cool.PwmUpperScaling と Cool.PwmLowerScaling の間に存在します。 Cool.PwmUpperScaling は、冷却出力が有効で(Config.Activate-Cooling = TRUE)、かつ OutputCool_PWM が冷却の出力として選択された場合(Cool.Select = 1)にのみ有効です。 100.0 ≥ Cool.PwmUpperScaling ≥ 0.0 Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling
Con-fig.Out-put.Cool.PwmLo-werScal-ing	REAL	0.0	冷却のスケールされた低 PWM 出力値 Cool.PwmLowerScaling および Cool.PidUpperLimit は冷却のパルス幅調整された出力値(OutputCool_PWM)に対して、PID 出力値(PidOutputSum)のスケーリングのために値ペアを形成します。 OutputCool_PWM 値は、常に Cool.PwmUpperScaling と CoolPwm.LowerScaling の間に存在します。 Cool.PwmLowerScaling は、冷却出力が有効で(Config.Activate-Cooling = TRUE)、かつ OutputCool_PWM が冷却の出力として選択された場合(Cool.Select = 1)にのみ有効です。 100.0 ≥ Cool.PwmLowerScaling ≥ 0.0 Cool.PwmUpperScaling ≠ Cool.PwmLowerScaling
Con-fig.Out-put.Cool.PerUp-perScal-ing	REAL	27648.0	冷却のスケールされた高アナログ出力値 Cool.PerUpperScaling および Cool.PidLowerLimit は冷却のアナログ出力値(OutputCool_PER)に対して、PID 出力値(PidOutput-Sum)のスケーリングのために値ペアを形成します。 OutputCool_PER 値は、常に Cool.PerUpperScaling と Cool.Per-LowerScaling の間に存在します。 Cool.PerUpperScaling は、冷却出力が有効で(Config.Activate-Cooling = TRUE)、かつ OutputCool_PER が冷却の出力として選択された場合(Cool.Select = 2)にのみ有効です。 32511.0 ≥ Cool.PerUpperScaling ≥ -32512.0 Cool.PerUpperScaling ≠ Cool.PerLowerScaling
Con-fig.Out-put.Cool.PerLo-werScal-ing	REAL	0.0	冷却のスケールされた低アナログ出力値 Cool.PerLowerScaling および Cool.PidUpperLimit は冷却のアナログ出力値(OutputCool_PER)に対して、PID 出力値(PidOutput-Sum)のスケーリングのために値ペアを形成します。 OutputCool_PER 値は、常に Cool.PerUpperScaling と Cool.Per-LowerScaling の間に存在します。 Cool.PerLowerScaling は、冷却出力が有効で(Config.Activate-Cooling = TRUE)、かつ OutputCool_PER が冷却の出力として選択された場合(Cool.Select = 2)にのみ有効です。 32511.0 ≥ Cool.PerLowerScaling ≥ -32512.0 Cool.PerUpperScaling ≠ Cool.PerLowerScaling
Con-fig.Out-put.Cool.	REAL	0.0	冷却のパルス幅変調の最小オン時間(OutputCool_PWM 出力) PWM パルスはこの値よりも決して短くなりません。

Minimu-mOnTime			値は以下に四捨五入されます。 Cool.MinimumOnTime = n × CycleTime.Value Cool.MinimumOnTime は、冷却の出力 OutputCool_PWM が選択された場合にのみ有効です(Cool.Select = 1)。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。 100000.0 ≥ Cool.MinimumOnTime ≥ 0.0
Con-fig.Out-put.Cool.Minimu-mOffTime	REAL	0.0	冷却のパルス幅変調の最小オフ時間(OutputCool_PWM 出力) PWM 一時停止はこの値よりも決して短くなりません。 値は以下に四捨五入されます。 Cool.MinimumOffTime = n × CycleTime.Value Cool.MinimumOffTime は、冷却の出力 OutputCool_PWM が選択された場合にのみ有効です(Cool.Select = 1)。 冷却出力が有効な場合のみ有効です(Config.ActivateCooling = TRUE)。 100000.0 ≥ Cool.MinimumOffTime ≥ 0.0
カスケードで PID_Temp を使用している場合、マスタコントローラおよびスレーブコントローラは、マスタパラメータとスレーブパラメータを使用して情報を交換します。 相互接続を作成する必要があります。 詳細については、マスタパラメータを参照してください。			
Con-fig.Cas-cade.Is-Master	BOOL	FALSE	カスケードでコントローラはマスタであり、スレーブセットポイントを提供します。 この PID_Temp インスタンスをカスケードでマスタコントローラとして使用している場合、IsMaster = TRUE に設定します。 マスタコントローラは、その出力によってスレーブコントローラのセットポイントを定義します。 PID_Temp インスタンスは、同時にマスタコントローラとスレーブコントローラになることができます。 コントローラをマスタコントローラとして使用する場合、冷却出力を無効にする必要があります(Config.ActivateCooling = FALSE)。
Con-fig.Cas-cade.Is-Slave	BOOL	FALSE	カスケードでコントローラはスレーブであり、マスタセットポイントを提供します。 この PID_Temp インスタンスをカスケードでスレーブコントローラとして使用している場合、IsSlave = TRUE に設定します。 スレーブコントローラはそのセットポイント(Setpoint パラメータ)をマスタコントローラの出力(OutputHeat パラメータ)から受信します。 PID_Temp インスタンスは、同時にマスタコントローラとスレーブコントローラになることができます。
Con-fig.Cas-cade.Anti-WindUp-Mode	INT	1	カスケードでのワインドアップ防止動作 オプションを以下に示します。 • ワインドアップ防止 = 0 AntiWindUp (ワインドアップ防止)機能は無効です。 マスタコントローラは、スレーブコントローラの限界値に応答しません。

			• ワインドアップ防止 = 1 マスタコントローラの積分動作は、比率「限界値のスレーブ」対「スレーブの数」に換算されます(「CountSlaves」パラメータ)。 つまり、制御応答に対して限界値の影響が減少します。 • ワインドアップ防止 = 2 マスタコントローラの積分動作は、スレーブコントローラが限界値に達すると直ちに停止します。 コントローラをマスタコントローラとして設定済みの場合にのみ有効です(Config.Cascade.IsMaster = TRUE)。
Config.Cascade.CountSlaves	INT	1	下位スレーブの数 ここには、このマスタコントローラからセットポイントを受信するすぐ下位のスレーブコントローラの数を入力します。 コントローラをマスタコントローラとして設定済みの場合にのみ有効です(Config.Cascade.IsMaster = TRUE)。 255 ≥ CountSlaves ≥ 1
CycleTime.StartEstimation	BOOL	TRUE	CycleTime.EnEstimation = TRUE の場合、CycleTime.StartEstimation = TRUE によって PID_Temp サンプリング時間の自動決定が開始されます(呼び出し側の OB のサイクルタイム)。 測定が完了すると、CycleTime.StartEstimation = FALSE が設定されます。
CycleTime.EnEstimation	BOOL	TRUE	CycleTime.EnEstimation = TRUE の場合、PID_Temp サンプリング時間が自動的に決定されます。 CycleTime.EnEstimation = FALSE の場合、サンプリング時間 PID_Temp は自動的に決定されず、CycleTime.Value を使用して手動で正しく設定する必要があります。
CycleTime.EnMonitoring	BOOL	TRUE	CycleTime.EnMonitoring = FALSE の場合、PID_Temp サンプリング時間はモニタされません。 PID_Temp をサンプリング時間内に実行できない場合、エラー(ErrorBits=0000800h)は出力されず、PID_Temp は ActivateRecoverMode によって設定されたとおりに応答しません。
CycleTime.Value	REAL	0.1	PID_Temp サンプリング時間(呼び出し側 OB のサイクルタイム)(秒単位) CycleTime.Value が自動的に測定されます。通常、呼び出し OB のサイクルタイムと同じ値です。
LoadBackUp= TRUE で、CtrlParamsBackUp 構造体から値を再読み込みできます。			
CtrlParamsBackUp.SetByUser	BOOL	FALSE	Retain.CtrlParams.SetByUser の保存された値。
CtrlParamsBackUp.Heat.Gain	REAL	1.0	加熱の保存された比例ゲイン
CtrlParamsBackUp.Heat.Ti	REAL	20.0	加熱用の保存された積分動作時間(秒単位)

CtrlParamsBackUp.Heat.Td	REAL	0.0	加熱用の保存された微分動作時間(秒単位)
CtrlParamsBackUp.Heat.TdFiltRatio	REAL	0.2	加熱用の保存された微分の遅延係数
CtrlParamsBackUp.Heat.PWeighting	REAL	1.0	加熱用の保存された重み付け比例動作
CtrlParamsBackUp.Heat.DWeighting	REAL	1.0	加熱用の保存された重み付け微分動作
CtrlParamsBackUp.Heat.Cycle	REAL	1.0	加熱用 PID アルゴリズムの保存されたサンプリング時間(秒単位)
CtrlParamsBackUp.Heat.ControlZone	REAL	3.402822e+38	加熱用の保存された制御ゾーン幅
CtrlParamsBackUp.Heat.DeadZone	REAL	0.0	加熱用の保存されたデッドバンド幅
CtrlParamsBackUp.Cool.Gain	REAL	1.0	冷却用の保存された比例ゲイン
CtrlParamsBackUp.Cool.Ti	REAL	20.0	冷却用の保存された積分動作時間(秒単位)
CtrlParamsBackUp.Cool.Td	REAL	0.0	冷却用の保存された微分動作時間(秒単位)
CtrlParamsBackUp.Cool.TdFiltRatio	REAL	0.2	冷却用の保存された微分の遅延係数
CtrlParamsBack-	REAL	1.0	冷却用の保存された比例動作重み係数

Up.Cool.PWeighting			
CtrlParamsBackUp.Cool.DWeighting	REAL	1.0	冷却用の保存された微分動作重み係数
CtrlParamsBackUp.Cool.Cycle	REAL	1.0	冷却用 PID アルゴリズムの保存されたサンプリング時間(秒単位)
CtrlParamsBackUp.Cool.ControlZone	REAL	3.402822e+38	冷却用の保存された制御ゾーン幅
CtrlParamsBackUp.Cool.DeadZone	REAL	0.0	冷却用の保存されたデッドバンド幅
PIDSelfTune.SUT.CalculateParamsHeat	BOOL	FALSE	制御システムの加熱分岐のプロパティは、加熱のプレチューニング中に保存されます。 SUT.CalculateParamsHeat = TRUE の場合、加熱の PID パラメータはこれらのプロパティに基づいて再計算されます(Retain.CtrlParams.Heat 構造体)。 これによって、調整を繰り返すことなく、パラメータ計算方法を変更できます(PIDSelfTune.SUT.TuneRuleHeat パラメータ)。 計算後、SUT.CalculateParamsHeat が FALSE に設定されます。 プレチューニングが成功した場合にのみ可能です(SUT.ProcParHeatOk = TRUE)。
PIDSelfTune.SUT.CalculateParamsCool	BOOL	FALSE	制御システムの冷却分岐のプロパティは、冷却の調整中に保存されます。 SUT.CalculateParamsCool = TRUE の場合、冷却の PID パラメータはこれらのプロパティに基づいて再計算されます(Retain.CtrlParams.Cool 構造体)。 これによって、調整を繰り返すことなく、パラメータ計算方法を変更できます(PIDSelfTune.SUT.TuneRuleCool パラメータ)。 計算後、SUT.CalculateParamsCool が FALSE に設定されます。 プレチューニングが成功した場合にのみ可能です(SUT.ProcParCoolOk = TRUE)。 Config.ActivateCooling = TRUE、かつ Config.AdvancedCooling = TRUE の場合にのみ有効です。
PIDSelfTune.SUT.TuneRuleHeat	INT	2	加熱のプレチューニングでの PID パラメータの計算方法 オプションを以下に示します。 • SUT.TuneRuleHeat = 0: PID から CHR • SUT.TuneRuleHeat = 1: PI から CHR • SUT.TuneRuleHeat = 2: CHR に対する温度プロセスの PID (SUT.TuneRuleHeat = 0 よりもオーバーシュートが少ない、より長く漸近的な制御応答になります)

			(CHR = チエン(Chien)、フローネス(Hrones)、レスウィック(Reswick)) SUT.TuneRuleHeat = 2 の場合のみ、制御ゾーン Retain.CtrlParams.Heat.ControlZone が加熱のプレチューニング中に自動的に設定されます。
PIDSelfTune.SUT.TuneRuleCool	INT	2	冷却のプレチューニングでの PID パラメータの計算方法 オプションを以下に示します。 • SUT.TuneRuleCool = 0: PID から CHR • SUT.TuneRuleCool = 1: PI から CHR • SUT.TuneRuleCool = 2: CHR に対する温度プロセスの PID (SUT.TuneRuleCool = 0 よりもオーバーシュートが少ない、より長く、むしろ無症候性制御応答になります) (CHR = チエン(Chien)、フローネス(Hrones)、レスウィック(Reswick)) SUT.TuneRuleCool = 2 の場合のみ、制御ゾーン Retain.CtrlParams.Cool.ControlZone が冷却のプレチューニング中に自動的に設定されます。 SUT.TuneRuleCool は、冷却出力と PID パラメータ切り替えが有効な場合のみ有効です(Config.ActivateCooling = TRUE, Config.AdvancedCooling = TRUE)。
PIDSelfTune.SUT.State	INT	0	SUT.State タグは、プレチューニングの現在のフェーズを示します。 • State = 0 プレチューニングの初期化 • State = 100: 加熱の標準偏差の計算 • State = 200: 冷却の標準偏差の計算 • State = 300: 加熱の変曲点の決定 • State = 400: 冷却の変曲点の決定 • State = 500: 変曲点に到達した後のセットポイントへの加熱を設定します。 • State = 600: 変曲点に到達した後のセットポイントへの冷却を設定します。 • State = 700: 加熱アクチュエータと冷却アクチュエータの効率を比較します。 • State = 800: 加熱および冷却が有効 • State = 900: 冷却が有効 • State = 1000: 加熱のスイッチオフ後の遅延時間を決定します。 • State = 9900 プレチューニングが成功 • State = 1 プレチューニングが失敗
PIDSelfTune.SUT.ProcParHeatOk	BOOL	FALSE	TRUE: 加熱のプレチューニングのプロセスパラメータの計算が正常に行われました。 このタグは調整中に設定されます。 加熱の PID パラメータの計算の場合、これは TRUE である必要があります。
PIDSelfTune.SU	BOOL	FALSE	TRUE: 冷却のプレチューニングに対するプロセスパラメータの計算が正常に行われました。

T.Proc-ParCoo-lOk			このタグは調整中に設定されます。 冷却に対する PID パラメータの計算の場合、これは TRUE である必要があります。
PIDSelf-Tune.SUT.Adapt-Delay-Time	INT	0	AdaptDelayTime タグは、操作ポイントでの加熱の遅延時間の適合を決定します(「加熱のプレチューニング」および「加熱および冷却のプレチューニング」について)。 オプションを以下に示します。 • SUT.AdaptDelayTime = 0: 遅延時間の適合なし SUT.State = 1000 フェーズはスキップされます。 このオプションを使用すると、 SUT.AdaptDelay-Time = 1 を使用した場合よりも、調整時間が短くなります。 • SUT.AdaptDelayTime = 1: 加熱を一時的にスイッチオフすることによって、SUT.State = 1000 フェーズでのセットポイントに対して遅延時間を適合させます。 このオプションを使用すると、SUT.AdaptDelayTime = 0 を使用した場合よりも、調整時間が長くなります。プロセス動作が操作ポイントに大幅に依存している場合は(非線形性)、制御応答を向上させることができます。 このオプションは、強い熱接続のある複数ゾーンアプリケーションに使用してはなりません。
PIDSelf-Tune.SUT.Cooling-Mode	INT	0	CoolingMode タグは、冷却パラメータを決定するために操作変数出力を決定します(加熱および冷却のプレチューニング用)。 オプションを以下に示します。 • SUT.CoolingMode = 0: セットポイントに達した後、加熱をスイッチをオフにして冷却のスイッチをオンにします。 SUT.State = 700 フェーズはスキップされます。 フェーズ SUT.State = 500 の後にフェーズ SUT.State = 900 が続きます。 このオプションは、冷却アクチュエータのゲインが、加熱アクチュエータのゲインと比較して低い場合に制御応答を向上できます。 これにより、SUT.CoolingMode = 1 または 2 を使用した場合よりも、調整時間が短くなります。 • SUT.CoolingMode = 1: セットポイントに達した後、加熱に加えて冷却のスイッチをオンにします。 SUT.State = 700 フェーズはスキップされます。 フェーズ SUT.State = 500 の後にフェーズ SUT.State = 800 が続きます。 このオプションは、冷却アクチュエータのゲインが、加熱アクチュエータのゲインと比較して高い場合に制御応答を向上できます。 • SUT.CoolingMode = 2: セットポイントまで加熱した後、加熱をスイッチオフするかどうかの決定がフェーズ SUT.State = 700 で自動的に行われ

			ます。 フェーズ SUT.State = 500 の後にフェーズ SUT.State = 700 が続き、その後 SUT.State = 800 または SUT.State = 900 になります。 このオプションは、オプション 0 および 1 よりも長い時間がかかります。
PIDSelf-Tune.TIR.RunIn	BOOL	FALSE	RunIn タグを使用して、微調整のシーケンスを自動モードからの開始中に指定します。 • RunIn = FALSE 自動モードで微調整が開始されると、システムは既存の PID パラメータを使用してセットポイントを制御します (TIR.State = 500 または 600)。 この場合のみ、微調整が開始します。 • RunIn = TRUE PID_Temp は、最小または最大出力値によってセットポイントに到達しようとします(TIR.State = 300 または 400)。 この結果、オーバーシュートが増大する可能性があります。 微調整が自動的に開始します。 微調整後、RunIn が FALSE に設定されます。 無効または手動モードから微調整を開始する間に、PID_Temp は RunIn = TRUE で記述されたとおりに応答します。
PIDSelf-Tune.TIR.Calcula-teParam-sHeat	BOOL	FALSE	制御システムの加熱分岐のプロパティは、加熱の微調整中に保存されます。 TIR.CalculateParamsHeat= TRUE の場合、加熱の PID パラメータはこれらのプロパティに基づいて再計算されます(Retain.CtrlParams.Heat 構造体)。 これによって、調整を繰り返すことなく、パラメータ計算方法を変更できます(PIDSelf-Tune.TIR.TuneRuleHeat パラメータ)。 計算後、TIR.CalculateParamsHeat が FALSE に設定されます。 事前に加熱の微調整が成功した場合にのみ可能です(TIR.Proc-ParHeatOk = TRUE)。
PIDSelf-Tune.TIR.Calcula-teParams-Cool	BOOL	FALSE	制御システムの冷却分岐のプロパティは、冷却の微調整中に保存されます。 TIR.CalculateParamsCool= TRUE の場合、冷却の PID パラメータはこれらのプロパティに基づいて再計算されます(Retain.CtrlParams.Cool 構造体)。 これによって、調整を繰り返すことなく、パラメータ計算方法を変更できます(PIDSelf-Tune.TIR.TuneRuleCool パラメータ)。 計算後、TIR.CalculateParamsCool が FALSE に設定されます。 事前に冷却の微調整が成功した場合にのみ可能です(TIR.Proc-ParCoolOk = TRUE)。 Config.ActivateCooling = TRUE、かつ Config.AdvancedCooling = TRUE の場合にのみ有効です。
PIDSelf-Tune.TIR.TuneRu-leHeat	INT	0	加熱の微調整中のパラメータ計算方法 オプションを以下に示します。 • TIR.TuneRuleHeat = 0: PID 自動 • TIR.TuneRuleHeat = 1: PID 高速(TIR.TuneRuleHeat = 2 を使用した場合よりも出力値が高振幅で、制御応答が高速) • TIR.TuneRuleHeat = 2: PID 低速(TIR.TuneRuleHeat = 1 を使用した場合よりも出力値が低振幅で、制御応答が低速)

			• TIR.TuneRuleHeat = 3: ZN PID • TIR.TuneRuleHeat = 4: ZN PI • TIR.TuneRuleHeat = 5: ZN P (ZN=ジーグラ・ニコルス) TIR.CalculateParamsHeat および TIR.TuneRuleHeat = 0、1 または 2 を使用して加熱の PID パラメータの計算を繰り返すには、前の微調整も TIR.TuneRuleHeat = 0、1 または 2 を使用して実行している必要があります。それ以外の場合、TIR.TuneRuleHeat = 3 が使用されます。 TIR.CalculateParamsHeat および TIR.TuneRuleHeat = 3、4 または 5 を使用した加熱の PID パラメータの再計算は、常に可能です。
PIDSelf-Tune.TIR.TuneRuleCool	INT	0	冷却の微調整中のパラメータ計算方法 オプションを以下に示します。 • TIR.TuneRuleCool = 0: PID 自動 • TIR.TuneRuleCool = 1: PID 高速(TIR.TuneRuleCool = 2 を使用した場合よりも出力値が高振幅で、制御応答が高速) • TIR.TuneRuleCool = 2: PID 低速(TIR.TuneRuleCool = 1 を使用した場合よりも出力値が低振幅で、制御応答が低速) • TIR.TuneRuleCool = 3: ZN PID • TIR.TuneRuleCool = 4: ZN PI • TIR.TuneRuleCool = 5: ZN P (ZN=ジーグラ・ニコルス) TIR.CalculateParamsCool および TIR.TuneRuleCool = 0､1 または 2 を使用して冷却の PID パラメータの計算を繰り返すには、前の微調整も TIR.TuneRuleCool = 0、1 または 2 を使用して実行している必要があります。それ以外の場合、TIR.TuneRuleCool = 3 が使用されます。 TIR.CalculateParamsCool および TIR.TuneRuleCool = 3､4 または 5 を使用した冷却の PID パラメータの再計算は、常に可能です。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(ConfigActivateCooling = TRUE かつ Config.AdvancedCooling = TRUE)。
PIDSelf-Tune.TIR.State	INT	0	TIR.State タグは、「微調整」の現在のフェーズを示します。 • State = 0: 微調整の初期化 • State = 100: 加熱の標準偏差の計算 • State = 200: 冷却の標準偏差の計算 • State = 300: 2 ステップコントロールによる加熱のセットポイント到達の試行 • State = 400: 2 ステップコントロールによる冷却のセットポイント到達の試行 • State = 500: PID 制御による加熱のセットポイント到達の試行 • State = 600: PID 制御による冷却のセットポイント到達の試行

			• State = 700: 加熱の標準偏差の計算 • State = 800: 冷却の標準偏差の計算 • State = 900: 振動の決定と加熱のパラメータの計算 • State = 1000: 振動の決定と冷却のパラメータの計算 • State = 9900: 微調整が成功 • State = 1: 微調整が成功
PIDSelf-Tune.TIR.ProcPar-HeatOk	BOOL	FALSE	TRUE: 加熱の微調整のプロセスパラメータの計算が正常に行われました。 このタグは調整中に設定されます。 加熱の PID パラメータの計算の場合、これが一致する必要があります。
PIDSelf-Tune.TIR.ProcPar-CoolOk	BOOL	FALSE	TRUE: 冷却の微調整のプロセスパラメータの計算が正常に行われました。 このタグは調整中に設定されます。 冷却に対する PID パラメータの計算の場合、これが一致する必要があります。
PIDSelf-Tune.TIR.Output-tOffse-tHeat	REAL	0.0	加熱の PID 出力値の調整オフセット TIR.OutputOffsetHeat が、加熱分岐の PidOutputSum から発生する値に追加されます。 加熱の出力で正のオフセットを受信するには、TIR.OutputOff-setHeat に正の値を定義します。 加熱の出力で発生する値は、設定済み出力スケーリングの結果です(Struktur Config.Output.Heat)。 この調整オフセットは、冷却の微調整に、冷却出力と PID パラメータ切り替えを有効にしたコントローラで使用できます (Config.ActivateCooling = TRUE、Config.AdvancedCooling = TRUE)。 調整対象のセットポイントで冷却の出力が有効ではない場合(PidOutputSum > 0.0)、冷却の微調整は不可能です。 この場合は、調整の開始前に定常状態のセットポイントで PID 出力値(PidOutputSum)よりも大きな加熱の正の調整オフセットを定義します。 このステップにより加熱の出力での値が増加し、冷却の出力が有効になります(PidOutputSum < 0.0)。 冷却の微調整が実行可能になりました。 微調整が完了すると、TIR.OutputOffsetHeat は 0.0 にリセットされます。 TIR.OutputOffsetHeat での 1 ステップの大きな変更により、一時的なオーバーシュートになる可能性があります。 Config.Output.Heat.PidUpperLimit ≥ PIDSelfTune.TIR.Output-tOffsetHeat ≥ Config.Output.Heat.PidLowerLimit
PIDSelf-Tune.TIR.Output-tOffset-Cool	REAL	0.0	冷却の PID 出力値の調整オフセット TIR.OutputOffsetCool が、冷却分岐の PidOutputSum から発生する値に追加されます。 冷却の出力で正のオフセットを受信するには、TIR.OutputOff-setCool に負の値を定義します。

			冷却の出力で発生する値は、設定済み出力スケーリングの結果です(Struktur Config.Output.Coool)。 この調整オフセットは、加熱の微調整で、冷却出力が有効なコントローラに使用できます(Config.ActivateCooling = TRUE)。調整対象のセットポイントで加熱の出力が有効ではない場合(PidOutputSum < 0.0)、加熱の微調整は不可能です。 この場合は、調整の開始前に定常状態のセットポイントで PID 出力値(PidOutputSum)よりも小さな冷却の負の調整オフセットを定義します。 このステップにより冷却の出力での値が増加し、加熱の出力が有効になります(PidOutputSum > 0.0)。 加熱の微調整が実行可能になりました。 微調整が完了すると、TIR.OutputOffsetCool は 0.0 にリセットされます。 TIR.OutputOffsetCool での 1 ステップの大きな変更により、一時的なオーバーシュートになる可能性があります。 Config.Output.Cool.PidUpperLimit ≥ PIDSelfTune.TIR.OutputOffsetCool ≥ Config.Output.Cool.PidLowerLimit
PIDSelfTune.TIR.WaitForControlIn	BOOL	FALSE	セットポイントに達した後、微調整で待機中 TIR.WaitForControlIn = TRUE の場合、TIR.FinishControlIn で FALSE -> TRUE エッジが与えられるまで、微調整はセットポイント到達(TIR.State = 500 または 600)と標準偏差の計算(TIR.State = 700 または 800)の間で待機します。 TIR.WaitForControlIn は、個々のゾーンの調整を同期するため複数ゾーンアプリケーションでの複数コントローラの同時微調整に使用できます。 これにより、実際の調整が開始する前にすべてのゾーンがセットポイントに到達するようになります。 調整に関するゾーン間の熱接続の影響をこうして減少できます。 TIR.WaitForControlIn は、微調整が PIDSelfTune.TIR.RunIn = FALSE により自動モードから開始した場合にのみ有効です。
PIDSelfTune.TIR.ControlInReady	BOOL	FALSE	TIR.WaitForControlIn = TRUE の場合、PID_Temp はセットポイントに達すると直ちに TIR.ControlInReady = TRUE を設定し、TIR.FinishControlIn で FALSE -> TRUE エッジが与えられるまで、追加の調整ステップを待機します。
PIDSelfTune.TIR.FinishControlIn	BOOL	FALSE	TIR.ControlInReady = TRUE の場合、TIR.FinishControlIn で FALSE -> TRUE エッジが与えられると、待機を停止し微調整を再開します。
PIDCtr.IOutputOld	REAL	0.0	最後のサイクルでの積分動作
Retain.CtrlParams.SetByUser	BOOL	FALSE	PID パラメータを構成エディタで手動入力した場合、SetByUser = TRUE になります。 このパラメータはエディタの表示に使用され、制御アルゴリズムには影響を与えません。 SetByUser は保持されます。
Retain.CtrlParams.Heat.Gain	REAL	1.0	加熱用の有効な比例ゲイン Heat.Gain は保持されます。 Heat.Gain ≥ 0.0

Re-tain..CtrlPar-ams.Heat.Ti	REAL	20.0	加熱用の有効な積分動作時間(秒単位) 加熱の積分動作は、Heat.CtrlParams.Ti = 0.0 でスイッチオフされます。 Heat.Ti は保持されます。 100000.0 ≥ Heat.Ti ≥ 0.0
Re-tain.CtrlPar-ams.Heat.Td	REAL	0.0	加熱用の有効な微分動作時間(秒単位) 加熱の微分動作は、Heat.CtrlParams.Td = 0.0 でスイッチオフされます。 Heat.Td は保持されます。 100000.0 ≥ Heat.Td ≥ 0.0
Re-tain.CtrlPar-ams.Heat.TdFiltRa-tio	REAL	0.2	加熱用の有効な微分の遅延係数 微分遅延係数は、微分動作の効果を遅らせます。 微分遅延 = 微分動作時間 × 微分遅延係数 • 0.0: 微分動作は 1 サイクルの間のみ有効であるため、ほとんど効果はありません。 • 0.5: この値は、主要な時定数が 1 つの制御システムで実際に有効であることが実証されています。 • > 1.0: この係数が大きいほど、微分動作の効果の遅延が長くなります。 Heat.TdFiltRatio は保持されます。 Heat.TdFiltRatio ≥ 0.0
Re-tain.CtrlPar-ams.Heat.PWeight-ing	REAL	1.0	加熱用の有効な重み付け比例動作 比例動作は、セットポイントの変更によって弱くなることがあります。 0.0 ~ 1.0 の値が適用できます。 • 1.0: セットポイントの変更に対する比例動作は完全に有効です。 • 0.0: セットポイントの変更に対する比例動作は有効ではありません。 プロセス値が変更されたとき、比例動作は常に完全に有効です。 Heat.PWeighting は保持されます。 1.0 ≥ Heat.PWeighting ≥ 0.0
Re-tain.CtrlPar-ams.Heat.DWeight-ing	REAL	1.0	加熱用の有効な重み付け微分動作 微分動作は、セットポイントの変更によって弱くなることがあります。 0.0 ~ 1.0 の値が適用できます。 • 1.0: セットポイントの変更時に、微分動作が完全に有効です。 • 0.0: セットポイントの変更時に、微分動作が有効ではありません。 プロセス値が変更されたとき、微分動作は常に完全に有効です。 Heat.DWeighting は保持されます。

			1.0 ≥ Heat.DWeighting ≥ 0.0
Retain.CtrlParams.Heat.Cycle	REAL	1.0	加熱用 PID アルゴリズムの有効なサンプリング時間(秒単位) CtrlParams.Heat.Cycle が調整時に計算され、CycleTime.Value の整数倍に丸められます。 Config.Output.Heat.PwmPeriode = 0.0 の場合、Heat.Cycle が加熱のパルス幅変調の時間として使用されます。 Config.Output.Cool.PwmPeriode = 0.0、かつ Config.AdvancedCooling = FALSE の場合、Heat.Cycle が冷却のパルス幅変調の時間として使用されます。 Heat.Cycle は保持されます。 100000.0 ≥ Heat.Cycle > 0.0
Retain.CtrlParams.Heat.ControlZone	REAL	3.402822e+38	加熱用の有効な制御ゾーン幅 加熱の制御ゾーンは、Heat.ControlZone = 3.402822e+38 でスイッチオフされます。 Heat.ControlZone は、PIDSelfTune.SUT.TuneRuleHeat = 2 がパラメータの計算方法として選択された場合にのみ、加熱のプレチューニングまたは加熱および冷却のプレチューニング中に自動的に設定されます。 冷却出力が無効にされたコントローラ(Config.ActivateCooling = FALSE)または冷却出力と冷却係数が有効にされたコントローラ(Config.AdvancedCooling = FALSE)の場合、制御ゾーンは Setpoint – Heat.ControlZone と Setpoint + Heat.ControlZone の間に対称的に存在します。 冷却出力と PID パラメータの切り替えが有効にされたコントローラ(Config.ActivateCooling = TRUE、Config.AdvancedCooling = TRUE)の場合、制御ゾーンは Setpoint – Heat.ControlZone と Setpoint + Cool.ControlZone の間に存在します。 Heat.ControlZone は保持されます。 Heat.ControlZone > 0.0
Retain.CtrlParams.Heat.DeadZone	REAL	0.0	加熱用の有効なデッドバンド幅(PID パラメータを参照) 加熱のデッドバンドは、Heat.DeadZone = 0.0 でスイッチオフされます。 Heat.DeadZone は自動的に設定されないか、調整中に調整されます。 Heat.DeadZone を手動で正しく設定する必要があります。 冷却出力が無効にされたコントローラ(Config.ActivateCooling = FALSE)または冷却出力と冷却係数が有効にされたコントローラ(Config.AdvancedCooling = FALSE)の場合、デッドバンドは Setpoint – Heat.DeadZone と Setpoint + Heat.DeadZone の間に対称的に存在します。 冷却出力と PID パラメータの切り替えが有効にされたコントローラ(Config.ActivateCooling = TRUE、Config.AdvancedCooling = TRUE)の場合、デッドバンドは Setpoint – Heat.DeadZone と Setpoint + Cool.DeadZone の間に存在します。 Heat.DeadZone は保持されます。

			Heat.DeadZone ≥ 0.0
Retain.CtrlParams.Cool.Gain	REAL	1.0	冷却用の有効な比例ゲイン Cool.Gain は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.AdvancedCooling = TRUE)。 Cool.Gain ≥ 0.0
Retain.CtrlParams.Cool.Ti	REAL	20.0	冷却用の有効な積分動作時間(秒単位) 冷却の積分動作は、Cool.CtrlParams.Ti = 0.0 でスイッチオフされます。 Cool.Ti は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.AdvancedCooling = TRUE)。 100000.0 ≥ Cool.Ti ≥ 0.0
Retain.CtrlParams.Cool.Td	REAL	0.0	冷却用の有効な微分動作時間(秒単位) 冷却の微分動作は、Cool.CtrlParams.Td = 0.0 でスイッチオフされます。 Cool.Td は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.AdvancedCooling = TRUE)。 100000.0 ≥ Cool.Td ≥ 0.0
Retain.CtrlParams.Cool.TdFiltRatio	REAL	0.2	冷却用の有効な微分の遅延係数 微分遅延係数は、微分動作の効果を遅らせます。 微分遅延 = 微分動作時間 × 微分遅延係数 • 0.0: 微分動作は 1 サイクルの間のみ有効であるため、ほとんど効果はありません。 • 0.5: この値は、主要な時定数が 1 つの制御システムで実際に有効であることが実証されています。 • > 1.0: この係数が大きいほど、微分動作の効果の遅延が長くなります。 Cool.TdFiltRatio は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.AdvancedCooling = TRUE)。 Cool.TdFiltRatio ≥ 0.0
Retain.CtrlParams.Cool.PWeighting	REAL	1.0	冷却用の有効な重み付け比例動作 比例動作は、セットポイントの変更によって弱くなることがあります。 0.0 ~ 1.0 の値が適用できます。 • 1.0: セットポイントの変更に対する比例動作は完全に有効です。

			• 0.0: セットポイントの変更に対する比例動作は有効ではありません。 プロセス値が変更されたとき、比例動作は常に完全に有効です。 Cool.PWeighting は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.Advanced-Cooling = TRUE)。 1.0 ≥ Cool.PWeighting ≥ 0.0
Retain.CtrlParams.Cool.DWeighting	REAL	1.0	冷却用の有効な重み付け微分動作 微分動作は、セットポイントの変更によって弱くなることがあります。 0.0 ~ 1.0 の値が適用できます。 • 1.0: セットポイントの変更時に、微分動作が完全に有効です。 • 0.0: セットポイントの変更時に、微分動作が有効ではありません。 プロセス値が変更されたとき、微分動作は常に完全に有効です。 Cool.DWeighting は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.Advanced-Cooling = TRUE)。 1.0 ≥ Cool.DWeighting ≥ 0.0
Retain.CtrlParams.Cool.Cycle	REAL	1.0	冷却用 PID アルゴリズムの有効なサンプリング時間(秒単位) CtrlParams.Cool.Cycle が調整時に計算され、CycleTime.の整数倍に丸められます。 Config.Output.Cool.PwmPeriode = 0.0、かつ Config.Advanced-Cooling = TRUE の場合、Cool.Cycle が冷却のパルス幅変調の時間として使用されます。 Config.Output.Cool.PwmPeriode = 0.0、かつ Config.Advanced-Cooling = FALSE の場合、Heat.Cycle が冷却のパルス幅変調の時間として使用されます。 Cool.Cycle は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.Advanced-Cooling = TRUE)。 100000.0 ≥ Cool.Cycle > 0.0
Retain.CtrlParams.Cool.ControlZone	REAL	3.402822e+38	冷却用の有効な制御ゾーン幅 冷却の制御ゾーンは、 Cool.ControlZone = 3.402822e+38 でスイッチオフされます。 Cool.ControlZone は、PIDSelfTune.SUT.TuneRuleCool = 2 がパラメータの計算方法として選択された場合にのみ、冷却のプレチューニングまたは加熱および冷却のプレチューニング中に自動的に設定されます。 Cool.ControlZone は保持されます。

			冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.Advanced-Cooling = TRUE)。 Cool.ControlZone > 0.0
Re-tain.CtrlPar-ams.Cool.Dead-Zone	REAL	0.0	冷却用の有効なデッドバンド幅(PID パラメータを参照) 冷却のデッドバンドは、Cool.DeadZone = 0.0 でスイッチオフされます。 Cool.DeadZone は自動的に設定されないか、調整中に調整されます。 Cool.DeadZone を手動で正しく設定する必要があります。 Cool.DeadZone は保持されます。 冷却出力および PID パラメータの切り替えが有効な場合にのみ有効(Config.ActivateCooling = TRUE かつ Config.Advanced-Cooling = TRUE)。 Cool.DeadZone ≥ 0.0

注記
PID コントローラの誤動作を防止するには、「無効」モードでこの表に記載されているタグを変更します。

PID_Temp の状態およびモードパラメータ

パラメータの相互関係

State パラメータは、PID コントローラの現在の動作モードを示します。 State パラメータを変更することはできません。

ModeActivate の立ち上がりエッジで、PID_Temp が ModeIN-OUT パラメータに保存された動作モードに切り替わります。

Heat.EnableTuning および Cool.EnableTuning は、加熱または冷却で調整を行うかどうかをプレチューニングおよび微調整に指定します。

CPU がオンになるか、Stop から RUN モードに切り替わると、PID_Temp が Mode パラメータに保存された動作モードで開始されます。 PID_Temp を「無効」モードのままにするには、RunModeByStartup = FALSE を設定してください。

値の意味

State / Mode	動作モードの説明
0	**無効** 以下の出力値が「無効」モードで出力されます。 • PID 出力値として 0.0 (PidOutputSum) • 加熱の出力値(OutputHeat)および冷却の出力値(OutputCool)として 0.0 • 加熱のアナログ出力値(OutputHeat_PER)および冷却のアナログ出力値(OutputCool_PER)として 0 • 加熱の PWM 出力値(OutputHeat_PWM)および冷却の PWM 出力値(OutputCool_PWM)として FALSE これは、構造体 Config.Output.Heat および Config.Output.Cool 内の設定済み出力値の限界値およびスケーリングに依存しません。
1	**プレチューニング** プレチューニングでは、出力値のジャンプ変動に対するプロセス応答が決定され、変曲点が検索されます。 PID パラメータは、最大立ち上がり速度と制御システムのデッドタイムから計算されます。 プレチューニングと微調整を実行すると最良の PID パラメータを取得できます。 PID_Temp は、設定に応じて異なるプレチューニングタイプを提供します。 • 加熱のプレチューニング: 加熱の PID パラメータが計算される、加熱の出力値の出力でジャンプ変更が出力され(Retain.CtrlParams.Heat 構造体)、セットポイントへの制御が自動モードで行われます。 プロセス動作が操作ポイントに強く依存する場合は、PIDSelfTune.SUT.AdaptDelayTime を使用して遅延時間の適応を有効にすることができます。 • 加熱および冷却のプレチューニング: ジャンプ変更が、加熱の出力値で出力されます。 プロセス値がセットポイントに近くなると、直ちにジャンプ変更が冷却の出力値で出力されます。 加熱の PID パラメータ(Retain.CtrlParams.Heat 構造体)と冷却の PID パラメータ(Retain.CtrlParams.Cool

構造体)が計算されます。 その後、セットポイントへの制御が自動モードで行われます。

プロセス動作が操作ポイントに強く依存する場合は、PIDSelfTune.SUT.AdaptDelay-Time を使用して遅延時間の適応を有効にすることができます。

加熱アクチュエータと比較した冷却アクチュエータの効果に応じて、調整の品質は、調整中に加熱出力と冷却出力が同時に操作されるかどうかにより影響を受ける可能性があります。 これは、PIDSelfTune.SUT.CoolingMode を使用して指定できます。

- プレチューニング冷却:

  ジャンプ変更が冷却の出力値で出力され、冷却の PID パラメータが計算されます (Struktur Retain.CtrlParams.Cool)。 その後、セットポイントへの制御が自動モードで行われます。

加熱および冷却の PID パラメータを調整する場合、「加熱のプレチューニング」に続いて「プレチューニング冷却」を行うほうが、「加熱および冷却のプレチューニング」を使用するよりも制御応答の向上を期待できます。 ただし、2 ステップのプレチューニングにはより長い時間がかかります。

プレチューニングの一般的な要件:

- サイクリック割り込み OB で PID_Temp 命令が呼び出されていること。
- 無効(State = 0)、手動モード(State = 4)、または自動モード(State = 3)であること。
- ManualEnable = FALSE
- Reset = FALSE
- セットポイントとプロセス値が、設定済みの制限範囲内であること。

加熱のプレチューニングの要件:

- Heat.EnableTuning = TRUE
- Cool.EnableTuning = FALSE
- プロセス値は、セットポイントに近すぎる値であってはいけません。

  |Setpoint - Input| > 0.3 * |Config.InputUpperLimit - Config.InputLowerLimit| および

  |Setpoint - Input| > 0.5 * |Setpoint|
- セットポイントがプロセス値よりも大きいこと。

  Setpoint > Input

加熱および冷却のプレチューニングの要件:

- Heat.EnableTuning = TRUE·
- Cool.EnableTuning = TRUE
- 冷却出力が有効になっていること(Config.ActivateCooling = TRUE)。
- PID パラメータの切り替えが有効になっていること(Config.AdvancedCooling = TRUE)。
- プロセス値は、セットポイントに近すぎる値であってはいけません。

  |Setpoint - Input| > 0.3 * |Config.InputUpperLimit - Config.InputLowerLimit| および

  |Setpoint - Input| > 0.5 * |Setpoint|
- セットポイントがプロセス値よりも大きいこと。

  Setpoint > Input

	冷却のプレチューニングの要件: • Heat.EnableTuning = FALSE· • Cool.EnableTuning = TRUE· • 冷却出力が有効になっていること(Config.ActivateCooling = TRUE)。 • PID パラメータの切り替えが有効になっていること(Config.AdvancedCooling = TRUE)。 • 「加熱のプレチューニング」または「加熱および冷却のプレチューニング」が、できれば同じセットポイントで正常に行われていること(PIDSelfTune.SUT.ProcParHeatOk = TRUE)。 • プロセス値は、セットポイントに近い値でなければなりません。 |Setpoint - Input| < 0.05 * |Config.InputUpperLimit - Config.InputLowerLimit| プロセス値が安定しているほど、PID パラメータの計算が簡単になり、結果が正確になります。 プロセス値の上昇率がノイズと比べて大幅に高ければ、プロセス値のノイズは許容できます。 これは、動作モード「無効」または「手動モード」で最も起こりやすくなります。 セットポイントが、CurrentSetpoint タグで固定されます。 調整は以下の場合にキャンセルされます。 • Setpoint > CurrentSetpoint + CancelTuningLevel または • Setpoint < CurrentSetpoint - CancelTuningLevel PID パラメータの計算方法は、PIDSelfTune.SUT.TuneRuleHeat および PIDSelfTune.SUT.TuneRuleCool を使用して、加熱および冷却について個別に指定できます。 PID パラメータは再計算される前に CtrlParamsBackUp 構造体でバックアップされ、LoadBackUp で再度有効になります。 プレチューニングが正常に行われた後、自動モードへの切り替えが行われます。 プレチューニングが正常に行われなかった場合、モードへの切り替えは ActivateRecoverMode によって決定されます。 プレチューニングのフェーズは、 PIDSelfTune.SUT.State で示されます。
2	**微調整** 微調整は、プロセス値の一定の制限された振動を生成します。この振動の振幅と周波数から、動作点について PID パラメータが調整されます。 通常は微調整に基づく PID パラメータの方が、プレチューニングに基づく PID パラメータよりも優れたマスタコントロールと外乱特性を実現します。 プレチューニングと微調整を実行すると最良の PID パラメータを取得できます。 PID_Temp は、プロセス値のノイズよりも大きな振動の生成を自動的に試行します。 プロセス値の安定性によって微調整が受ける影響は最小限です。 PID_Temp は、設定に応じて異なる微調整タイプを提供します。 • 加熱の微調整: PID_Temp は、加熱の出力値での周期的変更によってプロセス値の振動を生成し、加熱の PID パラメータを計算します(Struktur Retain.CtrlParams.Heat)。 • 冷却の微調整:

PID_Temp は、冷却の出力値での周期的変更によってプロセス値の振動を生成し、冷却の PID パラメータを計算します(Struktur Retain.CtrlParams.Cool)。

加熱/冷却コントローラの一時的な調整オフセット

PID_Temp を加熱/冷却コントローラとして使用する場合(Config.ActivateCooling = TRUE)、セットポイントの PID 出力値 (PidOutputSum)は、プロセス値振動を生成し、微調整を成功させるために以下の要件を満たす必要があります。

- 加熱の微調整に正の PID 出力値があること
- 冷却の微調整に負の PID 出力値があること

この要件を満たさない場合、反対の効果がある出力で出力される、微調整の一時的なオフセットを定義することができます。

- 加熱の微調整付き冷却出力のオフセット(PIDSelfTune.TIR.OutputOffsetCool)。

  調整の開始前に定常状態のセットポイントで PID 出力値(PidOutputSum)よりも小さな冷却の負の調整オフセットを定義します。
- 冷却の微調整付き加熱出力のオフセット(PIDSelfTune.TIR.OutputOffsetHeat)。

  調整の開始前に定常状態のセットポイントで PID 出力値(PidOutputSum)よりも大きな加熱の正の調整オフセットを定義します。

定義したオフセットは、プロセス値がセットポイントに留まるように PID アルゴリズムによってバランスを取られます。 つまり、オフセットのサイズは、上記に記載した要件を満たすように PID 出力値によって適切に調整することができます。

オフセットの定義時にプロセス値のオーバーシュート増大を防止するため、このサイズは複数のステップで増加することもできます。

PID_Temp が微調整モードで存在する場合、調整オフセットはリセットされます。

冷却の微調整のオフセットの定義例:

- オフセットなし:
  - セットポイント = プロセス値 (ScaledInput) = 80°C
  - PID 出力値(PidOutputSum) = 30.0
  - 加熱の出力値(OutputHeat) = 30.0
  - 冷却の出力値(OutputCool) = 0.0

    セットポイント周囲のプロセス値の振動を冷却出力のみで作成することはできません。

    ここでは微調整が失敗します。
- 加熱出力のオフセット定義付き(PIDSelfTune.TIR.OutputOffsetHeat) = 80.0
  - Setpoint = プロセス値(ScaledInput) = 80°C
  - PID 出力値(PidOutputSum) = -50.0
  - 加熱の出力値(OutputHeat) = 80.0
  - 冷却の出力値(OutputCool) = -50.0

    加熱出力にオフセットを定義することによって、冷却出力はセットポイント周囲のプロセス値の振動を作成できるようになります。

    つまり、微調整を正常に行うことができます。

微調整の一般的な必要条件:

- サイクリック割り込み OB で PID_Temp 命令が呼び出されていること。
- 外乱が予測されないこと。
- セットポイントとプロセス値が、設定済みの制限範囲内であること。

- 動作点で制御ループが安定していること。プロセス値がセットポイントに一致したときに動作点に達すること。
- ManualEnable = FALSE
- Reset = FALSE
- 自動(State = 3)、無効(State = 0)、または手動(State = 4)モードであること。

加熱の微調整の必要条件:

- Heat.EnableTuning = TRUE
- Cool.EnableTuning = FALSE
- PID_Temp が加熱/冷却コントローラとして設定されている場合(Config.ActivateCooling = TRUE)、調整を行う操作ポイントで加熱出力が有効である必要があります(PidOutputSum > 0.0 (調整オフセットを参照))。

冷却の微調整の必要条件:

- Heat.EnableTuning = FALSE
- Cool.EnableTuning = TRUE
- 冷却出力が有効になっていること(Config.ActivateCooling = TRUE)。
- PID パラメータの切り替えが有効になっていること(Config.AdvancedCooling = TRUE)。
- 調整を行う操作ポイントで冷却出力が有効である必要があります(PidOutputSum < 0.0 (調整オフセットを参照))。

微調整の経過は、開始したモードにより決まります。

- 自動モード(State = 3)PIDSelfTune.TIR.RunIn = FALSE (既定)。

  調整を使用して既存の PID パラメータを改善する場合は、自動モードで微調整を開始します。

  PID_Temp は、制御ループが安定し、微調整の必要条件が満たされるまでは、既存の PID パラメータを使用してシステムを制御します。 制御ループが安定して微調整の必要条件が満たされた場合のみ、微調整が開始します。
- 無効(State = 0)、手動モード(State = 4)、または自動モード(State = 3)PIDSelfTune.TIR.RunIn = TRUE に設定。

  最小または最大出力値でセットポイントに到達することを試みます。
  - 加熱の微調整の最小または最大出力値を使用
  - 冷却の微調整の最小または最大出力値を使用

  この結果、オーバーシュートが増大する可能性があります。 セットポイントに達すると微調整が開始されます。

  セットポイントに達することができない場合、PID_Temp は自動的に調整を中止しません。

セットポイントが、CurrentSetpoint タグで固定されます。調整は以下の場合にキャンセルされます。

- Setpoint > CurrentSetpoint + CancelTuningLevel

  または
- Setpoint < CurrentSetpoint - CancelTuningLevel

PID パラメータの計算方法は、PIDSelfTune.TIR.TuneRuleHeat および PIDSelfTune.TIR.TuneRuleCool を使用して、加熱および冷却について個別に指定できます。

	PID パラメータは再計算される前に CtrlParamsBackUp 構造体でバックアップされ、LoadBackUp で再度有効になります。 微調整が正常に終了すると、コントローラが自動モードに切り替わります。 微調整が正常に行われなかった場合、モードへの切り替えは ActivateRecoverMode によって決定されます。 「微調整」のフェーズは、 PIDSelfTune.TIR.State で示されます。
3	**自動モード** 自動モードでは、PID_Temp は、指定されたパラメータに従って、コントロールされるシステムを訂正します。 以下の必要条件の 1 つが満たされた場合、コントローラは自動モードに切り替わります。 • プレチューニングの正常終了 • 微調整の正常終了 • ModeIN-OUT パラメータの値 3 への変更と ModeActivate の立ち上がりエッジ 手動モードから自動モードへの切り替えは、コミッショニングエディタで実行した場合にのみバンプレスです。 ActivateRecoverMode タグは、自動モードで考慮されます。
4	**手動モード** 手動モードでは、手動 PID 出力値を ManualValue パラメータに指定します。 この手動値により発生する加熱および冷却の出力値は、設定済み出力スケーリングの結果です。 この動作モードは、ManualEnable = TRUE を使用して有効にすることもできます。Mode および ModeActivate のみを使用して動作モードを変更することをお勧めします。 手動モードから自動モードへの切り替えはバンプレスです。 ActivateRecoverMode タグは、手動モードで考慮されます。
5	**エラーモニタリング付きの代替出力値** 制御アルゴリズムは無効です。 SetSubstituteOutput タグは、この動作モードでどの PID 出力値(PidOutputSum)を出力するかを決定します。 • SetSubstituteOutput = FALSE: 最後の有効な PID 出力値 • SetSubstituteOutput = TRUE: 代替出力値(SubstituteOutput) この動作モードは、Mode = 5 を使用して有効にすることはできません エラー発生時は、以下のすべての条件が満たされている場合、「無効」モードの代わりに有効になります。 • 自動モード(State = 3) • ActivateRecoverMode = TRUE • ActivateRecoverMode が有効になる 1 つまたは複数のエラーが発生しました。 保留中のエラーがなくなると、PID_Temp は直ちに自動モードに戻ります。

ENO 特性

State = 0 の場合、ENO = FALSE です。

State ≠ 0 の場合、ENO = TRUE です。

コミッショニング時の動作モードの自動切り替え

プレチューニングまたは微調整が正常に行われると、自動モードが有効になります。 下の表に、プレチューニングが正常に行われた場合に Mode および State がどのように切り替わるかを示します。

サイクル番号	Mode	State	操作
0	4	4	セット Mode = 1
1	1	4	セット ModeActivate = TRUE
1	4	1	State の値が Mode パラメータに保存されます。 プレチューニングが開始されます。
n	4	1	プレチューニングの正常終了
n	3	3	自動モードが開始されます。

PID_Temp は、エラー発生時に自動的に動作モードを切り替えます。

下の表に、プレチューニングでエラーが発生した場合 Mode および State がにどのように切り替わるかを示します。

サイクル番号	Mode	State	操作
0	4	4	セット Mode = 1
1	1	4	セット ModeActivate = TRUE
1	4	1	State の値が Mode パラメータに保存されます。 プレチューニングが開始されます。
n	4	1	プレチューニングがキャンセルされました
n	4	4	手動モードが開始されます。

ActivateRecoverMode = TRUE の場合、Mode パラメータに保存された動作モードが有効になります。 プレチューニングまたは微調整を開始するとき、PID_Temp は Mode 入出力パラメータに State の値を保存しています。 つまり、PID_Temp は調整の開始時にそのモードに切り替わります。

ActivateRecoverMode = FALSE の場合、システムは「無効」動作モードに切り替わります。

PID_Temp ErrorBits パラメータ

複数のエラーが同時に保留中の場合、ErrorBits の値がバイナリ加算で表示されます。たとえば、ErrorBits = 0000003h の表示はエラー 0000001h および 0000002h が同時に保留中であることを示します。

ErrorBits (DW#16#...)	説明
0000000	エラーはありません。
0000001	「Input」パラメータがプロセス値制限範囲外です。 • Input > Config.InputUpperLimit または • Input < Config.InputLowerLimit エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は自動モードのままです。 エラーが発生する前に手動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。
0000002	「Input_PER」パラメータの値が無効です。 アナログ入力でエラーが保留中であるかどうかをチェックします。 自動モードがエラー発生前に有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Temp は直ちに自動モードに戻ります。 エラーが発生する前に手動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。
0000004	微調整中のエラー。 プロセス値の振幅を保持できませんでした。 PID_Temp が加熱/冷却コントローラとして使用される場合(Config.ActivateCooling = TRUE)、セットポイントの PID 出力値 (PidOutputSum)は、実際の振動値を生成するため、加熱の微調整について正および冷却の微調整について負でなければなりません。 • • この要件が満たされる場合、調整オフセット(PIDSelfTune.TIR.OutputOffsetCool タグおよび PIDSelfTune.TIR.OutputOffsetHeat タグを)を使用してください。**微調整**を参照。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして Mode パラメータに保存されている動作モードに切り替わります。
0000008	プレチューニングの開始時のエラー。 プロセス値がセットポイントに近すぎるか、セットポイントよりも大きいです。 微調整を開始してください。

	エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
0000010	セットポイントが、調整時に変更されました。 CancelTuningLevel タグで、セットポイントの許容変動を設定できます。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
0000020	プレチューニングを微調整中に行うことはできません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は微調整モードのままです。
0000040	プレチューニング中のエラー。 冷却をプロセス値に縮小できませんでした。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
0000100	微調整時のエラーによって、パラメータが無効になりました。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
0000200	「Input」パラメータの値が無効です。 値が無効な番号書式です。 自動モードがエラー発生前に有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Temp は直ちに自動モードに戻ります。 エラーが発生する前に手動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。
0000400	出力値の計算が失敗しました。 PID パラメータをチェックしてください。 自動モードがエラー発生前に有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Temp は直ちに自動モードに戻ります。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。
0000800	サンプリング時間エラー: PID_Temp が、サイクル割り込み OB のサンプリング時間内に呼び出されていません。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は自動モードのままです。 エラーが発生する前に手動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。
0001000	「Setpoint」パラメータまたは「SubstituteSetpoint」の値が無効です。 値が無効な番号書式です。 自動モードがエラー発生前に有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は設定された代替出力値を出力します。 保留中のエラーがなくなると、PID_Temp は直ちに自動モードに戻ります。

	エラーが発生する前に手動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。
0010000	「ManualValue」パラメータの値が無効です。 値が無効な番号書式です。 エラーが発生する前に ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままで、 PID 出力値として SubstituteOutput を使用します。 ManualValue で有効な値を指定するとすぐに、PID_Temp はそれを PID 出力値として使用します。
0020000	SubstituteOutput タグの値が無効です。 値が無効な番号書式です。 PID_Temp は「エラーモニタリング付き代替出力値」モードまたは手動モードのままで、PID 出力値として、加熱の PID 出力値の下限値を使用します(Config.Output.Heat.PidLowerLimit)。 SubstituteOutput で有効な値を指定するとすぐに、PID_Temp はそれを PID 出力値として使用します。
0040000	「Disturbance」パラメータの値が無効です。 値が無効な番号書式です。 エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、Disturbance はゼロに設定されます。 PID_Temp は自動モードのままです。 プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。 現在のフェーズの Disturbance が出力値に影響しない場合、調整はキャンセルされません。
0200000	カスケードでの Master エラー: Slaves が自動モードでないか、代替セットポイントが有効でマスタの調整を防止しています。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
0400000	加熱の微調整は、冷却が有効な間は許可されません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
0800000	プロセス値は、冷却のプレチューニングを開始するためセットポイントに近い値でなければなりません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
1000000	調整の開始時のエラー: Heat.EnableTuning および Cool.EnableTuning が設定されていないか、設定に一致しません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
2000000	冷房のプレチューニングは、加熱のプレチューニングの成功を必要とします。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
4000000	微調整の開始時のエラー: Heat.EnableTuning および Cool.EnableTuning を同時に設定することはできません。 エラー発生前に ActivateRecoverMode = TRUE の場合、PID_Temp は調整をキャンセルして、Mode パラメータに保存されている動作モードに切り替わります。
8000000	PID パラメータの計算中のエラーにより、パラメータが無効になりました。

	無効なパラメータは破棄され、元の PID パラメータが変更なしで保持されます。 以下の場合を区別できます。 • エラーが発生する前に自動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は自動モードのままです。 • エラーが発生する前に手動モードが有効になっていて ActivateRecoverMode = TRUE の場合、PID_Temp は手動モードのままです。 • プレチューニングモードまたは微調整モードがエラー発生前に有効になっていて、ActivateRecoverMode = TRUE の場合、PID_Temp は Mode パラメータに保存されている動作モードに切り替わります。

PID_Temp ActivateRecoverMode タグ

ActivateRecoverMode タグは、エラーに対する応答を決定します。 Error パラメータは、エラーが保留中かどうかを示します。 エラーが保留中でない場合、Error = FALSE となります。 ErrorBits パラメータは、どのエラーが発生したかを示します。

自動モードおよび手動モード

通知
システムが損傷することがあります。 ActivateRecoverMode = TRUE の場合、エラーが発生していてプロセス限界値を超えていても PID_Temp は自動モードまたは手動モードのままです。 そのため、システムが損傷することがあります。 システムを損傷から守るために、エラーの場合に制御システムがどのように応答するかを設定する必要があります。

ActivateRecoverMode	説明
FALSE	PID_Temp は、エラー発生時に「無効」モードに切り替わります。 コントローラは、Reset の立ち下りエッジまたは ModeActivate の立ち上がりエッジでのみ有効になります。
TRUE	**自動モード** 自動モードでエラーが頻繁に発生する場合、この設定は制御応答にマイナスの影響を及ぼします。エラーが発生するたびに、PID_Temp が計算された PID 出力値と代替出力値の間で切り替わるためです。 この場合は、ErrorBits パラメータをチェックして、エラーの原因を取り除いてください。 1 つまたは複数の以下のエラーが発生し、エラーが発生する前に自動モードが有効になっていた場合、PID_Temp は自動モードのままです。 • 0000001h: 「Input」パラメータがプロセス値制限範囲外です。 • 0000800h: サンプリング時間のエラー • 0040000h: 「Disturbance」パラメータの値が無効です。 • 8000000h: PID パラメータの計算中エラー 1 つまたは複数の以下のエラーが発生し、エラーが発生する前に自動モードが有効になっていた場合、PID_Temp は「エラーモニタリング付き代替出力値」モードに切り替わります。 • 0000002h: 「Input_PER」パラメータの値が無効です。 • 0000200h: 「Input」パラメータの値が無効です。 • 0000400h: 出力値の計算が失敗しました。 • 0001000h: 「Setpoint 」パラメータまたは「SubstituteSetpoint」の値が無効です。 保留中のエラーがなくなると、PID_Temp は直ちに自動モードに戻ります。 「エラーモニタリング付き代替出力値」モードで以下のエラーが発生すると、PID_Temp は、エラーが保留中である限り、PID 出力値を Config.Output.Heat.PidLowerLimit に設定します。

	• 0020000h: SubstituteOutput タグの値が無効です。値が無効な番号書式です。 この動作は SetSubstituteOutput と無関係です。
	手動モード 1 つまたは複数のエラーが発生し、エラーが発生する前に手動モードが有効になっていた場合、PID_Temp は手動モードのままです。 手動モードで以下のエラーが発生し、エラーが保留中である限り、PID_Temp は PID 出力値を SubstituteOutput:に設定します。 • 0010000h:「ManualValue」パラメータの値が無効です。 値が無効な番号書式です。 手動モードでエラー 0010000h が保留中で、以下のエラーが発生すると、このエラーが保留中である限り、PID_Temp は PID 出力値を Config.Output.Heat.PidLowerLimit に設定します。 • 0020000h:SubstituteOutput タグの値が無効です。値が無効な番号書式です。 この動作は SetSubstituteOutput と無関係です。

プレチューニングと微調整

ActivateRecoverMode	説明
FALSE	PID_Temp は、エラー発生時に「無効」モードに切り替わります。コントローラは、Reset の立ち下りエッジまたは ModeActivate の立ち上がりエッジでのみ有効になります。
TRUE	以下のエラーが発生した場合、PID_Temp はアクティブモードのままです。 • 0000020h: プレチューニングを微調整中に行うことはできません。 以下のエラーは無視されます。 • 0010000h: 「ManualValue」パラメータの値が無効です。 • 0020000h:SubstituteOutput タグの値が無効です。 これ以外のエラーが発生した場合、PID_Temp は調整をキャンセルして調整が開始されたモードに切り替わります。

PID_Temp 警告タグ

複数の警告が同時に保留中の場合、Warning タグの値がバイナリ加算で表示されます。 警告 0000003h の表示は、たとえば、警告 0000001h と 0000002h が同時に保留中であることを示します。

Warning (DW#16#....)	説明
0000000	保留中の警告がありません。
0000001	プレチューニング時に変曲点が見つかりませんでした。
0000004	セットポイントが、設定済みの限界値に制限されました。
0000008	選択された計算方法で、コントロールされるシステムの必要なプロパティの中に定義されなかったものがあります。 代わりに、PID パラメータが TIR.TuneRuleHeat メソッドまたは TIR.TuneRuleCool = 3 によって計算されました。
0000010	Reset = TRUE または ManualEnable = TRUE.のために、動作モードを変更できませんでした。
0000020	呼び出し OB のサイクルタイムが、PID アルゴリズムのサンプリング時間を制限します。 さらに短い OB サイクルタイムを使用して、結果を改善してください。
0000040	プロセス値が、その警告限界値の 1 つを超えました。
0000080	Mode での値が不正です。 動作モードは変更されません。
0000100	手動値が、PID 出力値の限界値に制限されました。
0000200	調整の指定されたルールはサポートされていません。 PID パラメータは計算されません。
0001000	代替出力値が出力限界値を超えているため、代替出力値に達することができません。
0004000	加熱/冷却の出力値の指定された数はサポートされていません。 出力 OutputHeat または OutputCool のみが使用されます。
0008000	PIDSelfTune.SUT.AdaptDelayTime での値が不正です。 既定値 0 が使用されます。
0010000	PIDSelfTune.SUT.CoolingMode での値が不正です。 既定値 0 が使用されます。
0020000	冷却の有効化(Config.ActivateCooling タグ)は、マスタとして使用されるコントローラではサポートされていません(Config.Cascade.IsMaster タグ)。 PID_Temp は加熱コントローラとして作動します。 Config.ActivateCooling タグを FALSE に設定します。
0040000	Retain.CtrlParams.Heat.Gain、Retain.CtrlParams.Cool.Gain または Config.CoolFactor.の値が無効です。 PID_Temp は、比例ゲイン(加熱および冷却)と冷却係数の正の値のみをサポートしています。 自動モードは、PID 出力値が 0.0 のとき有効なままです。 積分要素は停止します。

以下の警告は、原因の対策が取られるか、または有効なパラメータによって操作を繰り返すと、直ちに削除されます。

- 0000001H
- 0000004h
- 0000008h

- 0000040h
- 0000100h

他のすべての警告は、Reset または ErrorAck の立ち上がり エッジで解除されます。

PwmPeriode タグ

OutputHeat_PWM または OutputCool_PWM を使用するとき、PID アルゴリズムサンプリング時間(Retain.CtrlParams.Heat.Cycle または Retain.CtrlParams.Heat.Cycle)、つまりパルス幅変調の時間が非常に長い場合、Config.Output.Heat.PwmPeriode または Config.Output.Cool.PwmPeriode パラメータで時間のずれを短く設定して、プロセス値の滑らかさを向上させることができます。

OutputHeat_PWM でのパルス幅変調の時間

Config.Output.Heat.PwmPeriode に依存する出力 OutputHeat_PWM の PWM の時間:

- Heat.PwmPeriode = 0.0 (既定)

  加熱の PID アルゴリズムのサンプリング時間(Retain.CtrlParams.Heat.Cycle)が PWM の時間として使用されます。

- Heat.PwmPeriode > 0.0

  値は PID_Temp サンプリング時間の整数倍に四捨五入され(CycleTime.Value)、PWM の時間として使用されます。

  値は、以下の条件を満たす必要があります。

  - Heat.PwmPeriode ≤ Retain.CtrlParams.Heat.Cycle
  - Heat.PwmPeriode > Config.Output.Heat.MinimumOnTime
  - Heat.PwmPeriode > Config.Output.Heat.MinimumOffTime

OutputCool_PWM でのパルス幅変調の時間

Config.Output.Cool.PwmPeriode および加熱/冷却の方法に依存する出力 OutputCool_PWM の PWM の時間:

- Cool.PwmPeriode = 0.0 で、かつ冷却係数 (Config.AdvancedCooling = FALSE):

  加熱の PID アルゴリズムのサンプリング時間(Retain.CtrlParams.Heat.Cycle)が PWM の時間として使用されます。

- Cool.PwmPeriode = 0.0 で、かつ PID パラメータの切り替え(Config.AdvancedCooling = TRUE):

  冷却の PID アルゴリズムのサンプリング時間(Retain.CtrlParams.Cool.Cycle)が PWM の時間として使用されます。

- Cool.PwmPeriode > 0.0:

  値は PID_Temp サンプリング時間の整数倍に四捨五入され(CycleTime.Value)、PWM の時間として使用されます。

  値は、以下の条件を満たす必要があります。

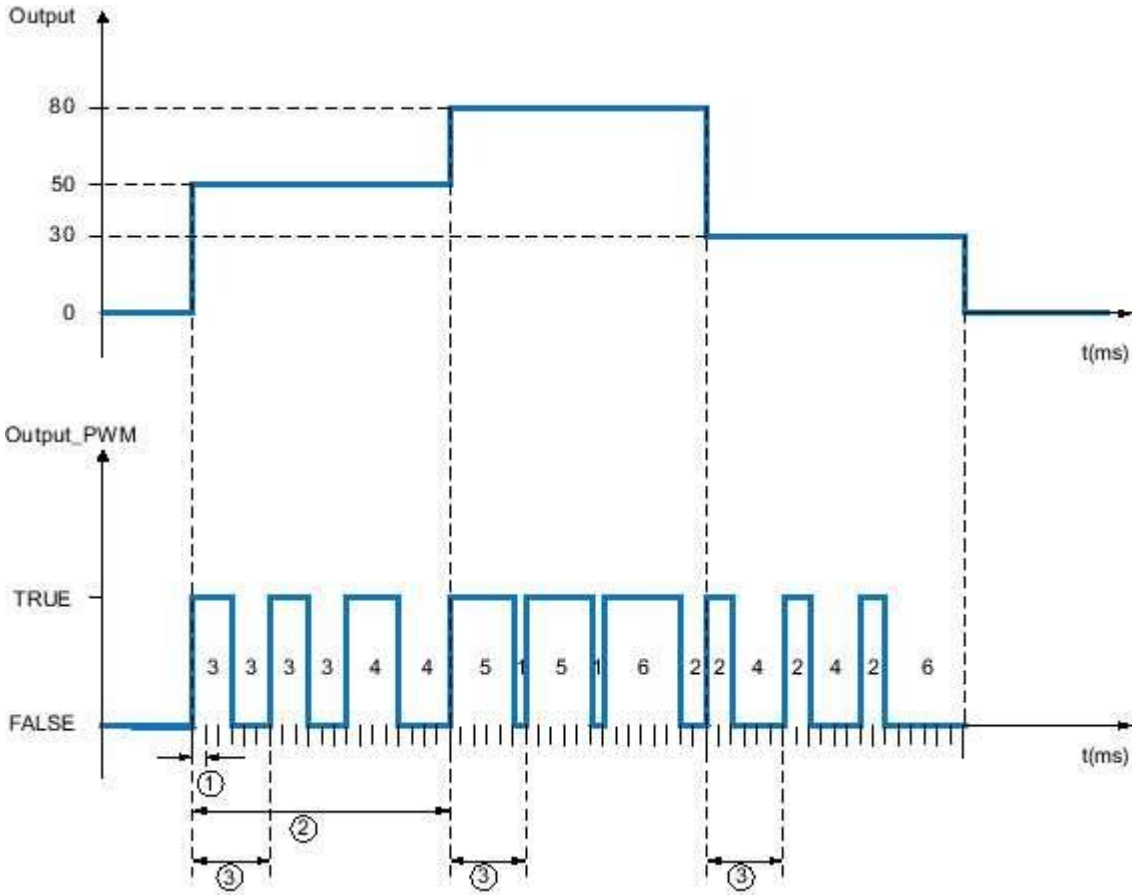
  - Cool.PwmPeriode ≤ Retain.CtrlParams.Cool.Cycle または Retain.CtrlParams.Heat.Cycle
  - Cool.PwmPeriode > Config.Output.Cool.MinimumOnTime
  - Cool.PwmPeriode > Config.Output.Cool.MinimumOffTime

Config.Output.Cool.PwmPeriode は、冷却出力が有効な場合のみ有効です(Config.ActivateCooling =TRUE)。

PwmPeriode を使用するとき、PWM 出力信号の精度は、PID_Temp サンプリング時間(OB のサイクルタイム)に対する PwmPeriode の関係によって決まります。PwmPeriode は、PID_Temp サンプリング時間の 10 倍以上でなければなりません。

PID アルゴリズムのサンプリング時間が PwmPeriode の整数倍でない場合、PID アルゴリズムのサンプリング時間内で PWM の最後の時間ごとに適切に延長されます。

OutputHeat_PWM の例



① PID_Temp サンプリング時間 = 100.0 ms (呼び出し側のサイクリック割り込み OB のサイクルタイム、CycleTime.Value タグ)

② PID アルゴリズムのサンプリング時間 = 2000.0 ms (Retain.CtrlParams.Heat.Cycle タグ)

③ 加熱の PWM の時間 = 600.0 ms (Config.Output.Heat.PwmPeriode タグ)

PID 基本ファンクション

この章には下記に関する情報が記載されています：

- CONT_C S7-300/400 と比較した場合の相違点 (S7-1500)
- CONT_S S7-300/400 と比較した場合の相違点 (S7-1500)
- PULSEGEN S7-300/400 と比較した場合の相違点 (S7-1500)
- TCONT_CP S7-300/400 と比較した場合の相違点 (S7-1500)
- TCONT_S S7-300/400 と比較した場合の相違点 (S7-1500)

CONT_C S7-300/400 と比較した場合の相違点

CONT_C 命令が最適化したブロックアクセスで作成されます。

すべてのパラメータが保持されます。 この保持性は変更できません。 保持タグの現在値は、CONT_C 全体をダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

それ以外は、S7-1500 CPU での CONT_C の動作は S7-300、S7-400 CPU の場合と全く同じです。

CONT_C の説明

CONT_C の動作

CONT_C ブロックダイアグラム

CONT_C の入力パラメータ

CONT_C の出力パラメータ

CONT_S S7-300/400 と比較した場合の相違点

CONT_S 命令が最適化したブロックアクセスで作成されます。S

すべてのパラメータが保持されます。 この保持性は変更できません。 保持タグの現在値は、CONT_S 全体をダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

それ以外は、S7-1500 CPU での CONT_S の動作は S7-300、S7-400 CPU の場合と全く同じです。

CONT_S の説明

CONT_S の動作モード

CONT_S ブロックダイアグラム

CONT_S の入力パラメータ

CONT_S の出力パラメータ

PULSEGEN S7-300/400 と比較した場合の相違点

PULSEGEN 命令が最適化したブロックアクセスで作成されます。

すべてのパラメータが保持されます。 この保持性は変更できません。 保持タグの現在値は、PULSEGEN 全体をダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

S7-1500 CPU での PULSEGEN の動作は S7-300、S7-400 CPU の場合と全く同じです。

PULSEGEN の説明

PULSEGEN の動作モード

PULSEGEN の動作モード

3 ステップコントロール

2 ステップコントロール

PULSEGEN の入力パラメータ

PULSEGEN の出力パラメータ

TCONT_CP S7-300/400 と比較した場合の相違点

TCONT_CP 命令が最適化したブロックアクセスで作成されます。

すべてのパラメータが保持されます。 この保持性は変更できません。 保持タグの現在値は、TCONT_CP 全体をダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

それ以外は、S7-1500 CPU での TCONT_CP の動作は S7-300、S7-400 CPU の場合と全く同じです。

TCONT_CP の説明

TCONT_CP の動作モード

パルスジェネレータの動作原理

TCONT_CP のブロックダイアグラム

TCONT_CP の入力パラメータ

TCONT_CP の出力パラメータ

TCONT_CP の I/O パラメータ

TCONT_CP の静的変数

パラメータ STATUS_H

パラメータ STATUS_D

TCONT_S S7-300/400 と比較した場合の相違点

TCONT_S 命令が最適化したブロックアクセスで作成されます。

すべてのパラメータが保持されます。 この保持性は変更できません。 保持タグの現在値は、TCONT_S 全体をダウンロードしたときにのみ更新されます。

テクノロジーオブジェクトのデバイスへのダウンロード

それ以外は、S7-1500 CPU での TCONT_S の動作は S7-300、S7-400 CPU の場合と全く同じです。

TCONT_S の説明

TCONT_S の動作モード

TCONT_S のブロックダイアグラム

TCONT_S の入力パラメータ

TCONT_S の出力パラメータ

TCONT_S の I/O パラメータ

TCONT_S の静的変数

Time-based IO

この章には下記に関する情報が記載されています：

- Time-based IO (S7-1500)

Time-based IO

この章には下記に関する情報が記載されています：

- TIO_SYNC: TIO モジュールを同期 (S7-1500)
- TIO_IOLink_IN: タイムスタンプによる入力信号の読み出しプロセス (S7-1500)
- TIO_DI: デジタル入力のエッジおよび関連するタイムスタンプを読み出す (S7-1500)
- TIO_IOLink_OUT: 出力時刻制御されたプロセス出力信号 (S7-1500)
- TIO_DQ: 出力エッジがデジタル出力で時刻制御されている (S7-1500)
- UDT "TIO_SYNC_Data" (S7-1500)

TIO_SYNC: TIO モジュールを同期

説明

TIO_SYNC 命令と一緒に Time-based IO を使用できます。TIO_SYNC は、共有時間ベース TIO_Time に TIO モジュールを同期します。

最大 8 つの TIO モジュール with TIO_SYNC を同期できます。 すべての TIO モジュールは、同じプロセスイメージパーティションに割り当てる必要があります(PIP)。 入力パラメータ PIP_Mode に「0」を選択する場合、PIP_PART 入力パラメータに、プロセスイメージパーティションの数を割り当てます。

スタートアップ特性

CPU のスタートアップ時に、TIO_SYNC 命令は入力パラメータを 1 回受信およびチェックし、TIO_Time を初期化します。 エラーなしでスタートアップが行われると、命令は通常の操作に切り替わります。 エラーが発生した場合、命令は通常の操作に切り替わらず、エラーメッセージを生成します。

動作原理

通常の操作で、命令は HWID 入力で設定されたすべての TIO モジュールを確実に同期します。

TIO モジュールの命令に計算された TIO_Time は、TIO_SYNC_Data 出力で提供されます。

エラーに対する応答

Error 出力は、命令パラメータが正しく割り当てられたかどうか、エラーが発生したかどうかを示します。 命令が正常に処理されなかった場合、エラーの理由が Status 出力に表示されます。

パラメータ

次の表に、TIO_SYNC 命令のパラメータを示します。

パラメータ	宣言	データタイプ S7-1500	既定	説明
HWID_1 ... HWID_8*	入力	HW_IO	-1	HWCN からの TIO モジュールのハードウェア識別子
Send-Clock*	入力	LTime	LT#0ns	同期ドメインの送信クロック。 PROFINET 設定から送信クロックを適用します。
PIP_Mode*	入力	USInt	0	データ更新のモード**: • SYNC_PO および SYNC_PI によるプロセスイメージの内部更新のある 0 : OIP モデル • プロセスイメージの内部更新のない 1: OIP モデル • プロセスイメージの内部更新のない 2: IPO モデル
PIP_PART*	入力	USInt	1	PIP_Mode = 0 の場合にのみ関連

				アイソクロナス方式で更新するプロセスイメージパーティションの番号。
TIO_SYNC_Data	出力	"TIO_SYNC_Data"		同期に使用する TIO モジュールおよび内部データの TIO 命令に計算されたシステム時間。 UDT "TIO_SYNC_Data"を参照してください。
Status	出力	DWord	16#0	命令のステータス Status パラメータの説明を参照してください。
Error	出力	Bool	False	True の場合、この出力はエラーが発生したことを示します。 詳細は Status パラメータを参照してください。 Error は、エラーの対策が取られると直ちにリセットされます。
* CPU のスタートアップ時に 1 回チェックされます ** IPO モデル(PIP_Mode = 2)は応答時間が最も短いですが、システム性能に最も大きな負荷もかけます。 すべての TIO 命令およびその他のプログラムセクションの処理は、1 つの送信クロック内で完了する必要があります。				

ステータスパラメータ

エラーコードまたはステータス情報は、ダブルワードとして Status 出力で出力されます。

ダブルワードは以下のように分割されます。

エラーコードのセグメント	意味
z0yywwww	システムファンクションのエラー: • スタートアップ中(z = 0) • 通常の操作中(z = 1) 下位使用のあるシステムファンクションは yy でコーディングされます。 エラーコードの表を参照してください。 wwww はシステムファンクションの RET_VAL を指定します。 エラー情報については、システムファンクションのヘルプを参照してください。
z0yy0000	システムファンクションで発生していないエラー。 このエラーは連続したエラー番号 yy を受信します。 エラーは以下の場合に発生する可能性があります。 • スタートアップ中(z = 0) • 通常の操作中(z = 1)

エラーコードの表

エラーコード (DW#16#...)	意味	対処法
00000000	エラーは発生していません。	—

1001xxxx	エラーがシステムファンクション RD_SINFO で発生しました。 下位ワード xxxx は、RD_SINFO からの RET_VAL 戻り値のエラー情報を示します。	• TIA Portal 情報システムの RD_SINFO の説明をお読みください。 • TIO_SYNC を必ず「Synchronous Cycle」OB で呼び出してください。
10020000	アイソクロナス OB の読み取りサイクルタイムは LT#0ms または負の値であるため、無効です。 命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	• サイクルタイムの値を訂正します。 • TIO_SYNC を必ず「Synchronous Cycle」OB で呼び出してください。
10030000	TIO_SYNC 命令は、アイソクロナス OB で呼び出されません。この命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。 考えられる原因: The TIO_SYNC 命令が"MC-Servo"または"MC-Interpolator" OB.で呼び出された。	TIO_SYNC を必ず「Synchronous Cycle」OB で呼び出してください。
10040000	PIP_Mode 入力パラメータに割り当てられた値が、有効な 0～2 の範囲外である。	PIP_Mode 入力パラメータの値を訂正します。
10050000	割り当てられた送信クロックが許可されている 0 < SendClock <= 4 ms の範囲外であるため無効である。	送信クロックの値を訂正します。
0006xxxx	エラーが SYNC_PI システムファンクションの実行中に発生しました。 下位ワード xxxx は、SYNC_PI からの RET_VAL 戻り値のエラー情報を示します。	TIA Portal 情報システムの SYNC_PI の説明をお読みください。
0007xxxx	エラーが SYNC_PO システムファンクションの実行中に発生しました。 下位ワード xxxx は、SYNC_PO からの RET_VAL 戻り値のエラー情報を示します。	TIA Portal 情報システムの SYNC_PO の説明をお読みください。
10080000	アイソクロナスモードが見つかりません。Time-based IO を使用できません。 考えられる原因: アドレス計算中の内部エラー	入力パラメータ HWID_1～HWID_8 の値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。
10090000	TIO モジュール HWID_1: HWID_1 の読み出し中のエラー。	個々の入力パラメータの値をチェックします。 それぞれの場合で TIO モジュールのハードウェア識別子を HWCN のモジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が､シンボリックアドレス指定で使用可能です。
100A0000	TIO モジュール HWID_2: HWID_2 の読み出し中のエラー。	
100B0000	TIO モジュール HWID_3: HWID_3 の読み出し中のエラー。	
100C0000	TIO モジュール HWID_4: HWID_4 の読み出し中のエラー。	
100D0000	TIO モジュール HWID_5: HWID_5 の読み出し中のエラー。	
100E0000	TIO モジュール HWID_6: HWID_6 の読み出し中のエラー。	
100F0000	TIO モジュール HWID_7: HWID_7 の読み出し中のエラー。	
10100000	TIO モジュール HWID_8: HWID_8 の読み出し中のエラー。	

10110000	入力パラメータ PIP_PART に設定したプロセスイメージパーティションが、有効な範囲 1～31 内にありません。	PIP_PART 入力パラメータの値を訂正します。
x0FF0000	一般的な内部エラー。	—

TIO_IOLink_IN: タイムスタンプによる入力信号の読み出しプロセス

説明

TIO_IOLink_IN 命令で、 Time-based IO を使用できます。TIO_IOLink_IN は IO-Link Device でイベントを検出し、関連するタイムスタンプを含めてプロセス値を返します。

IO-Link Device はタイムスタンプファンクションを備えている必要があり、Port は「IO-Link, Time based IN」モードにある必要があります。

スタートアップ特性

CPU のスタートアップ中、命令 TIO_IOLink_IN は入力パラメータを 1 回適用し、以下をチェックします。

- HWID のチェック
- Port 番号が範囲内(1～4)にあるかをチェック
- TIO_SYNC_Data.Error のチェック: TIO_SYNC にエラーが存在するか?
- 正の値の T_o チェック
- タイプ IO-Link への設定のチェック
- IO-Link Time based IN の設定に関する Port モードのチェック
- OB が「Synchronous Cycle」OB であるかをチェック
- PortQualifier のチェック

エラーなしでスタートアップが行われると、命令は通常の操作に切り替わります。 エラーが発生した場合、命令は通常の操作に切り替わらず、エラーメッセージを生成します。

動作原理

通常の操作で、命令は IO-Link Device および最後の有効な変更の関連するタイムスタンプ(時刻 = TIO_Time)のプロセスデータ(SA_Data)を検出します。 SA_Data を正しく動作するポートで正しく変更するたびに、有効なタイムスタンプが付きます。

入力 TIO_SYNC_Data を TIO_SYNC 命令の同じ名前の出力に接続します。 これにより、共有時間ベースを確保します。

エラーに対する応答

Error 出力は、命令が正しく処理されたかどうかを示します。 エラーの原因は Status 出力に表示されます。

パラメータ

次の表に、TIO_IOLink_IN 命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定	説明
		S7-1500		
HWID*	入力	HW_IO	0	HWCN からの TIO モジュールのハードウェア識別子

Port*	入力	USInt	0	Port 接続済みの番号(1～4)IO-Link Device
TIO_SYNC_Data*	入力	"TIO_SYNC_Data"		TIO_SYNC 命令によって TIO モジュールの TIO 命令に提供されるシステム時刻。 UDT "TIO_SYNC_Data"を参照してください。 この入力パラメータを TIO_SYNC 命令の「TIO_SYNC_Data」出力パラメータに接続します。
TO*	入力	LTime	LT#0ns	T_O: アイソクロナス出力データの出力時刻。 T_O を ET 200 ステーション(PROFINET インターフェースのプロパティ)から適用します。
SA_Bit0	出力	Bool	False	プロセスデータの変更を表示します(SA_Data のビット 0)。 SA_Bit0 はイベント指向です。 有効なタイムスタンプが取得されない限り(EventCount = 0)、プロセスデータは無効です。
SA_Bit1	出力	Bool	False	プロセスデータの変更を表示します(SA_Data のビット 1)。 SA_Bit1 はイベント指向です。 有効なタイムスタンプが取得されない限り(EventCount = 0)、プロセスデータは無効です。
SA_Data	出力	DWord	16#0	プロセスデータ SA_Data (センサアプリケーションデータ)
Time-Stamp	出力	LTime	LT#0ns	タイムスタンプ: プロセス信号の変更 SA-Data が IO-Link Device で発生した時刻。
Even-tCount	出力	UInt	0	カウンタ: 新しい、有効なタイムスタンプごとにインクリメントします。 カウンタは、CPU がスタートアップするたびにリセットされます。
Status	出力	DWord	16#0	命令のステータス Status パラメータの説明を参照してください。
Error	出力	Bool	False	True の場合、この出力はエラーが発生したことを示します。 詳細は Status パラメータを参照してください。 Error は、エラーの対策が取られると直ちにリセットされます。
* CPU のスタートアップ時に 1 回チェックされます				

ステータスパラメータ

エラーコードまたはステータス情報は、ダブルワードとしてステータス出力で出力されます。

ダブルワードは以下のように分割されます。

エラーコードのセグメント	意味
z0yywwww	システムファンクションのエラー: • スタートアップ中(z = 0) • 通常の操作中(z = 1)

	下位使用のあるシステムファンクションは yy でコーディングされます。 エラーコードの表を参照してください。 wwww はシステムファンクションの RET_VAL を指定します。 システムファンクションについては、システムファンクションのヘルプを参照してください。
z0yy0000	システムファンクションで発生していないエラー。 このエラーは連続したエラー番号 yy を受信します。 エラーは以下の場合に発生する可能性があります。 • スタートアップ中(z = 0) • 通常の操作中(z = 1)

エラーコードの表

エラーコード (DW#16#...)	意味	対処法
00000000	エラーは発生していません。	—
10010000	Port 入力パラメータに割り当てられた番号が、有効な 1～4 の範囲外である。	Port 入力パラメータの値を訂正します。
10020000	TIO_IOLink_IN 命令は、アイソクロナス OB で呼び出されません。この命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	TIO_IOLink_IN を必ず「Synchronous Cycle」OB で呼び出してください。
10030000	HWID 入力パラメータの読み出し時にエラーが発生しました。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が、シンボリックアドレス指定で使用可能です。
x0040000	TIO_SYNC_Data のデータが無効/不正です。	TIO_SYNC 命令およびその TIO_SYNC_Data 出力との相互接続をチェックします。
1005xxxx	エラーが RD_SINFO システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_SINFO からの RET_VAL 戻り値のエラー情報を示します。	• TIA Portal 情報システムの RD_SINFO の説明をお読みください。 • TIO_IOLink_IN を必ず「Synchronous Cycle」OB で呼び出してください。
10060000	IO-Link Device が見つかりません。 考えられる原因: ハードウェア識別子を使用して構成されたモジュールが、Time-based IO 用の IO-Link Master ではない。	• 構成されたモジュールが、Time-based IO 用の IO-Link Master であることを確認します。 • HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。
10070000	アドレス計算中に発生した内部エラー。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数

		が、シンボリックアドレス指定で使用可能です。
00080000	TIO モジュールが、TIO_SYNC 命令を使用して同期されていません。	命令 TIO_SYNC.をチェックします。
10090000	アイソクロナス OB の読み出しサイクルタイムが許可されている 0 < T_{APP} <= 16 ms 範囲外であるため、無効です。 命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	• サイクルタイムを訂正します。 • TIO_IOLink_IN を必ず「Synchronous Cycle」OB で呼び出してください。
100A0000	TO 入力パラメータに割り当てられた時間が、許可されている 0 < T_o <= 4 ms 範囲外です。	TO 入力パラメータの値を訂正します。
100Bxxxx	エラーが RD_ADDR システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_ADDR からの RET_VAL 戻り値のエラー情報を示します。	TIA Portal 情報システムの RD_ADDR の説明をお読みください。
000C0000	変換されたタイムスタンプが無効です。	接続されたセンサおよび IO-Link Master とセンサ間の相互作用をチェックします(設定など)。
000D0000	IO-Link の値のステータス PortQualifier が、プロセスデータが無効であることを示しています。	接続されているセンサとその設定をチェックします。
100E0000	IO-Link の設定されたポートモードが不正です。	S7-PCT を使用して、接続されたセンサの設定をチェックします。
x0FF0000	一般的な内部エラー。	—

TIO_DI: デジタル入力のエッジおよび関連するタイムスタンプを読み出す

説明

Time-based IO を TIO_DI 命令と一緒に使用できます。TIO_DI は、TM Timer DIDQ のデジタル入力のエッジを継続的に検出し、関連するタイムスタンプを返します。

スタートアップ特性

CPU のスタートアップ中、命令 TIO_DI は入力パラメータを 1 回適用し、以下をチェックします。

- HWID のチェック
- デジタル入力の数(Channel)が許可されている範囲であるかどうかをチェック(アドレス指定されたモジュールとチャンネル構成に依存する)
- TIO_SYNC_Data.Error のチェック: TIO_SYNC にエラーが存在するか?
- T_o 有効期限のチェック(0 ms ~ 4 ms)
- OB が「Synchronous Cycle」OB であるかをチェック

エラーなしでスタートアップが行われると、命令は通常の操作に切り替わります。 入力パラメータ EdgeSel は、通常の操作中に変更できます。 エラーが発生した場合、命令は通常の操作に切り替わらず、エラーメッセージを生成します。

動作原理

通常の操作で、命令は、最後の有効な、定義されたエッジペアのデジタル入力のエッジと関連するタイムスタンプを検出します(時刻 = TIO_Time)。 入力パラメータ EdgeSel を使用して、タイムスタンプが検出されるエッジを決定します。

入力 TIO_SYNC_Data を TIO_SYNC 命令の同じ名前の出力に接続します。 これにより、共有時間ベースを確保します。

エラーに対する応答

Error 出力は、命令が正しく処理されたかどうかを示します。 エラーの原因は Status 出力に表示されます。

パラメータ

次の表に、TIO_DI 命令のパラメータを示します。

パラメータ	宣言	データタイプ	既定	説明
		S7-1500		
HWID*	入力	HW_IO	0	HWCN からの TIO モジュールのハードウェア識別子
Channel*	入力	UInt	0	接続される TM Timer DIDQ のデジタル入力の数 (0 ... m)
TIO_SYNC_Data*	入力	"TIO_SYNC_Data"		TIO_SYNC 命令によって TIO モジュールの TIO 命令に提供されるシステム時刻。 UDT "TIO_SYNC_Data"を参照してください。

				この入力パラメータを TIO_SYNC 命令の「TIO_SYNC_Data」出力パラメータに接続します。
EdgeSel	入力	UInt	3	タイムスタンプが検出されるエッジを指定します。 0_D: 予備 1_D: 2 つの立ち上がりエッジ 2_D: 2 つの立ち下がりエッジ 3_D: 立ち上がりエッジと立ち下がりエッジ(発生による順序) 4_D: 最初の立ち上がりエッジ、その後立ち下がりエッジ 5_D: 最初の立ち下がりエッジ、その後立ち上がりエッジ 6 bis 255_D: 予備
TO*	入力	LTime	LT#0ns	T_o: アイソクロナス出力データの出力時刻。 T_o を ET 200 ステーション (PROFINET インターフェースのプロパティ)から適用します。
DI	出力	Bool	False	デジタル入力のステータス。 デジタル入力の反転が設定されている場合、このパラメータも反転されます。
Time-StampRE	出力	LTime	LT#0ns	タイムスタンプ: 立ち上がりエッジが検出された時刻。 例外: EdgeSel = 2_D: 立ち下がりエッジが検出された時刻。
Time-StampFE	出力	LTime	LT#0ns	タイムスタンプ: 立ち下がりエッジが検出された時刻。 例外: EdgeSel = 1_D: 立ち上がりエッジが検出された時刻。
Even-tCountRE	出力	UInt	0	カウンタ: 立ち上がりエッジの新しい、有効なタイムスタンプごとにインクリメントします。 カウンタは、CPU がスタートアップするたびにリセットされます。
Even-tCountFE	出力	UInt	0	カウンタ: 立ち下がりエッジの新しい、有効なタイムスタンプごとにインクリメントします。 カウンタは、CPU がスタートアップするたびにリセットされます。
LEC	出力	UInt	0	カウンタ: タイムスタンプを保存できなかったエッジの数。 モジュールは、アプリケーションサイクル当たり最大 7 つのエッジをカウントできます。 カウンタは、新しいアプリケーションサイクルごとにリセットされます。
Status	出力	DWord	16#0	命令のステータス Status パラメータの説明を参照してください。

Error	出力	Bool	False	True の場合、この出力はエラーが発生したことを示します。 詳細は Status パラメータを参照してください。 Error は、エラーの対策が取られると直ちにリセットされます。
* CPU のスタートアップ時に 1 回チェックされます				

ステータスパラメータ

エラーコードまたはステータス情報は、ダブルワードとしてステータス出力で出力されます。

ダブルワードは以下のように分割されます。

エラーコードのセグメント	意味
z0yywwww	システムファンクションのエラー: • スタートアップ中(z = 0) • 通常の操作中(z = 1) 下位使用のあるシステムファンクションは yy でコーディングされます。 エラーコードの表を参照してください。 wwww はシステムファンクションの RET_VAL を指定します。 システムファンクションについては、システムファンクションのヘルプを参照してください。
z0yy0000	システムファンクションで発生していないエラー。 このエラーは連続したエラー番号 yy を受信します。 エラーは以下の場合に発生する可能性があります。 • スタートアップ中(z = 0) • 通常の操作中(z = 1)

エラーコードの表

エラーコード (DW#16#...)	意味	対処法
00000000	エラーは発生していません。	—
10010000	Channel 入力パラメータに割り当てられたデジタル入力の数が、許可されている範囲外です (アドレス指定されたモジュールとチャンネル構成に依存する)。	Channel 入力パラメータの値を訂正します。
10020000	TIO_DI 命令は、アイソクロナス OB で呼び出されません。この命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	TIO_DI を必ず「Synchronous Cycle」OB で呼び出してください。
10030000	HWID 入力パラメータの読み出し時にエラーが発生しました。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数

		が、シンボリックアドレス指定で使用可能です。
x0040000	TIO_SYNC_Data のデータが無効/不正です。	TIO_SYNC 命令およびその TIO_SYNC_Data 出力との相互接続をチェックします。
1005xxxx	エラーが RD_SINFO システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_SINFO からの RET_VAL 戻り値のエラー情報を示します。	• TIA Portal 情報システムの RD_SINFO の説明をお読みください。 • TIO_DI を必ず「Synchronous Cycle」OB で呼び出してください。
10060000	TM Timer DIDQ が見つかりません。 考えられる原因: ハードウェア識別子を使用して構成されたモジュールが、TM Timer DIDQ ではない。	• 設定されたモジュールが TM Timer DIDQ であることを確認します。 • HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。
10070000	アドレス計算中に発生した内部エラー。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が、シンボリックアドレス指定で使用可能です。
00080000	TIO モジュールが、TIO_SYNC 命令を使用して同期されていません。	命令 TIO_SYNC.をチェックします。
10090000	アイソクロナス OB の読み出しサイクルタイムが許可されている 0 < T_{APP} <= 16 ms 範囲外であるため、無効です。 命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	• サイクルタイムを訂正します。 • TIO_DI を必ず「Synchronous Cycle」OB で呼び出してください。
100A0000	TO 入力パラメータに割り当てられた時間が、許可されている 0 < T_o <= 4 ms 範囲外です。	TO 入力パラメータの値を訂正します。
100Bxxxx	エラーが RD_ADDR システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_ADDR からの RET_VAL 戻り値のエラー情報を示します。	TIA Portal 情報システムの RD_ADDR の説明をお読みください。
000C0000	変換されたタイムスタンプが無効です。 考えられる原因: 通信エラー	TIO モジュールとの PROFINET 通信をチェックします。
000D0000	デジタル入力の Quality Information は、デジタル入力でエラーが発生したことを示します。	デジタル入力の配線をチェックします。
000E0000	Channel 入力パラメータに割り当てられた数が、Timer DI として設定されたデジタル入力ではありません。	• チャンネル構成をチェックします (TM Timer DIDQ 16x24V の場合のみ)。 • デジタル入力の動作モードをチェックします。
000F0000	EdgeSel 入力パラメータに割り当てられた値が、有効な 1～5 の範囲外である。	EdgeSel 入力パラメータの値を訂正します。
10100000	送信クロックが許可されている 0 < SendClock <= 4 ms の範囲外であるため無効である。	送信クロックを修正します。

	エラーコードは以下のことを示す場合もあります。 • TIO_SYNC_Data のデータが無効か、存在しない。 • TIO_DI 命令がアイソクロナス OB で呼び出されていない。	
x0FF0000	一般的な内部エラー。	—

TIO_IOLink_OUT: 出力時刻制御されたプロセス出力信号

説明

Time-based IO を TIO_IOLink_OUT 命令と一緒に使用できます。TIO_IOLink_OUT を使用して、指定した時刻に IO-Link Device の出力データを有効にすることができます。

IO-Link Device はタイムスタンプファンクションを備えている必要があり、Port は「IO-Link, Time based OUT」モードにある必要があります。

スタートアップ特性

CPU のスタートアップ中、命令 TIO_IOLink_OUT は入力パラメータを 1 回適用し、以下をチェックします。

- HWID のチェック
- Port 番号が範囲内(1～4)にあるかをチェック
- TIO_SYNC_Data.Error のチェック: TIO_SYNC にエラーが存在するか?
- 正の値の T_o チェック
- タイプ IO-Link への設定のチェック
- IO-Link Time based OUT の設定に関する Port モードのチェック
- OB が「Synchronous Cycle」OB であるかをチェック
- PortQualifier のチェック

エラーなしでスタートアップが行われると、命令は通常の操作に切り替わります。 入力パラメータ REQ、Out_Mode、TimeStamp および AA_Data は、通常の操作中に変更できます。 エラーが発生した場合、命令は通常の操作に切り替わらず、エラーメッセージを生成します。

動作原理

通常の操作で、命令はプロセスデータ(AA_Data)を IO-Link Device に送信します。 出力データ AA_Data は、入力パラメータ TimeStamp で定義された時刻に有効になります。

入力 TIO_SYNC_Data を TIO_SYNC 命令の同じ名前の出力に接続します。 これにより、共有時間ベースを確保します。

「REQ」パラメータの立ち上がりエッジで、出力ジョブを開始します。 保留中のエラーがなく、ジョブが有効な場合にのみ新しいジョブを開始できます。 出力ジョブを開始するとき、AA_Data (ビット 0、1)が、TimeStamp で定義された時刻に、IO-Link Device で有効になります。 ジョブは、出力時刻に達する前に最後のアプリケーションサイクルが実行されると完了します(Done)。Status および Error は、ジョブランタイム中に常に更新されています。

注記

ジョブが立ち上がりエッジで開始すると、ジョブを再起動せずに新しい TimeStamp の入力によって、出力時刻を適合させることができます。

制約:

適合されたタイムスタンプが出力時刻前の 16 ms 未満の場合(TimeStamp - TIO_Time < 16)、最後の有効なタイムスタンプが使用されます。

TimeStamp として値 0 を指定すると、出力は入力 AA_Data で指定されたデータによって直接書き込まれます。 これにより、手動モードでタイムスタンプを使用せずに、TIO モジュールから直接制御を実行することができます。 進行中のジョブを中断するために、直接制御を使用できます。

エラーに対する応答

Error 出力は、命令が正しく処理されたかどうかを示します。 エラーの原因は Status 出力に表示されます。

パラメータ

次の表に、TIO_IOLink_OUT 命令のパラメータを示します。

パラメータ	宣言	データタイプ S7-1500	既定	説明
REQ	入力	Bool	False	ジョブは立ち上がりエッジで開始します。
HWID*	入力	HW_IO	0	HWCN からの TIO モジュールのハードウェア識別子
TO*	入力	LTime	LT#0ns	T_O: アイソクロナス出力データの出力時刻。 T_O を ET 200 ステーション (PROFINET インターフェースのプロパティ)から適用します。
Port*	入力	USInt	0	指定された IO-Link Device のポート番号(1～4)
TIO_SYNC_Data*	入力	"TIO_SYNC_Data"		TIO_SYNC 命令によって TIO モジュールの TIO 命令に提供されるシステム時刻。 UDT "TIO_SYNC_Data"を参照してください。 この入力パラメータを TIO_SYNC 命令の「TIO_SYNC_Data」出力パラメータに接続します。
Time-Stamp	入力	LTime	LT#0ns	タイムスタンプ: プロセスデータ(AAE1, AAE2)が出力される時刻。
AA_Data	入力	Word	16#0	プロセス出力データ: 出力される日付(ワード)。 ビット 0、1 の AAE1 および AAE2 を含みます。
Busy	出力	Bool	False	Busy = True: ジョブが未完了です。
Done	出力	Bool	False	Done = True が 1 つのサイクルについて表示されます。 ジョブは「完了、エラーなし」としてシグナルが送信されます。
Error	出力	Bool	False	True の場合、この出力はエラーが発生したことを示します。 この場合、BUSY および DONE が False に設定されます。 詳細は Status パラメータを参照してください。 Error は、エラーの対策が取られると直ちにリセットされます。
Status	出力	DWord	16#0	命令のステータス Status パラメータの説明を参照してください。
* CPU のスタートアップ時に 1 回チェックされます				

ステータスパラメータ

エラーコードまたはステータス情報は、ダブルワードとしてステータス出力で出力されます。

ダブルワードは以下のように分割されます。

エラーコードのセグメント	意味
z0yywwww	システムファンクションのエラー: • スタートアップ中(z = 0) • 通常の操作中(z = 1) 下位使用のあるシステムファンクションは yy でコーディングされます。 エラーコードの表を参照してください。 wwww はシステムファンクションの RET_VAL を指定します。 エラー情報については、システムファンクションのヘルプを参照してください。
z0yy0000	システムファンクションで発生していないエラー。 このエラーは連続したエラー番号 yy を受信します。 エラーは以下の場合に発生する可能性があります。 • スタートアップ中(z = 0) • 通常の操作中(z = 1)

エラーコードの表

エラーコード (DW#16#...)	意味	対処法
00000000	エラーは発生していません。	—
10010000	Port 入力パラメータに割り当てられた番号が、有効な 1～4 の範囲外である。	Port 入力パラメータの値を訂正します。
10020000	TIO_IOLink_OUT 命令は、アイソクロナス OB で呼び出されません。この命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	TIO_IOLink_OUT を必ず「Synchronous Cycle」OB で呼び出してください。
10030000	HWID 入力パラメータの読み出し時にエラーが発生しました。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が、シンボリックアドレス指定で使用可能です。
x0040000	TIO_SYNC_Data のデータが無効/不正です。	TIO_SYNC 命令およびその TIO_SYNC_Data 出力との相互接続をチェックします。
1005xxxx	エラーが RD_SINFO システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_SINFO からの RET_VAL 戻り値のエラー情報を示します。	• TIA Portal 情報システムの RD_SINFO の説明をお読みください。 • TIO_IOLink_OUT を必ず「Synchronous Cycle」OB で呼び出してください。
10060000	IO-Link Device が見つかりません。 考えられる原因:	• 構成されたモジュールが、Time-based IO 用の IO-Link Master であることを確認します。 • HWID 入力パラメータの値をチェックします。 TIO モジュールのハード

	ハードウェア識別子を使用して構成されたモジュールが、Time-based IO 用の IO-Link Master ではない。	ウェア識別子を HWCN モジュールプロパティから指定します。
10070000	アドレス計算中に発生した内部エラー。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が、シンボリックアドレス指定で使用可能です。
00080000	TIO モジュールが、TIO_SYNC 命令を使用して同期されていません。	命令 TIO_SYNC.をチェックします。
10090000	アイソクロナス OB の読み出しサイクルタイムが許可されている 0 < T_{APP} <= 16 ms 範囲外であるため、無効です。 命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	• サイクルタイムを訂正します。 • TIO_IOLink_OUT を必ず「Synchronous Cycle」OB で呼び出してください。
100A0000	TO 入力パラメータに割り当てられた時間が、許可されている 0 < T_o <= 4 ms 範囲外です。	TO 入力パラメータの値を訂正します。
100Bxxxx	エラーが RD_ADDR システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_ADDR からの RET_VAL 戻り値のエラー情報を示します。	TIA Portal 情報システムの RD_ADDR の説明をお読みください。
000C0000	TimeStamp 入力パラメータのタイムスタンプが無効です。	TimeStamp 入力パラメータをチェックします。
000D0000	IO-Link の値のステータス PortQualifier が、プロセスデータが無効であることを示しています。	接続されているセンサとその設定をチェックします。
100E0000	IO-Link の設定されたポートモードが不正です。	S7-PCT を使用して、接続されたセンサの設定をチェックします。
100F0000	「Synchronous Cycle」OB の読み出しサイクルタイムが長すぎます。 T_{APP} > 16 ms.	サイクルタイムとして送信クロックの倍数を小さく設定します。
x0FF0000	一般的な内部エラー。	—

TIO_DQ: 出力エッジがデジタル出力で時刻制御されている

説明

Time-based IO を TIO_DQ 命令と一緒に使用できます。TIO_DQ が、指定した時刻に TM Timer DIDQ のデジタル出力を切り替えます。

スタートアップ特性

CPU のスタートアップ中、命令 TIO_DQ は入力パラメータを 1 回適用し、以下をチェックします。

- HWID のチェック
- デジタル入力の数(Channel)が許可されている範囲であるかどうかをチェック(アドレス指定されたモジュールとチャンネル構成に依存する)
- TIO_SYNC_Data.Error のチェック: TIO_SYNC にエラーが存在するか?
- T_o 有効期限のチェック(0 ms ~ 4 ms)
- OB が「Synchronous Cycle」OB であるかをチェック

エラーなしでスタートアップが行われると、命令は通常の操作に切り替わります。 入力パラメータ REQ、Out_Mode、TimeStampRE および TimeStampFE は、通常の操作中に変更できます。 エラーが発生した場合、命令は通常の操作に切り替わらず、エラーメッセージを生成します。

動作原理

命令は、通常の操作中にデジタル出力で時刻制御されたエッジを出力します。

- TimeStampRE 入力パラメータで定義した時刻で、立ち上がりエッジがデジタル出力.で出力されます。
- TimeStampFE 入力パラメータで定義した時刻で、立ち下がりエッジがデジタル出力.で出力されます。

入力パラメータ Out_Mode を使用して、片方のエッジのみ、または両方のエッジが出力されるかどうかを決定します。

入力 TIO_SYNC_Data を TIO_SYNC 命令の同じ名前の出力に接続します。 これにより、共有時間ベースを確保します。

「REQ」パラメータの立ち上がりエッジで、出力ジョブを開始します。 保留中のエラーがなく、ジョブが有効な場合にのみ新しいジョブを開始できます。 出力ジョブを開始するとき、デジタル出力が TimeStampRE および TimeStampFE によって定義した時刻で切り替わります。

- デジタル出力が既に時刻 TimeStampRE で設定されている場合、出力ジョブは立ち上がりエッジがあっても、モジュールに転送されません。
- デジタル出力が時刻 TimeStampFE で設定されていない場合、出力ジョブは立ち下がりエッジがあっても、モジュールに転送されません。

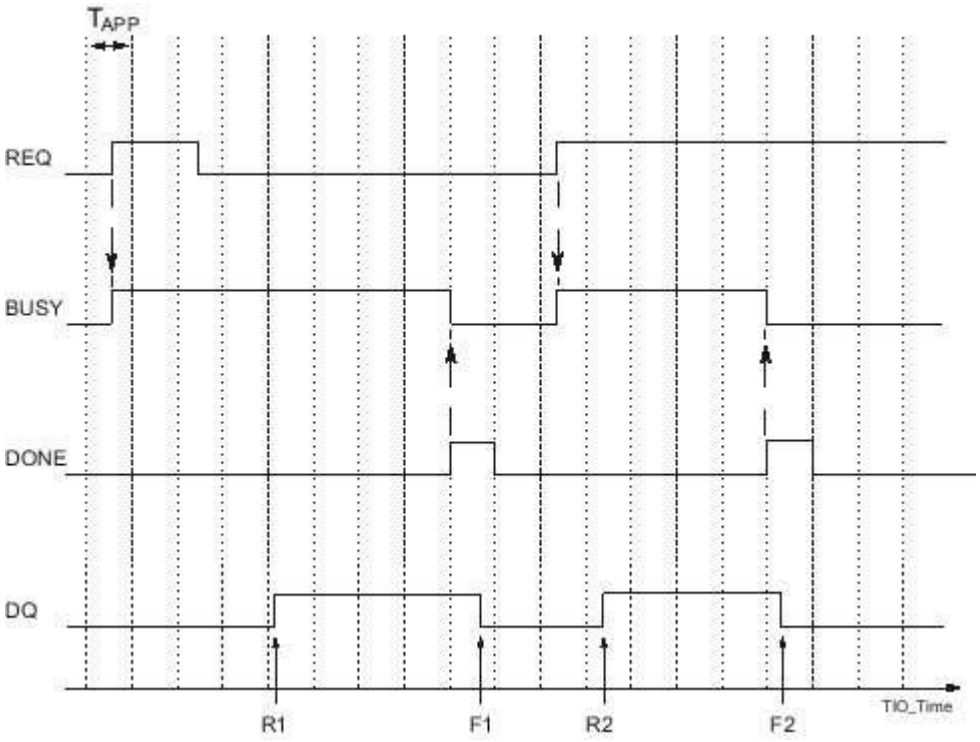
つまり、デジタル出力はいずれの場合も切り替わりません。

ジョブは、出力時刻に達する前に最後のアプリケーションサイクルが実行されると完了します(DONE)。Status および Error は、ジョブランタイム中に常に更新されています。 TimeStampRE または TimeStampFE を無効なタイムスタンプによって適合させることによって、有効な出力ジョブをキャンセルできます。

次の図に、以下の条件での出力ジョブの開始でビット DONE および BUSY の応答の例を示します。

- Out_Mode = 2

- 2 つのタイムスタンプは、ジョブの開始と出力の間で適合されていません。



T_{APP}	アプリケーションサイクル
R1, R2	DQ の立ち上がりエッジの指定された時刻
F1, F2	DQ の立ち下がりエッジの指定された時刻

注記
ジョブが立ち上がりエッジで開始すると、ジョブを再起動せずに TimeStampRE および TimeStampFE の入力によって、出力時刻を適合させることができます。

制約:

適合されたタイムスタンプが出力時刻からの 2 つのアプリケーションサイクルよりも小さい場合 (TimeStampRE - TIO_Time < 2*T_{APP} または TimeStampFE - TIO_Time < 2*T_{APP})、最後の有効なタイムスタンプが使用されます。

TimeStampRE または TimeStampFE として値「0」を指定する場合、入力パラメータ Out_Mode = 3 によってデジタル出力で直接個々のエッジを出力できます。これにより、手動モードでタイムスタンプなしに TIO モジュールから直接制御を実行することができます。 進行中のジョブを中断するために、直接制御を使用できます。

入力パラメータへの変更は、TIO_DQ 命令が CPU のスタートアップを検出するまで適用されません。

エラーに対する応答

Error 出力は、命令が正しく処理されたかどうかを示します。 エラーの原因は Status 出力に表示されます。

パラメータ

次の表に、TIO_DQ 命令のパラメータを示します。

パラメータ	宣言	データタイプ S7-1500	既定	説明
REQ	入力	Bool	False	ジョブは立ち上がりエッジで開始します。
HWID*	入力	HW_IO	0	HWCN からの TIO モジュールのハードウェア識別子
Channel*	入力	UInt	0	接続される TM Timer DIDQ のデジタル出力の数 (0 ... m)
TIO_SYNC_Data*	入力	"TIO_SYNC_Data"		TIO_SYNC 命令によって TIO モジュールの TIO 命令に提供されるシステム時刻。 UDT "TIO_SYNC_Data"を参照してください。 この入力パラメータを TIO_SYNC 命令の「TIO_SYNC_Data」出力パラメータに接続します。
Out_Mode	入力	UInt	2	デジタル出力でのエッジの出力モードを指定します。 0_D: 立ち上がりエッジのみが出力されます(TimeStampRE)。 1_D: 立ち下がりエッジのみが出力されます(TimeStampFE)。 2_D: 両方のエッジが出力されます(TimeStampRE および TimeStampFE)。 3_D: 各エッジは TimeStampRE = 0 または TimeStampFE = 0 のとき直接出力されます。 両方のタイムスタンプの値が「0」、またはいずれのタイムスタンプの値も「0」でない場合、エッジは出力されません。 4 bis 255_D: 予備
TO*	入力	LTime	LT#0ns	T_O: アイソクロナス出力データの出力時刻。 T_O を ET 200 ステーション (PROFINET インターフェースのプロパティ)から適用します。
TimeStampRE	入力	LTime	LT#0ns	タイムスタンプ: 立ち上がりエッジが出力される時刻。
TimeStampFE	入力	LTime	LT#0ns	タイムスタンプ: 立ち下がりエッジが出力される時刻。
StatusDQ	出力	Bool	False	デジタル出力の現在のステータス。 デジタル出力の反転が設定されている場合、 StatusDQ も反転されます。 TIO_DQ は、特定のデジタル出力とその現在の状態のパラメータ割り当てをチェックできません。 StatusDQ は、HW イネーブルが設定され、出力ジョブ中にデジタル出力が有効にされない場合、不正な値を返す可能性があります。
BUSY	出力	Bool	False	BUSY = True: ジョブが未完了です。

DONE	出力	Bool	False	DONE = True が 1 つのサイクルについて表示されます。 ジョブは「完了、エラーなし」としてシグナルが送信されます。
Status	出力	DWord	16#0	命令のステータス Status パラメータの説明を参照してください。
Error	出力	Bool	False	True の場合、この出力はエラーが発生したことを示します。 この場合、BUSY および DONE が False に設定されます。 詳細は Status パラメータを参照してください。 Error は、エラーの対策が取られると直ちにリセットされます。
* CPU のスタートアップ時に 1 回チェックされます				

ステータスパラメータ

エラーコードまたはステータス情報は、ダブルワードとしてステータス出力で出力されます。

ダブルワードは以下のように分割されます。

エラーコードのセグメント	意味
z0yywwww	システムファンクションのエラー: • スタートアップ中(z = 0) • 通常の操作中(z = 1) 下位使用のあるシステムファンクションは yy でコーディングされます。 エラーコードの表を参照してください。 wwww はシステムファンクションの RET_VAL を指定します。 エラー情報については、システムファンクションのヘルプを参照してください。
z0yy0000	システムファンクションで発生していないエラー。 このエラーは連続したエラー番号 yy を受信します。 エラーは以下の場合に発生する可能性があります。 • スタートアップ中(z = 0) • 通常の操作中(z = 1)

エラーコードの表

エラーコード (DW#16#...)	意味	対処法
00000000	エラーは発生していません。	—
10010000	Channel 入力パラメータに割り当てられたデジタル出力の数が、許可されている範囲外です(アドレス指定されたモジュールとチャンネル構成に依存する)。	Channel 入力パラメータの値を訂正します。

10020000	TIO_DQ 命令は、アイソクロナス OB で呼び出されません。この命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	TIO_DQ を必ず「Synchronous Cycle」OB で呼び出してください。
10030000	HWID 入力パラメータの読み出し時にエラーが発生しました。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が、シンボリックアドレス指定で使用可能です。
x0040000	TIO_SYNC_Data のデータが無効/不正です。	TIO_SYNC 命令およびその TIO_SYNC_Data 出力との相互接続をチェックします。
1005xxxx	エラーが RD_SINFO システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_SINFO からの RET_VAL 戻り値のエラー情報を示します。	• TIA Portal 情報システムの RD_SINFO の説明をお読みください。 • TIO_SYNC を必ず「Synchronous Cycle」OB で呼び出してください。
10060000	TM Timer DIDQ が見つかりません。 考えられる原因: ハードウェア識別子を使用して構成されたモジュールが、TM Timer DIDQ ではない。	• 設定されたモジュールが TM Timer DIDQ であることを確認します。 • HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。
10070000	アドレス計算中に発生した内部エラー。	HWID 入力パラメータの値をチェックします。 TIO モジュールのハードウェア識別子を HWCN モジュールプロパティから指定します。 データタイプ Hw_SubModule の内部システム定数が、シンボリックアドレス指定で使用可能です。
x0080000	TIO モジュールが、TIO_SYNC 命令を使用して同期されていません。 エラーコードは以下のことを示す場合もあります。 • 命令の最初の実行前に、ジョブが既に存在していた。 • Channel 入力パラメータに割り当てられた数がデジタル出力ではない。	命令 TIO_SYNC.をチェックします。
10090000	アイソクロナス OB の読み取りサイクルタイムは LT#0ms または負の値であるため、無効です。 値を訂正します。 命令は、アイソクロナス OB でエラーがない場合にのみ使用できます。	• サイクルタイムを訂正します。 • TIO_DQ を必ず「Synchronous Cycle」OB で呼び出してください。
100A0000	TO 入力パラメータに割り当てられた時間が、許可されている 0 < T_o <= 4 ms 範囲外です。	TO 入力パラメータの値を訂正します。
100Bxxxx	エラーが RD_ADDR システムファンクションの実行中に発生しました。 下位ワード xxxx は、RD_ADDR からの RET_VAL 戻り値のエラー情報を示します。	TIA Portal 情報システムの RD_ADDR の説明をお読みください。
000C0000	TimeStampRE および TimeStampFE 入力パラメータの片方または両方のタイムスタンプが無	TimeStampRE および TimeStampFE 入力パラメータをチェックします。

	効です。 エラーは、1 つのアプリケーションサイクルについてのみシグナルが送信されます。	
000D0000	デジタル出力の Quality Information は、デジタル入力でエラーが発生したことを示します。	• デジタル出力のパラメータ割り当てをチェックします。 • デジタル出力の配線で短絡、過負荷、および過熱をチェックします。
000E0000	Channel 入力パラメータに設定された数が、Timer DQ として設定されたデジタル出力ではありません。	• チャンネル構成をチェックします (TM Timer DIDQ 16x24V の場合のみ)。 • デジタル出力の動作モードをチェックします。
100F0000	「Synchronous Cycle」OB の読み出しサイクルタイムが長すぎます。 T_{APP} > 16 ms.	サイクルタイムとして送信クロックの倍数を小さく設定します。
10100000	送信クロックが許可されている 0 < SEND_CLOCK <= 4 ms の範囲外であるため無効である。 エラーコードは以下のことを示す場合もあります。 • TIO_SYNC_Data のデータが無効か、存在しない。 • TIO_DQ 命令がアイソクロナス OB で呼び出されていない。	送信クロックを修正します。
00110000	Out_Mode 入力パラメータに割り当てられた値が、有効な 0 ~ 3 の範囲外である。	Out_Mode 入力パラメータの値を訂正します。
x0FF0000	一般的な内部エラー。	—

UDT "TIO_SYNC_Data"

説明

データタイプ UDT「TIO_SYNC_Data」は、モジュールの同期および TIO_Time の経過の中央構造体とデータを含みます。

パラメータ

パラメータ	データタイプ	説明
	S7-1500	
TIO_TIME_BASE	LTime	内部使用
TIO_TIME	LTime	TIO モジュールの共有時刻ベース(相対時刻)。
PIP_MODE	USInt	データ更新のモード(TIO_SYNC 命令の PIP_Mode 入力パラメータによって転送されます)
APP_CYC	LTime	「Synchronous Cycle」OB のアプリケーションサイクル
SEND_CLOCK	LTime	同期ドメインの送信クロック(TIO_SYNC 命令の Send-Clock 入力パラメータによって転送されます)
TBase	LTime	内部使用
ERROR	Bool	内部使用

通信

この章には下記に関する情報が記載されています：

- S7 通信 (S7-1200, S7-1500)
- 開放型ユーザー間通信 (S7-1200, S7-1500)
- Web サーバー (S7-1200, S7-1500)
- 通信プロセッサ (S7-1200, S7-1500)
- TeleService (S7-1200)

S7 通信

この章には下記に関する情報が記載されています：

データの一貫性

定義

並行プロセスによって変更できないデータブロックは、整合性のあるデータ領域で呼び出されます。その結果、整合性のあるデータ領域より大きいデータブロックを転送時に全体として変更できます。すなわち、グループを形成し、整合性のあるデータ領域より大きいデータブロックが、同時に、新しい整合性のあるデータの部分と古い整合性のあるデータの部分から構成できます。

例

たとえば、高優先度のハードウェア割り込み OB によって通信用命令が割り込まれた場合、不整合が生じる可能性があります。 この OB 内のユーザープログラムが、既に命令によって部分的に処理されているデータを変更する場合、転送データは以下から発生します。

- 部分的に、ハードウェア割り込みが処理される前の時点から
- および部分的に、ハードウェア割り込みが処理された後の時点から

つまり、これらのデータは矛盾しています(一貫性がありません)。

データ整合性の確保

通信プロセスが割り込み OB によって割り込まれる可能性がある場合は、デーの整合性を持つ転送を確保する必要があります。 転送するデータでなく、データのイメージのみが割り込み OB によって直接に変更されないことを確認します。次のデータ転送よりも前に、データのイメージを通信命令の転送領域にコピーします。

- ユーザープログラムに、共通データにアクセスする通信命令が含まれる場合、このデータ領域へのアクセスは、たとえば、DONE パラメータによって調整できます。 このため、通信命令でローカルに転送される通信領域のデータの整合性は、ユーザープログラム内で確保できます。
- S7 通信命令「PUT」/「GET」の場合、ユーザープログラムへの通信データを同期化するために使用できる通信ブロックがターゲットデバイス(サーバー)のユーザープログラム内にないため、一貫性データ領域のサイズは、プログラミング時または設定時に既に考慮されていることが必要です。
- S7-300 および C7-300 の場合(例外: CPU 318-2 DP)、通信データはオペレーティングシステムのサイクル制御ポイントで、32 バイトのブロック内のユーザーメモリに連続的にコピーされます。 これよりも大きいデータ領域に対してはデータの整合性は保証されていません。 定義済みのデーター貫性が必要な場合、ユーザープログラム内の通信データは 32 バイトを超えることはできません(バージョンに応じて、最大 8 バイト)。
- 他方、S7-400 および S7-1500 では、462 バイトのブロック内の通信データは、サイクル制御ポイントではなく、プログラムサイクル中に固定タイムスライスで処理されます。 タグの一貫性はシステムによって確保されます。 これらの通信領域には、「PUT」/「GET」命令を使用して、あるいは、たとえば、OP または OS によるタグの読み取り/書き込み時に一貫してアクセスできます。

> **注記**
> データの整合性に関する追加情報については、特定の命令の説明を参照してください。

割り込み応答時間に対する影響

CPU の割り込み応答時間は、データのコピーによって少し長くなります。 絶対的な一貫性を持って転送する必要があるデータの量が多ければ多いほど、システムの割り込み応答時間は長くなります。

S7 通信用の命令の共通パラメータ

分類

S7 通信用の命令のパラメータは、そのファンクションに応じて次の 5 つのカテゴリに分けることができます。

1. 制御パラメータを使用して、命令を有効にします。
2. アドレス指定パラメータを使用して、リモート通信パートナーをアドレス指定します。
3. 送信パラメータは、リモートパートナーに送信されるデータ領域をポイントします。
4. 受信パラメータは、リモートパートナーから受信したデータが入力されるデータ領域をポイントします。
5. ステータスパラメータを使用して、命令がそのタスクを エラーなしで完了したかどうかをモニタしたり、発生した エラーを分析したりします。

制御パラメータ

データ交換が有効になるのは、命令が呼び出されたときや、前の呼び出し以降に値が特定の変更を受けたとき(たとえば、立ち上がり エッジ)に、適切な制御パラメータが定義済み値を持つ(たとえば、セットされる)場合のみです。

アドレス指定パラメータ

パラメータ	説明
ID	(接続の構成によって指定された)ローカル接続記述子への参照。
R_ID	R_ID パラメータを使用して、送信命令と受信命令がともに属するように指定します。R_ID パラメータは、送信側の命令と受信側の命令において一致する必要があります。 これによって、同じ論理接続を介した複数の命令ペアの通信が可能になります。 • R_ID は、DW#16#wxyzWXYZ 形式で指定する必要があります。 • R_ID で指定された論理接続の命令ペアは、この接続に対して一意である必要があります。

注記

アドレス指定パラメータ ID および R_ID

ランタイム中にアドレス指定パラメータ ID および R_ID を再割り当てできます。 新しいパラメータは、前のジョブが終了した後に、新しい各ジョブによって検証されます。

次のオプションを使用して、インスタンス DB の数ひいては必要なワークメモリを減らすことができます。

1. タグ ID で、1 つのデータインスタンスブロックを介して複数の接続を使用できます。
2. タグ R_ID で、シングルインスタンスを持つジョブについて送信命令と受信命令の複数のペアを定義できます。
3. ケース 1 とケース 2 を組み合わせることができます。

前回のジョブが実行された後に新しいパラメータが有効になることに注意してください。 送信操作を有効にするときは、送信側の命令の R_ID パラメータと、受信側のこれに対応するものが一致する必要があります。

STATUS パラメータ

ステータスパラメータを使用して、命令がそのタスクを正常に完了したかどうかや、まだアクティブであるかどうかをモニタします。 また、ステータスパラメータはエラーも示します。

注記

ステータスパラメータは、1 サイクルに対してのみ有効です。つまり、呼び出しに続く最初のコマンドから次の呼び出しまでです。 そのため、各命令サイクルの後でこれらのパラメータを評価する必要があります。

送信パラメータと受信パラメータ

両端で設定された通信命令の場合

- 使用する SD_i および RD_i パラメータの数が送信側と受信側で一致する必要があります。
- ともに属する SD_i および RD_i パラメータのデータタイプが送信側と受信側で一致する必要があります。
- SD_i パラメータに応じて送信されるデータの量は、対応する RD_i によって使用可能になる範囲を超えてはなりません(「BSEND」/「BRCV」には適用されません)。 RD_i パラメータのデータサイズは同一である必要があります(「BSEND」/「BRCV」は例外)。

上のルールに従っていない場合、このことは ERROR = 1 および STATUS = 4 によって示されます。

注記

送信パラメータと受信パラメータの提供

VARIANT データタイプの場合、通信命令が呼び出されるときには必ず送信および受信パラメータが提供される必要があります。 たとえば、スタートアップ時に通信命令の送信パラメータを提供することや、サイクリックオペレーションでのみ送信ジョブのトリガを行うことはできません。

ユーザーデータサイズ

「USEND」、「URCV」、「GET」および「PUT」命令では、転送されるデータの量は、定義済みのユーザーデータサイズを超えてはなりません。 最大ユーザーデータサイズは次に依存します。

- 使用する命令
- 通信パートナー

1～4 つのタグを持つ命令に対して保証されているユーザーデータの最小サイズを次の表に示します。

命令	パートナー: S7-300	パートナー: S7-400	パートナー: S7-1200	パートナー: S7-1500
PUT / GET	160 バイト	400 バイト	160 バイト	880 バイト
USEND / URCV	160 バイト	440 バイト	-	920 バイト
BSEND / BRCV	32768/65534 バイト	65534 バイト	-	• 標準アクセスの 65534 バイト • 最適化されたアクセスの 65535 バイト

ユーザーデータサイズに関する詳細情報については、各 CPU の技術データを参照してください。

正確なユーザーデータサイズ

上で指定されたユーザーデータサイズが不十分である場合は、次のようにユーザーデータの最大バイト長を決定することができます。

まず、次の表から通信に対して有効なデータブロックサイズを読み取ります。

ローカル CPU	リモート CPU	バイト単位のデータブロックサイズ
S7-1200	任意	240
S7-1500	S7-300	240
	S7-400	480
	S7-1200	240
	S7-1500	960

次の表でこの値を使用して、使用されるパラメータの合計として、バイト単位の可能な最大ユーザーデータ長を読み取ります。 これは、領域 SD_i、RD_i および ADDR_i の偶数長に適用されます。

奇数長の範囲ごとに、可能な最大ユーザーデータ長は 1 バイト減らされます。

		使用される SD_i、RD_i、ADDR_i パラメータの数			
データブロックサイズ	命令	1	2	3	4
240 (S7-300)	PUT/GET/ USEND	160	-	-	-
240 (統合インターフェースを介した S7-300)	PUT	212	-	-	-
	GET	222	-	-	-
	USEND	212	-	-	-
240 (S7-400)	PUT	212	196	180	164
	GET	222	218	214	210
	USEND	212	-	-	-
480 (S7-400)	PUT	452	436	420	404
	GET	462	458	454	450
	USEND	452	448	444	440
240 (S7-1200)	PUT	212	196	180	164
	GET	222	218	214	210
960 (S7-1500)	PUT	932	916	900	884
	GET	942	938	934	930
	USEND	932	928	924	920

GET: リモート CPU からデータを読み取ります。

説明

「GET」命令を使用して、リモート CPU からデータを読み取ることができます。

命令は、制御入力 REQ での立ち上がりエッジで開始されます。

- その後、読み出される領域への関連ポインタ(ADDR_i)がパートナー CPU に送信されます。 パートナー CPU は、RUN または STOP モードでも問題ありません。
- パートナー CPU は、以下のデータを返します。
  - 応答が最大ユーザーデータ長を超える場合、これは STATUS パラメータのエラーコード「2」で表示されます。
  - 受信したデータは、次の呼び出し時に設定済み受信領域(RD_i)にコピーされます。
- この操作が完了したことは、ステータスパラメータ NDR の値が「1」になることによって示されます。

前の読み取りプロセスが完了した後にしか読み取りを再度有効にできません。 データの読み出し中にアクセス問題が発生した場合や、データタイプチェックの結果、エラーが発生した場合は、エラーおよび警告が ERROR および STATUS を介して出力されます。

パートナー CPU 上にアドレス指定されたデータ領域内での変更は、「GET」命令では登録されません。

命令使用の必要条件

- 「リモートパートナーからの PUT/GET 通信によるアクセスの許可」ファンクションが、パートナー CPU のプロパティの[保護]で有効にされていること。
- 「GET」命令でアクセスするブロックが、アクセスタイプ「標準」で作成されているっこと。
- ADDR_i および SD_i パラメータで定義された領域が番号、長さ、およびデータタイプの点で確実に一致するようにしてください。
- 読み出し元の領域(ADDR_i パラメータ)をデータストレージ領域(RD_i パラメータ)よりも大きくすることはできません。

パラメータ

次の表に、「GET」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ request。立ち上がりエッジでデータ交換を有効にします。
ID	Input	WORD	I、Q、M、D、L、または定数	パートナー CPU への接続を指定するアドレスパラメータ。
NDR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ NDR: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブが正常に完了しました。

ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR および STATUS、エラーコード: • ERROR=0 STATUS の値は次のとおりです。 ○ 0000H: 警告もエラーもなし ○ <> 0000H: 警告、STATUS に詳細情報を表示。 • ERROR=1 エラーが発生しました。STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	
ADDR_1	InOut	REMOTE	I、Q、M、D	読み取られるパートナー CPU 上の領域へのポインタ。 REMOTE ポインタが DB にアクセスするときは、その DB を指定する必要があります。 例: P#DB10.DBX5.0 Byte 10.
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
RD_1	InOut	VARIANT	I、Q、M、D、L	読み取りデータが入力されるローカル CPU 上の領域へのポインタ。
RD_2	InOut	VARIANT		
RD_3	InOut	VARIANT		
RD_4	InOut	VARIANT		

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

次の表には、ERROR および STATUS パラメータを介して出力できる「GET」命令に固有のエラー情報がすべて含まれています。

ERROR	STATUS (10 進)	説明
0	11	警告: 前のジョブがまだ使用中であるため、新しいジョブはアクティブではありません。
0	25	通信が開始しました。 ジョブが処理中です。
1	1	たとえば、通信上の問題。 • 接続記述子がロードされていません(ローカルまたはリモート) • 接続が中断されました(例: ケーブル、CPU オフ、CP が STOP モード) • パートナーへの接続がまだ確立されていません
1	2	• パートナーデバイスからの否定確認。 ファンクションを実行できません。 • リモートステーションからの応答が、最大ユーザーデータ長を超過しました(参照: S7 通信用の命令の共通パラメータ). • パートナー CPU で、アクセス保護が有効にされています。 CPU 設定で、アクセス保護を無効にします。
1	4	データストレージ RD_i へのポインタのエラー: • パラメータ RD_i と ADDR_i のデータタイプには互換性はありません。

		• 領域 RD_i の長さが、読み取られる ADDR_i パラメータのデータの長さよりも小さな値です。
1	8	パートナー CPU 上でアクセスエラーが発生しました。
1	10	ローカルユーザーメモリにアクセスできません(たとえば、削除された DB にアクセスした場合など)。
1	20	• 並列ジョブの最大数を超えました。 • 優先度の低い優先度クラスで、現在ジョブが処理されています(最初の呼び出し)。

注記

データの整合性

別のジョブを開始する前に、受信領域 RD_i の現在使用中の部分を完全に読み取った場合、データは一貫した状態で受信されます。

PUT: リモート CPU にデータを書き込みます。

説明

「PUT」命令を使用して、リモート CPU にデータを書き込むことが可能です。

命令は、制御入力 REQ での立ち上がりエッジで開始されます。

- その後、書き込まれる領域へのポインタ(ADDR_i)とデータ(SD_i)がパートナー CPU に送信されます。 パートナー CPU は、RUN または STOP モードでも問題ありません。
- 送信するデータは、設定された送信領域((SD_i)からコピーされます。 パートナー CPU は、データで指定されたアドレスに送信されたデータを保存し、実行確認を返します。
- エラーが発生しない場合、このことは、次の命令呼び出し時にステータスパラメータ DONE =「1」で示されます。 前回のジョブが完了した後にしか書き込みプロセスを再度有効にできません。

データの書き込み中にアクセス問題が発生した場合や、実行チェックの結果、エラーが発生した場合は、エラーおよび警告が ERROR および STATUS を介して出力されます。

命令使用の必要条件

- 「リモートパートナーからの PUT/GET 通信によるアクセスの許可」ファンクションが、パートナー CPU のプロパティの[保護]で有効にされていること。
- 「PUT」命令でアクセスするブロックが、アクセスタイプ「標準」で作成されていること。
- ADDR_i および SD_i パラメータで定義された領域が番号、長さ、およびデータタイプの点で確実に一致するようにしてください。
- 書き込まれる領域(ADDR_i パラメータ)は送信領域(SD_i パラメータ)と同じ大きさであることが必要です。

パラメータ

次の表に、「PUT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ request。立ち上がりエッジでデータ交換を有効にします。
ID	Input	WORD	I、Q、M、D、L、または定数	パートナー CPU への接続を指定するアドレスパラメータ。
DONE	Output	BOOL	I、Q、M、D、L	ステータスパラメータ DONE: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR および STATUS、エラーコード: • ERROR=0 STATUS の値は次のとおりです。 ○ 0000H: 警告もエラーもなし
STATUS	Output	WORD	I、Q、M、D、L	

				○ <> 0000H: 警告、STATUS に詳細情報を表示。 • ERROR=1 エラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。
ADDR_1	InOut	REMOTE	I、Q、M、D	データが書き込まれるパートナー CPU 上の領域へのポインタ。 REMOTE ポインタが DB にアクセスするときは、その DB を指定する必要があります。 例: P#DB10.DBX5.0 Byte 10. データ構造体を送信するときは(たとえば、Struct)、ADDR_i パラメータでデータタイプ CHAR を使用する必要があります。
ADDR_2	InOut	REMOTE		
ADDR_3	InOut	REMOTE		
ADDR_4	InOut	REMOTE		
SD_1	InOut	VARIANT	I、Q、M、D、L	送信されるデータを格納するローカル CPU 上の領域へのポインタ。 データタイプ BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL のみが許可されています。 データ構造体を送信するときは(たとえば、Struct)、 SD_i パラメータでデータタイプ CHAR を使用する必要があります。
SD_2	InOut	VARIANT		
SD_3	InOut	VARIANT		
SD_4	InOut	VARIANT		

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

次の表には、ERROR および STATUS パラメータを介して出力できる「PUT」命令に固有のエラー情報がすべて含まれています。

ERROR	STATUS (10 進)	説明
0	11	警告: 前のジョブがまだ使用中であるため、新しいジョブはアクティブではありません。
0	25	通信が開始しました。 ジョブが処理中です。
1	1	たとえば、通信上の問題。 • 接続記述子がロードされていません(ローカルまたはリモート)。 • 接続が切断されました(たとえば、ケーブル、CPU オフ、CP が STOP モード)。 • パートナーへの接続がまだ確立されていません。
1	2	• パートナー CPU の否定確認。 ファンクションを実行できません。 • パートナー CPU でのアクセスが保証されませんでした。 CPU 設定で、アクセスを有効にします。
1	4	データストレージへのポインタのエラー: • パラメータ SD_i と ADDR_i のデータタイプには互換性はありません。

		• 領域 SD_i の長さが、書き込まれる ADDR_i パラメータのデータの長さよりも大きな値です。 • SD_i にアクセスできません。 • 最大ユーザーデータサイズを超過しました。 • SD_i および ADDR_i パラメータの数が一致しません。
1	8	パートナー CPU とのアクセスエラー(たとえば、DB が読み込まれていない、または書き込み保護されている)。
1	10	ローカルユーザーメモリにアクセスできません(たとえば、削除された DB にアクセスした場合など)。
1	20	• 並列ジョブの最大数を超えました。 • 優先度の低い優先度クラスで、現在ジョブが処理されています(最初の呼び出し)。

データの整合性

送信操作が有効にされると(REQ での立ち上がりエッジ)、送信領域 SD_i から送信されるデータが、ユーザープログラムからコピーされます。 ブロック呼び出しの後で、現在の送信データを破損することなく、これらの領域に書き込むことができます。

注記
送信操作は、DONE ステータスパラメータの値が「1」であるときにのみ完了します。

その他

この章には下記に関する情報が記載されています：

- USEND: 未調整データの送信 (S7-1500)
- URCV: 未調整のデータ受信 (S7-1500)
- BSEND: セグメントのデータ送信 (S7-1500)
- BRCV: セグメントのデータ受信 (S7-1500)

USEND: 未調整データの送信

説明

「USEND」命令は、「URCV」タイプのリモートパートナー命令にデータを送信します。 送信プロセスは、パートナー命令との調整なしで実行されます。 すなわち、データ転送は、パートナー命令による確認なしで実行されます。

送信操作が有効にされると(REQ での立ち上がりエッジ)、送信領域 SD_i から送信されるデータが、ユーザープログラムからコピーされます。 命令呼び出しの後で、現在の送信データを破損することなく、これらの送信領域に書き込むことができます。

送信操作が正常に完了したことは、DONE ステータスパラメータの値が「1」にセットされることによって示されます。

パラメータ

次の表に、「USEND」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ request。立ち上がりエッジでデータ交換を有効にします。
ID	Input	CONN_PRG	I、Q、M、D、L、P、または定数	パートナー CPU への接続を指定するアドレスパラメータ。
R_ID	Input	CONN_R_ID	I、Q、M、D、L、または定数	命令ペア「USEND」および「URCV」を定義するアドレス指定パラメータ R_ID。 関連項目 S7 通信用の命令の共通パラメータ
DONE	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 警告もエラーもなし。 • 1: エラーが発生しました。STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ 「ERROR および STATUS パラメータ」の表を参照してください。
SD_i (1≤ i ≤4)	InOut	VARIANT	I、Q、M、D	i 番目の送信領域のポインタ。 BOOL 、 BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、または STRUCT データタイプのみが許可され

				ています(ビット配列は許可されていません)。. SD_i パラメータの最大ユーザーデータサイズは、パートナー CPU(「URCV」命令)および使用されるパラメータ数によって異なります。 追加情報は「」を参照してください。 S7 通信用の命令の共通パラメータ

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

ER-ROR	STATUS (10 進)	説明
0	11	警告: 前のジョブがまだ使用中であるため、新しいジョブはアクティブではありません。
0	25	通信が開始しました。 ジョブが処理中です。
1	1	通信上の問題が発生しました。 考えられる原因: • 接続記述子がロードされていません(ローカルまたはリモート) • 接続が中断されました(例: ケーブル、CPU オフ、CP が STOP モード) • パートナーへの接続がまだ確立されていません
1	4	• データ長やデータタイプに関する送信領域ポインタ SD_i でエラーが発生しました。 • 最大ユーザーデータ長を超過しました。
1	10	ローカルユーザーメモリにアクセスできません(たとえば、削除された DB にアクセスした場合など)。
1	18	R_ID パラメータの値が、ID パラメータで指定された接続内に既に存在しています(R_ID の値は接続内で一意である必要があります)。
1	20	• 並列ジョブの最大数を超えました。 • 優先度のさらに低い優先度クラスで、現在ジョブが処理されています(最初の呼び出し)。

URCV: 未調整のデータ受信

説明

「URCV」命令は、「USEND」タイプのリモートパートナー命令からデータを非同期的に受信し、このデータを設定済み受信領域にコピーします。

EN_R 入力に論理 1 がある場合に、この命令は受信準備ができています。 アクティブなジョブは、EN_R=0 でキャンセルできます。

受信データ領域は、パラメータ RD_1 ~ RD_4 によって参照されます。 (関連するパートナー命令「USEND」で)パラメータ RD_i / RD_1 および SD_i / SD_1 によって定義された領域の数と長さが一致することを確認してください。

コピー操作が正常に完了したことは、NDR ステータスパラメータの値が論理「1」にセットされたときに示されます。 ステータスパラメータ NDR が「1」に切り替わると、受信領域(RD_i)に新しい受信データが格納されてることを示します。 新しいブロック呼び出しを行うと、これらのデータが新しい受信データで上書きされる可能性があります。 これを防ぐには、受信データの処理を終了するまでに、EN_R に値 0 を入れて「URCV」を呼び出します(たとえば、サイクリックブロックの処理中に)。

パラメータ

次の表に、「URCV」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ enabled to receive。入力が設定された場合に受信準備完了を信号で通知します。
ID	Input	CONN_PRG	I、Q、M、D、L、P、または定数	パートナー CPU への接続を指定するアドレスパラメータ。
R_ID	Input	CONN_R_ID	I、Q、M、D、L、または定数	命令ペア「USEND」および「URCV」を定義するアドレス指定パラメータ。 関連項目: S7 通信用の命令の共通パラメータ
NDR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブが正常に完了しました。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 警告もエラーもなし • 1: エラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ 「ERROR および STATUS パラメータ」の表を参照してください。
RD_i (1≤ i ≤4)	InOut	VARIANT	I、Q、M、D	i 番目の受信領域へのポインタ: BOOL、 BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、または STRUCT

				データタイプのみが許可されています (ビット配列は許可されていません)。. 追加情報は「」を参照してください。 S7 通信用の命令の共通パラメータ

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

ERROR	STATUS (10 進)	説明
0	9	警告: 古い受信データが新しい受信データで上書きされました。
0	11	警告: 優先度のさらに低い優先度クラスで受信データが既に処理されています(受信領域にデータをコピーしているときにエラーが発生する可能性があります)。
0	25	通信が開始しました。ジョブが処理中です。
1	1	通信上の問題が発生しました。 考えられる原因: • 接続記述子がロードされていません(ローカルまたはリモート) • 接続が中断されました(例: ケーブル、CPU オフ、CP が STOP モード) • パートナーへの接続がまだ確立されていません
1	4	データ長やデータタイプに関する受信領域ポインタ RD_i でエラーが発生しました。
1	10	ローカルユーザーメモリにアクセスできません(たとえば、削除された DB にアクセスした場合など)。
1	18	R_ID パラメータの値が、ID パラメータで指定された接続内に既に存在しています(R_ID の値は接続内で一意である必要があります)。
1	19	関連する「USEND」命令がデータを送信する速度が、「URCV」によってデータが受信領域にコピーされる速度よりも速くなっています。
1	20	• 並列ジョブの最大数を超えました。 • 優先度のさらに低い優先度クラスで、現在ジョブが処理されています(最初の呼び出し)。

BSEND: セグメントのデータ送信

説明

「BSEND」命令は、「BRCV」タイプのリモートパートナー命令にデータを送信します。 このタイプのデータ転送では、構成された S7 接続用の他のどの通信命令で可能であるよりも多くのデータを通信パートナー間で転送することができます。 どの場合でも、統合インターフェースおよび SIMATIC NET CP の最大データ容量は、65534 バイト(標準アクセス)または 65535 バイト(最適化されたアクセス)です。

ファンクションの説明

命令ペア「BSEND」および「BRCV」を入力パラメータ R_ID とともに指定します。 R_ID パラメータは、関連する命令において同一である必要があります。

送信ジョブは、命令が呼び出され、制御入力 REQ で立ち上がりエッジが検出された後に有効になります。 この呼び出しの後は、「BSEND」はバックグラウンドで処理されません。これは、データがユーザープログラム内でしか読み出せないことを意味します。

送信されるデータ領域はセグメント化されます。 各セグメントはパートナーに個別に送信されます。各セグメントは、「BRCV」によるセグメントの適用後、パートナーによって確認されます。 データがセグメントに分割されている場合、すべてのセグメントが転送されるまで「BSEND」を複数回呼び出す必要があります。

送信されるデータのデータ領域は、SD_1 で指定します。 データの整合性を保証するため、現在の送信操作が完了しないと、現在使用中の送信領域 SD_1 の部分に書き込むことができません。 これは、ステータスパラメータ DONE の値が「1」に切り替わった場合です。

送信データの長さは、ジョブに基づいて LEN によって指定されます。 LEN = "0"の場合、SD_1 パラメータでアドレス指定されたすべてのデータが送信されます。

制御入力 R に立ち上がりエッジがある場合、現在の送信プロセスはキャンセルされます。

非同期のデータ転送のため、新規のデータ転送は、パートナー命令の呼び出しによって前のデータがフェッチされた場合にしか開始できません。 データが収集されると、ステータスパラメータ「NDR」がパートナー命令「BRCV」内にセットされます。

注記

S7-400 ユーザープログラムの移行

S7-400 CPU は、パラメータ SD_1 をポインタと解釈し、データ領域とは解釈しません。

S7-1500 の場合は、LEN が SD1 の領域を超えてはなりません。 これは、S7-400 では許可されていました。推奨事項: SD_1 パラメータのデータ領域のサイズとして、LEN パラメータには最大サイズを使用します(統合インターフェースでは 65534 バイト)。

パラメータ

次の表に、「BSEND」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ request。立ち上がりエッジでデータ交換を有効にします

R	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ reset。立ち上がりエッジで現在のデータ交換の中止を有効にします。
ID	Input	CONN_PRG	I、Q、M、D、L、P、または定数	パートナー CPU への接続を指定するアドレスパラメータ。
R_ID	Input	CONN_R_ID	I、Q、M、D、L、または定数	命令ペア「BSEND」および「BRCV」を定義するアドレス指定パラメータ。 関連項目: S7 通信用の命令の共通パラメータ
SD_1	InOut	VAR-IANT	I、Q、M、D	送信領域へのポインタ
LEN	InOut	WORD	I、Q、M、D、L	送信されるデータブロックのバイト単位の長さ LEN = "0"の場合、SD_1 によってすべてのデータが送信されます。
DONE	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 警告もエラーもなし。 • 1: エラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ 「ERROR および STATUS パラメータ」の表を参照してください。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

次の表には、ERROR および STATUS パラメータを介して出力できる「BSEND」に固有のエラー情報がすべて含まれています。

ERROR	STA-TUS (10 進)	説明
0	11	警告: 前のジョブがまだ使用中であるため、新しいジョブはアクティブではありません。
0	25	通信が開始しました。 ジョブが処理中です。
1	1	通信上の問題が発生しました。 考えられる原因: • 接続記述子がロードされていません(ローカルまたはリモート) • 接続が中断されました(例: ケーブル、CPU オフ、CP が STOP モード) • パートナーへの接続がまだ確立されていません
1	2	パートナー命令の否定確認。 命令を実行できません。

1	3	ID によって指定された接続上で R_ID が不明であるか、受信ブロックがまだ呼び出されていません。
1	4	• 送信領域ポインタ SD_1 でデータ長、またはデータタイプに関するエラーが発生しました。 • LEN の値が、領域 SD_1 よりも大きくなっています。
1	5	リセット要求が実行されました。
1	6	パートナー命令が DISABLED ステータスにあります(EN_R の値が「0」です)。「BSEND」で BRCV の入力パラメータの整合性チェックも行ってください。
1	7	"最後のデータ転送以降、「BRCV」パートナー命令が呼び出されていません。
1	8	ユーザーメモリ内のリモートオブジェクトへのアクセスが拒否されました。関連する「BRCV」の宛先領域が小さすぎます。 ERROR = 1、STATUS = 4 または ERROR = 1、STATUS = 10 が 「BRCV」の出力パラメータでレポートされています。
1	10	ローカルユーザーメモリにアクセスできません(たとえば、削除された DB にアクセスした場合など)。
1	18	R_ID は既に接続内に存在します。
1	20	• 並列ジョブの最大数を超えました。 • 優先度のさらに低い優先度クラスで、現在ジョブが処理されています(最初の呼び出し)。

BRCV: セグメントのデータ受信

説明

「BRCV」命令は、「BSEND」タイプのリモートパートナー命令からデータを受信します。 R_ID パラメータは、関連する命令において同一である必要があります。

制御入力 EN_R に値「1」を入れて命令が呼び出された後に、この命令はデータ受信準備が完了します(STATUS = 25)。 アクティブなジョブは、EN_R=0 でキャンセルできます。

最大受信領域が、RD_1 によって指定されます。 制御入力 EN_R の値を「1」としてブロックを再度呼び出す前に、現在使用中の RD_1 受信領域の部分を完全に評価すると、データが連続して受信されます。

データセグメントを受信するたびに、確認がパートナー命令に送信されます。 複数のセグメントが存在する場合、すべてのセグメントを受信するまで「BRCV」を複数回呼び出す必要があります。 STATUS = 17 は、非同期のデータ受信を示します。現在受信済みのデータ数は、パラメータ LEN に示されます。 RD_1 パラメータは、処理中に変更してはなりません。

NDR ステータスパラメータの値「1」は、すべてのデータセグメントがエラーなしで受信されたことを示しています。 受信したデータは、EN_R=1 で次の呼び出しが行われるまで変更されません。

パラメータ

次の表に、「BRCV」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ enabled to receive。入力が設定された場合に受信準備完了を信号で通知します。
ID	Input	CONN_PRG	I、Q、M、D、L、P、または定数	パートナー CPU への接続を指定するアドレスパラメータ。
R_ID	Input	CONN_R_ID	I、Q、M、D、L、または定数	命令ペア「BSEND」および「BRCV」を定義するアドレス指定パラメータ。関連項目: S7 通信用の命令の共通パラメータ
RD_1	InOut	VARIANT	I、Q、M、D	受信領域へのポインタ。
LEN	InOut	WORD	I、Q、M、D、L	既に受信したデータのバイト単位の長さ。
NDR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブが正常に完了しました。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • 0: 警告もエラーもなし • 1: エラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。

STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ 「ERROR および STATUS パラメータ」の表を参照してください。

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

次の表には、ERROR および STATUS パラメータを介して出力できる「BRCV」に固有のエラー情報がすべて含まれています。

ER-ROR	STATUS (10 進)	説明
0	17	警告: 命令がデータを非同期的に受信しています。 LEN パラメータは、既に受信したデータの量をバイト単位で表示します。
0	25	通信が開始しました。ジョブが処理中です。
1	1	通信上の問題が発生しました。 考えられる原因: • 接続記述子がロードされていません(ローカルまたはリモート) • 接続が中断されました(例: ケーブル、CPU オフ、CP が STOP モード) • パートナーへの接続がまだ確立されていません
1	2	ファンクションを実行できません(プロトコルエラー)
1	4	データ長やデータタイプに関する受信領域ポインタ RD_1 でエラーが発生しました。 送信データブロックが受信領域よりも長くなっています。
1	5	リセット要求を受信したため、転送が不完全です。
1	8	関連する「BSEND」でアクセスエラーが発生しました。 最後の有効なデータセグメントが送信された後に、ERROR = 1 および STATUS = 4 または ER-ROR = 1 および STATUS = 10 がレポートされます。
1	10	ローカルユーザーメモリにアクセスできません(たとえば、削除された DB にアクセスした場合など)。
1	18	R_ID パラメータの値が、ID パラメータで指定された接続内に既に存在しています(R_ID の値は接続内で一意である必要があります)。
1	20	• 並列ジョブの最大数を超えました。 • 優先度のさらに低い優先度クラスで、現在ジョブが処理されています(最初の呼び出し)。

開放型ユーザー間通信

この章には下記に関する情報が記載されています：

- TSEND_C: イーサネット経由のデータ送信 (S7-1200, S7-1500)
- TRCV_C: イーサネット経由のデータ受信 (S7-1200, S7-1500)
- TMAIL_C: 電子メールの転送 (S7-1200, S7-1500)
- その他 (S7-1200, S7-1500)
- 開放型ユーザー間通信ライブラリ V3.1 と V4.0 の相違点 (S7-1200, S7-1500)

TSEND_C: イーサネット経由のデータ送信

この章には下記に関する情報が記載されています：

- TSEND_C: イーサネット経由のデータ送信 (S7-1200)
- TSEND_C: イーサネット経由のデータ送信 (S7-1500)

TSEND_C: イーサネット経由のデータ送信

説明

「TSEND_C」命令は、TCP または ISO-on-TCP 通信接続を設定および確立します。 接続の設定と確立が終了すると、CPU によって自動的に維持およびモニタされます。 CONNECT パラメータに指定された接続記述子が、通信接続の設定に使用されます。

この命令は非同期的に実行され、次のファンクションを持ちます。

- 通信接続が設定され、確立されます。
  「CONT=1」の場合、通信接続が設定され、確立されます。接続が正常に確立されると、1 つのサイクルに対して DONE パラメータが「1」にセットされます。 CPU が STOP モードに移行すると、既存の接続が終了し、設定されていた接続が削除されます。 接続を再度設定して確立するには、「TSEND_C」を再度実行する必要があります。 実行可能な通信接続の数については、使用している CPU の技術仕様を参照してください。
- 既存の通信接続によるデータの送信:
  DATA パラメータで送信領域を指定します。 これには、アドレスと送信データの長さが含まれます。 DATA パラメータで BOOL または Array of BOOL データタイプのデータ領域を使用しないでください。 DATA パラメータで単にシンボリックな値を使用する場合、LEN パラメータの値は「0」である必要があります。
- 送信ジョブは、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。 LEN パラメータを使用して、送信ジョブで送信する最大バイト数を指定します。 データの送信(REQ パラメータでの信号立ち上がりエッジ)時に、接続を確立または維持するには、CONT パラメータの値は「1」である必要があります。 送信データは、送信ジョブが完了するまで編集しないでください。 送信ジョブが正常に実行された場合、DONE パラメータが「1」にセットされます。 DONE パラメータのシグナル状態が「1」であっても、送信データが通信パートナーに読み取られているとは限りません。
- 通信接続の終了:

  進行中のデータ送信がまだ完了していない場合でも、CONT パラメータが「0」にセットされると、通信接続が終了します。 「TSEND_C」に対して既に設定した接続を使用している場合、これは適用されません。

COM_RST パラメータを「1」にセットすると、現在確立されている接続または現在のデータ伝送をいつでもリセットできます。 これによって、既存の通信接続が終了し、新しい通信が確立されます。再実行時にデータを転送中の場合、データが失われる恐れがあります。

実行後(DONE = 1)に「TSEND_C」を再度有効にするには、REQ = 0 として命令を 1 回呼び出します。

パラメータ

以下の表に、「TSEND_C」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	信号立ち上がりエッジで送信ジョブを開始します。
CONT	Input	BOOL	I、Q、M、D、L	通信接続を制御します。 • 0: 通信接続の切断 • 1: 通信接続を確立し、維持します データの送信(REQ パラメータでの信号立ち上がりエッジ)時に、接続を確立また

				は維持するには、CONT パラメータの値は TRUE である必要があります。
LEN	Input	UINT	I、Q、M、D、L、または定数	ジョブで送信する最大バイト数。 DATA パラメータで単にシンボリックな値を使用する場合、LEN パラメータの値は「0」である必要があります。
CONNECT	InOut	TCON_Par-am	D	接続記述子へのポインタ 関連項目: TCON_Param に従う構造の接続パラメータ
DATA	InOut	VARIANT	I、Q、M、D、L	アドレスと送信データの長さの入っている送信領域へのポインタ(最大長: 8192 バイト)。
COM_RST	InOut	BOOL	I、Q、M、D、L	命令を再起動します。 • 0: 対象外 • 1: 既存の接続を終了させ、新しい接続を確立させる命令の完全な再起動
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ BUSY、DONE および ERROR

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、ジョブが正常に実行されたかどうかをチェックできます。 「TSEND_C」の実行中にエラーが発生すると、ERROR パラメータが設定されます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ER-ROR	説明
1	-	-	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。

0	0	1	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータに明記されています。
0	0	0	新しいジョブが割り当てられませんでした。

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	ジョブがエラーなしで完了済み。
0	0001	接続の確立が完了しています。
0	0003	接続の終了が完了しています。
0	7000	ジョブ処理が行われていません。
0	7001	• ジョブの実行を開始します。 • 接続の確立 • 接続パートナーを待っています。
0	7002	データが送信されました。
0	7003	接続を終了中です。
0	7004	接続が確立され、モニタされています。ジョブ処理は行われていません。
1	80A0	エラーコード 80A1 および 80A2 のグループエラーです。
1	80A1	• 接続またはポートが既にユーザーによって使用中です。 • 通信エラー: ○ 指定された通信がまだ確立されていません。 ○ 指定された接続を終了中です。 この接続による転送は行えません。 ○ インターフェースを再初期化中です。
1	80A2	ローカルまたはリモートポートがシステムによって使用中です。
1	80A3	存在していない接続を終了しようとしています。
1	80A4	接続のリモートエンドポイントの IP アドレスが無効です。つまり、ローカルパートナーの IP アドレスと一致しています。
1	80A7	通信エラー: 送信ジョブの完了前に、COM_RST = 1 で命令を呼び出しました。
1	80AA	別のブロックによって同じ接続 ID を持つ接続が現在確立されています。 REQ パラメータの新しい立ち上がりエッジでジョブを繰り返します。
1	80B2	CONNECT パラメータが、属性「ロードメモリにのみ保存」を使用して生成されたデータブロックをポイントしています。
1	80B3	パラメータの割り当てに整合性がありません。 エラーコード 80A0～80A2、80A4、80B4～80B9 のグループエラーです。
1	80B4	ISO-on-TCP プロトコルの種類(connection_type = B#16#12)を使用するときに、パッシブ接続の確立(active_est = FALSE)について、次の条件の一方または両方に違反しました。 "local_tsap_id_len >= B#16#02"および/または"local_tsap_id[1] = B#16#E0"。
1	80B5	接続タイプ 13 = UDP の場合、パッシブ接続の確立以外は許可されていません。

1	80B6	接続記述子のデータブロックの connection_type パラメータにパラメータ割り当てエラーがあります。
1	80B7	接続記述のデータブロックの次のパラメータのいずれかにエラーがあります: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	LEN パラメータが最大許容値よりも大きくなっています。
1	8086	CONNECT パラメータ内の ID パラメータが許容範囲外です。
1	8087	最大接続数に達しました。これ以上の接続は行えません。
1	8088	LEN パラメータの値が、DATA パラメータで設定された受信領域と一致していません。
1	8089	CONNECT パラメータがデータブロックをポイントしていません。
1	8091	最大ネスト深さを超えています。
1	809A	CONNECT パラメータが、接続記述子の長さと一致しないフィールドをポイントしています。
1	809B	接続記述子のローカルデバイスの ID が、CPU と一致していません。
1	80C3	• すべての接続リソースが使用中です。 • この ID を持つブロックは、別の優先度グループで既に処理中です。
1	80C4	一時的な通信エラー: • 現時点では、接続を確立できません。 • インターフェースが新しいパラメータを受信中であるか、接続が確立中です。 • 設定した接続が現在「TDISCON」命令によって削除されています。 • 使用された接続が、COM_RST= 1 での呼び出しによって終了されています。
1	80C6	リモートネットワークエラー。 リモートパートナーに到達できません。
1	8722	パラメータ CONNECT: ソース領域が無効です。 領域が DB 内に存在していません。
1	873A	パラメータ CONNECT : 接続記述子にアクセスできません(DB が存在していないなど)。
1	877F	パラメータ CONNECT : 内部エラー
1	8822	パラメータ DATA : ソース領域が無効です。領域が DB 内に存在していません。
1	8824	パラメータ DATA : VARIANT ポインタで領域エラーが発生しました。
1	8832	パラメータ DATA : DB 番号が大きすぎます。
1	883A	パラメータ CONNECT : 指定された接続データにアクセスできません(たとえば、DB が存在しないため)。
1	887F	パラメータ DATA : 内部エラー。たとえば、VARIANT 参照が無効です。
1	893A	パラメータ DATA : 送信領域にアクセスできません(たとえば、DB が存在しないため)。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

注記

「TCON」、「TSEND」、「T_DIAG」および「TDISCON」命令のエラーメッセージ

内部的には、「TSEND_C」命令は、命令「TCON」、「TSEND」、「T_DIAG」、「T_RESET」および「TDISCON」を使用します。 これらの命令のエラーメッセージは、ステータスパラメータで出力されることもあります。 エラーコードの意味は、対応する命令に記載されています。 内部的に使用されている異なる意味を持つ命令に対して同一のエラーコードが発生した場合は、「TSEND_C」のインスタンスデータブロックを使用してどの命令がエラーを出力したかを識別できます。

TSEND_C: イーサネット経由のデータ送信

説明

「TSEND_C」命令は、通信接続を設定および確立します。 接続の設定と確立が終了すると、CPU によって自動的に維持およびモニタされます。

この命令は非同期的に実行され、次のファンクションを持ちます。

- 通信接続の設定および確立
- 既存の通信接続を経由したデータ送信
- 通信接続の終了またはリセット

内部的には、命令「TSEND_C」は、通信命令「TCON」、「TSEND」、「T_DIAG」、「T_RESET」および「TDISCON」を使用します。

通信接続の設定および確立

CONT=1 の場合、通信接続が設定され、確立されます。実行可能な通信接続の数については、使用している CPU の技術仕様を参照してください。 CONNECT パラメータに指定された接続記述子が、通信接続の設定に使用されます。 次の接続タイプを使用できます。

- プログラミングされた接続(「TCON」を経由した接続の構造):
  - TCP / UDP: TCON_IP_v4 システムデータタイプによる接続記述子
  - ISO-on-TCP: TCON_IP_RFC システムデータタイプによる接続記述子
  - ISO: TCON_ISOnative システムデータタイプによる接続記述子(CP1543-1 の場合のみ)
- 設定した接続
  - 既存の接続を TCON_Configured システムデータタイプに指定します。

CPU が STOP モードに移行すると、既存の接続が終了し、設定されていた接続が削除されます。 接続を再度設定して確立するには、「TSEND_C」を再度実行する必要があります。

既存の通信接続を経由したデータ送信

送信ジョブは、REQ パラメータで信号立ち上がり工ッジが検出されたときに実行されます。 前述の通り、先に通信接続が確立されます。

DATA パラメータで送信領域を指定します。 これには、アドレスと送信データの長さが含まれます。 DATA パラメータで BOOL または Array of BOOL データタイプのデータ領域を使用しないでください。 LEN パラメータを使用して、送信ジョブで送信する最大バイト数を指定します。 DATA パラメータで最適化されたアクセスのある送信領域を使用する場合、LEN パラメータの値は「0」である必要があります。

送信データは、送信ジョブが完了するまで編集しないでください。

通信接続の終了およびリセット

進行中のデータ送信がまだ完了していない場合でも、CONT パラメータが「0」にセットされると、通信接続が終了します。 「TSEND_C」に対して設定した接続を使用している場合、これは適用されません。

パラメータ COM_RST を「1」にセットすることによって、接続をいつでもリセットできます。 これによって、既存の通信接続が終了し、新しい通信が確立されます。 この時にデータを転送中の場合、データが失われる恐れがあります。

パラメータ

以下の表に、「TSEND_C」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数	信号立ち上がり工ッジで送信ジョブを開始します。
CONT	Input	BOOL	I、Q、M、D、L	通信接続を制御します。 • 0: 通信接続を切断します。 • 1: 通信接続を確立し、維持します。
LEN	Input	UDINT	I、Q、M、D、L、または定数	オプションパラメータ(非表示) ジョブで送信する最大バイト数。 DATA パラメータで最適化されたアクセスのある送信領域を使用する場合、LEN パラメータでは値「0」が使用される必要があります。
CONNECT	InOut	VARIANT	D	接続記述子の構造へのポインタ: • プログラミングされた接続: ○ TCP または UDP の場合、TCON_IP_v4 システムデータタイプを使用します。 詳細については、次を参照してください: TCON_IP_v4 に従う構造の接続パラメータ ○ ISO-on-TCP の場合、TCON_IP_RFC システムデータタイプを使用します。 詳細については、次を参照してください: TCON_IP_RFC に従う構造の接続パラメータ ○ ISO の場合、TCON_ISOnative システムデータタイプを使用します(CP1543-1 の場合のみ)。 詳細については、命令「TCON」を参照してください。 • 設定した接続: ○ 既存の接続の場合、TCON_Configured システムデータタイプを使用します。 説明については、下の「設定した接続のシステムデータタイプ」を参照してください
DATA	InOut	VARIANT	I、Q、M、D、L	アドレスと送信データの長さの入っている送信領域へのポインタ
ADDR	InOut	VARIANT	D	オプションパラメータ(非表示) 宛先のアドレスへのポインタ。

COM_RST	InOut	BOOL	I、Q、M、D、L	オプションパラメータ(非表示) 接続のリセット: • 0: 対象外 • 1: 既存の接続がリセットされます。 COM_RST パラメータは、「TSEND_C」命令による評価の後にリセットされるため、スタティックに相互接続してはなりません。
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: 送信ジョブがまだ開始されていないか、または実行中です。 • 1: 送信ジョブがエラーなしで実行されました。 この状態は、1 つのサイクルに対してのみ常時されます。 処理中(接続の確立、送信、接続の終了)に中間ステップが正常に完了した場合、および「TSEND_C」の実行が正常に完了した場合、出力パラメータ DONE が設定されます。
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: 送信ジョブがまだ開始されていないか、または既に完了しています。 • 1: 送信ジョブがまだ完了していません。 新しい送信ジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: 接続確立、データ転送、または接続終了時にエラーが発生しました。 「TSEND_C」命令のエラーまたは内部で使用される通信命令のために、出力パラメータ ERROR が設定されることがあります。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス(「ERROR および STATUS パラメータ」の説明を参照)。

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

REQ、CONT および COM_RST パラメータ

パラメータ CONT は、REQ パラメータに関係なく、「TSEND_C」命令の接続の確立を制御します。CONT パラメータの動作は、プログラミングされた接続が使用されるか、または設定した接続が使用されるかに、部分的に依存します。

- CONT = "0"の場合: データは送信されません(プログラミングされた接続が使用されるか、または設定した接続が使用されるかに無関係)。
- CONT = "0"を"1"に変更する場合:
  - プログラミングされた接続を使用する場合、「TCON」で確立されます。
  - 設定した接続を使用する場合、「T_DIAG」でチェックされます。

- CONT = "1"の場合:
  - データが送信されない限り(REQ="0")、接続は「T_DIAG」でチェックされます。
  - 内部で使用される通信命令が、接続エンドポイントが存在しないことを信号で示した場合、接続が「TCON」で自動的に再確立されます。
- CONT = "1"を"0"に変更する場合:
  - プログラミングされた接続を使用する場合、「TDISCON」で終了されます。
  - 設定した接続を使用する場合、「T_RESET」でリセットされます。

パラメータ COM_RST は、「0」から「1」に切り替わると、接続をリセットします。

- 接続が確立された場合、「T_RESET」でリセットされます(プログラミングされた接続が使用されるか、または設定した接続が使用されるかに無関係)。
- 接続が確立されない場合、パラメータの設定は影響を及ぼしません。

REQ および COM_RST パラメータが影響を及ぼすのは、CONT が「1」にセットされている場合のみです。 次の表に、 REQ、CONT 、 COM_RST パラメータの関係を示します。

REQ	CONT	COM_RST	命令のステータス	説明
対象外	0	対象外	まだ実行されていません	動作中のジョブはありません(STATUS = 7000)。
対象外	0	対象外	初期化	接続を終了中です。 命令をリセット中です。
対象外	0 > 1	対象外	接続の確立	接続を確立中です。 データはまだ転送されていません。
0	1	0	接続が確立されました。	接続が確立され、命令「T_DIAG」でモニタされます。
対象外	1	0 > 1	接続が確立されました。	接続が「T_RESET」によって一時的に割り込まれ、リセットされます。
0 > 1	1	0	接続が確立されました。	命令が送信を開始します。
対象外	1	0 > 1	データを送信中です	データ伝送が割り込みされます。 接続をリセットしています。

設定した接続のシステムデータタイプ

CONNECT パラメータで設定した接続の場合、接続の記述には、TCON_Configured に従って以下の構造を使用します。

Byte (バイト)	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceID	HW_ANY	-	ローカルインターフェースのハードウェア識別子(値の範囲: 0～65535)。
2 … 3	ID	CONN_OUC	-	接続への参照(値の範囲: 1～4095)。 既存の接続の接続 ID を入力します。
4	Connection-Type	BYTE	-	接続タイプ

				設定した接続に対して 254 (10 進数)を選択します。

BUSY、DONE および ERROR パラメータ

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、送信ジョブが正常に実行されたかどうかをチェックできます。 「TSEND_C」の実行中にエラーが発生すると、ERROR パラメータが設定されます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

DONE	BUSY	ER-ROR	説明
0	0	0	命令はまだ実行されていません(REQ パラメータに立ち上がりエッジなし)。
0	1	0	命令を実行中であり、内部で使用される通信命令を呼び出します。
1	0	0	送信ジョブが正常に完了しました。「0000」が、ステータスパラメータに出力されます。 DONE = "1"が、1 つのサイクルに対してのみ表示されます。
0	0	1	命令の実行または処理中の中間ステップが、エラーで終了されました。内部で使用される通信命令のために、その後のエラーが発生した場合は、処理中に最初に発生したエラーが表示されます。 この状態は、1 つのサイクルに対してのみ常時されます。

ERROR および STATUS パラメータ

ERROR	STA-TUS* (W#16#...)	説明
0	0000	送信ジョブがエラーなしで実行されました。
0	0001	通信接続が確立されました。
0	0003	通信接続が終了しました。
0	7000	実行中の送信ジョブはありません。接続された通信接続はありません。
0	7001	接続を確立するための最初の呼び出し。
0	7002	接続を確立するための 2 回目の呼び出し
0	7003	通信接続を終了中です。
0	7004	通信接続が確立されており、モニタされています。 有効な送信ジョブの実行はありません。
0	7005	データ伝送が進行中です。
1	80A1	• 接続またはポートが既にユーザーによって使用中です。 • 通信エラー: ○ 指定された通信がまだ確立されていません。 ○ 指定された接続を終了中です。 この接続による転送は行えません。 ○ インターフェースを再初期化中です。

1	80A3	ネストされた「T_DIAG」命令によって、接続が終了したことがレポートされました。
1	80A4	接続のリモートエンドポイントの IP アドレスが無効であるか、またはローカルパートナーの IP アドレスと一致します。
1	80A7	通信エラー: 送信ジョブの完了前に、COM_RST = 1 で命令を呼び出しました。
1	80AA	別のブロックによって同じ接続 ID を持つ接続が現在確立されています。REQ パラメータの新しい立ち上がりエッジでジョブを繰り返します。
1	80B4	ISO-on-TCP プロトコルの種類(connection_type = B#16#12)を使用するときに、パッシブ接続の確立(active_est = FALSE)について、次の条件の一方または両方に違反しました。 • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0
1	80B5	接続タイプ 13 = UDP の場合、パッシブ接続の確立以外は許可されていません。
1	80B6	接続記述子のデータブロックの connection_type パラメータにパラメータ割り当てエラーがあります。
1	80B7	接続記述のデータブロックの次のパラメータのいずれかにエラーがあります: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	8085	LEN パラメータが最大許容値よりも大きくなっています。
1	8086	CONNECT パラメータ内の ID パラメータが許容範囲外です。
1	8087	最大接続数に達しました。これ以上の接続は行えません。
1	8088	LEN パラメータの値が、DATA パラメータで設定された受信領域と一致していません。
1	8089	CONNECT パラメータがデータブロックをポイントしていません。
1	8091	最大ネスト深さを超えています。
1	809A	CONNECT パラメータが、接続記述子の長さと一致しないフィールドをポイントしています。
1	809B	InterfaceID が無効です。 ゼロであるか、ローカル CPU インターフェースまたは CP をポイントしていません。
1	80C3	• すべての接続リソースが使用中です。 • この ID を持つブロックは、別の優先度グループで既に処理中です。
1	80C4	一時的な通信エラー: • 現時点では、接続を確立できません。 • インターフェースが新しいパラメータを受信中であるか、接続が確立中です。 • 設定した接続が、現在「TDISCON」命令によって削除されています。 • 使用された接続が、COM_RST = 1 で呼び出されたことで終了しています。
1	80C6	リモートネットワークエラー。 リモートパートナーに到達できません。
1	8722	パラメータ CONNECT: ソース領域が無効です。 領域が DB 内に存在していません。
1	873A	パラメータ CONNECT : 接続記述子にアクセスできません(DB が使用できないなど)。
1	877F	パラメータ CONNECT : 内部エラー

1	8822	パラメータ DATA： ソース領域が無効です。領域が DB 内に存在していません。
1	8824	パラメータ DATA： VARIANT ポインタで領域エラーが発生しました。
1	8832	パラメータ DATA： DB 番号が大きすぎます。
1	883A	パラメータ CONNECT： 指定された接続データにアクセスできません(たとえば、DB が存在しないため)。
1	887F	パラメータ DATA： 内部エラー。たとえば、VARIANT 参照が無効です。
1	893A	パラメータ DATA： 送信領域にアクセスできません(たとえば、DB が存在しないため)。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

注記

「TCON」、「TSEND」、「T_DIAG」、「T_RESET」および「TDISCON」命令のエラーメッセージ

内部的には、「TSEND_C」命令は、命令「TCON」、「TSEND」、「T_DIAG」、「T_RESET」および「TDISCON」を使用します。 これらの命令のエラーメッセージは、STATUS パラメータで出力されることもあります。エラーコードの意味は、対応する命令に記載されています。内部的に使用されている異なる意味を持つ命令に対して同一のエラーコードが発生した場合は、「TSEND_C」のインスタンスデータブロックを使用してどの命令がエラーを出力したかを識別できます。

TRCV_C: イーサネット経由のデータ受信

この章には下記に関する情報が記載されています：

- TRCV_C: イーサネット経由のデータ受信送信 (S7-1200)
- TRCV_C: イーサネット経由のデータ受信送信 (S7-1500)

TRCV_C: イーサネット経由のデータ受信送信

説明

「TRCV_C」命令は非同期的に実行され、次のファンクションを持ちます。

1. **通信接続の設定および確立:**

   "TRCV_C」は、TCP または ISO-on-TCP 通信接続を設定および確立します。 接続の設定と確立が終了すると、CPU によって自動的に維持およびモニタされます。

   CONNECT パラメータに指定された接続記述子が、通信接続の設定に使用されます。 接続を確立するには、CONT パラメータを値「1」にセットする必要があります。 接続が正常に確立されると、DONE パラメータが「1」にセットされます。

   CPU が STOP モードに移行すると、既存の接続が終了し、設定されていた接続が削除されます。 接続を再度設定して確立するには、「TRCV_C」を再度実行する必要があります。

   実行可能な通信接続の数については、使用している CPU の技術仕様を参照してください。
2. **既存の通信接続によるデータの受信:**
   EN_R パラメータが値「1」に設定されている場合、データの受信が可能です。 データの受信(EN_R パラメータでの信号立ち上がりエッジ)時に、接続を確立または維持するには、CONT パラメータの値は TRUE である必要があります。

   受信データは、受信領域に入力されます。 使用中のプロトコルの種類に応じて、LEN パラメータ(LEN <> 0 の場合)または DATA パラメータの長さ情報(LEN = 0 の場合)のいずれかを使用して、受信領域の長さを指定します。 DATA パラメータで単にシンボリックな値を使用する場合、LEN パラメータの値は「0」である必要があります。

   データが正常に受信されると、DONE パラメータのシグナル状態は「1」です。 データ転送で エラーが発生した場合、DONE パラメータが「0」にセットされます。
3. **通信接続の終了:**

   CONT パラメータが「0」にセットされると、すぐに通信接続が終了します。

COM_RST パラメータが設定されると、TRCV_C が再度実行されます。 これによって、既存の通信接続が終了し、新しい通信が確立されます。 再実行時にデータを受信中の場合、データが失われる恐れがあります。

TRCV_C の受信モード

次の表に、受信データがどのように受信領域に入力されるかを示します。

プロトコルの種類	受信領域内のデータの可用性	接続記述子の connection_type パラメータ	パラメータ LEN
TCP (アドホックモード)	データはすぐに使用可能です。	16 進数値: B#16#11 整数値: 17	0
TCP (指定された長さのデータの受信)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#11 整数値: 17	1～8192

ISO on TCP (メッセージ指向のデータ転送)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#12 整数値: 18	• 1～1452、CP を使用する場合。 • 1～8192、CP を使用しない場合。

TCP (アドホックモード)

アドホックモードは、TCP プロトコル種類でのみ使用可能です。「TRCV」命令で長さが動的なデータを受信するには、アドホックモードを使用します。

アドホックモードは、値「0」を LEN パラメータに割り当てることで設定できます。 アドホックモードを使用すると、データブロックに対して標準アクセスですべてのデータタイプを使用できます。 最適化されたアクセスのあるデータブロックには、ARRAY of BYTE または長さが 8 ビットのデータタイプのみを使用できます(CHAR、USINT、SINT など)。 実際の受信データ長は、RCVD_LEN パラメータに出力されます。

TCP (指定された長さのデータの受信)

LEN パラメータの値を使用し、データ受信の長さを指定します。 LEN パラメータで指定された長さのデータが完全に受信されるまで、データの受信は完了しません。 受信領域でデータが使用可能になるのは、そのときだけです(DATA パラメータ)。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

ISO on TCP (メッセージ指向のデータ転送)

プロトコルのバリエーション ISO on TCP を使用する接続を経由して、メッセージブロック全体が送信されます。これらは受信によってそのように認識されます。 ISO on TCP を使用している場合、メッセージブロックが完全に受信されると直ちに、「TRCV_C」がデータの受信を信号で示します。 受信領域は、LEN および DATA パラメータによって定義されます。 送信データに対して受信バッファ(DATA パラメータ)が小さすぎる場合、「TRCV_C」がエラーを信号で示します。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

パラメータ

以下の表に、「TRCV_C」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L	受信可能です。
CONT	Input	BOOL	I、Q、M、D、L	通信接続を制御します。 • 0: 通信接続の切断 • 1: 通信接続を確立し、維持します データの受信(EN_R パラメータでの信号立ち上がりエッジ)時に、接続を確立または維持するには、CONT パラメータの値は TRUE である必要があります。
LEN	Input	UINT	I、Q、M、D、L、または定数	受信するデータの最大長(最大値: 8192 バイト)。 DATA パラメータで単にシンボリックな値を使用する場合、LEN パラメータの値は「0」である必要があります。

CONNECT	InOut	TCON_Par-am	D	接続記述子へのポインタ 関連項目: TCON_Param に従う構造の接続パラメータ
DATA	InOut	VARIANT	I、Q、M、D、L	受信領域へのポインタ
COM_RST	InOut	BOOL	I、Q、M、D、L	命令を再起動します。 • 0: 対象外 • 1: 既存の接続を終了させる命令の完全な再起動
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス
RCVD_LEN	Output	UDINT	I、Q、M、D、L	実際に受信されたバイト単位のデータ量

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ BUSY、DONE および ERROR

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、ジョブが正常に実行されたかどうかをチェックできます。 「TRCV_C」の実行中にエラーが発生すると、ERROR パラメータが設定されます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ERROR	説明
1	-	-	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 STATUS パラメータにエラーの原因が出力されます。
0	0	0	新しいジョブが割り当てられませんでした。

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	ジョブがエラーなしで完了済み。
0	0001	接続の確立が完了しています。
0	0003	接続の終了が完了しています。
0	7000	ジョブ処理が行われていません。
0	7001	• ジョブの実行を開始します。 • 接続の確立 • 接続パートナーを待っています。
0	7002	データを受信中です。
0	7003	接続を終了中です。
0	7004	• 接続が確立され、モニタされています。 • ジョブ処理が行われていません。
0	7006	現在データを受信中です。
1	8085	• LEN パラメータが最大許容値よりも大きくなっています。 • 最初の呼び出しの後に、LEN または DATA パラメータの値が変更されました。
1	8086	ID パラメータが許容範囲外です。
1	8087	最大接続数に達しました。これ以上の接続は行えません。
1	8088	LEN パラメータの値が、DATA パラメータで設定された受信領域と一致していません。
1	8089	CONNECT パラメータがデータブロックをポイントしていません。
1	8091	最大ネスト深さを超えています。
1	809A	CONNECT パラメータが、接続記述子の長さと一致しないフィールドをポイントしています。
1	809B	接続記述子のローカルデバイスの ID (local_device_id)が、CPU と一致していません。
1	80A0	エラーコード W#16#80A1 および W#16#80A2 のグループエラーです。
1	80A1	• 接続またはポートが既にユーザーによって使用中です。 • 通信エラー: ○ 指定された通信がまだ確立されていません。 ○ 指定された接続を終了中です。 この接続による転送は行えません。 ○ インターフェースを再初期化中です。
1	80A2	ローカルまたはリモートポートがシステムによって使用中です。
1	80A3	• 既存の接続を再確立しようとしています。 • 存在していない接続を終了しようとしています。
1	80A4	接続のリモートエンドポイントの IP アドレスが無効です。つまり、ローカルパートナーの IP アドレスと一致しています。
1	80A7	通信エラー: 送信ジョブの完了前に、COM_RST = 1 で命令を呼び出しました。

1	80B2	CONNECT パラメータが、属性「ロードメモリにのみ保存」を使用して生成されたデータブロックをポイントしています。
1	80B3	パラメータの割り当てに整合性がありません。 エラーコード W#16#80A0 ~ W#16#80A2、W#16#80A4、W#16#80B4 ~ W#16#80B9 のグループエラーです。
1	80B4	ISO-on-TCP プロトコルの種類(connection_type = B#16#12)を使用するときに、パッシブ接続の確立(active_est = FALSE)について、次の条件の一方または両方に違反しました。 "local_tsap_id_len >= B#16#02"および/または"local_tsap_id[1] = B#16#E0"。
1	80B5	接続タイプ 13 = UDP の場合、パッシブ接続の確立以外は許可されていません。
1	80B6	接続記述子のデータブロックの connection_type パラメータにパラメータ割り当てエラーがあります。
1	80B7	接続記述のデータブロックの次のパラメータのいずれかにエラーがあります: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• すべての接続リソースが使用中です。 • この ID を持つブロックは、別の優先度グループで既に処理中です。
1	80C4	一時的な通信エラー: • 現時点では、接続を確立できません。 • インターフェースが新しいパラメータを受信中であるか、接続が確立中です。 • 設定した接続が現在「TDISCON」命令によって削除されています。 • 使用された接続が、COM_RST= 1 での呼び出しによって終了されています。
1	80C6	リモートパートナーに到達できません(ネットワークエラー)。
1	8722	CONNECT パラメータでエラーが発生しました。 ソース領域(データブロックで宣言されていない領域)が無効です。
1	873A	CONNECT パラメータでエラーが発生しました。 接続記述へのアクセスが行えません(データブロックにアクセスできません)。
1	877F	CONNECT パラメータでエラーが発生しました。 内部エラー
1	8922	パラメータ DATA： ターゲット領域が無効です。領域が DB 内に存在していません。
1	8924	パラメータ DATA： VARIANT ポインタで領域エラーが発生しました。
1	8932	パラメータ DATA： DB 番号が大きすぎます。
1	893A	パラメータ CONNECT： 指定された接続データにアクセスできません(たとえば、DB が存在しないため)。
1	897F	パラメータ DATA： 内部エラー。たとえば、VARIANT 参照が無効です。
1	8A3A	パラメータ DATA： たとえば、データブロックが存在しないため、データ領域にアクセスできません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

注記

命令「TCON」、「TRCV」および「TDISCON」のエラーメッセージ

内部的には、TRV_C 命令は「TCON」、「TRCV」および「TDISCON」命令を使用します。 これらの命令のエラーメッセージは、それぞれの説明に含まれています。

TRCV_C: イーサネット経由のデータ受信送信

説明

「TRCV_C」命令は非同期的に実行され、シーケンス内の次のファンクションを実行します。

- 通信接続の設定および確立
- 既存の通信接続を経由したデータ受信
- 通信接続の終了またはリセット

内部的には、命令「TRCV_C」は、通信命令「TCON」、「TRCV」、「T_DIAG」、「T_RESET」および「TDISCON」を使用します。

通信接続の設定および確立

CONT=1 の場合、通信接続が設定され、確立されます。実行可能な通信接続の数については、使用している CPU の技術仕様を参照してください。 CONNECT パラメータに指定された接続記述子が、通信接続の設定に使用されます。 次の接続タイプを使用できます。

- プログラミングされた接続(「TCON」を経由した接続の構造):
  - TCP / UDP: TCON_IP_v4 システムデータタイプによる接続記述子
  - ISO-on-TCP: TCON_IP_RFC システムデータタイプによる接続記述子
  - ISO: TCON_ISOnative システムデータタイプによる接続記述子(CP1543-1 の場合のみ)
- 設定した接続
  - 既存の接続を TCON_Configured システムデータタイプに指定します。

CPU が STOP モードに移行すると、既存の接続が終了し、設定されていた接続が削除されます。 接続を再度設定して確立するには、「TRCV_C」を再度実行する必要があります。

既存の通信接続を経由したデータ受信

EN_R パラメータが値「1」にセットされると、データの受信が可能になります。 受信データは、受信領域に入力されます。 使用されているプロトコル変数に応じて、受信エリアの長さを LEN パラメータ(LEN <> 0 の場合)、または DATA パラメータの長さの情報(LEN = 0 の場合)で指定します。 DATA パラメータで単にシンボリックな値を使用する場合、LEN パラメータの値は「0」である必要があります。

TRCV_C の受信モード:

- **TCP (アドホックモード)**

  アドホックモードは、TCP プロトコル種類でのみ使用可能です。「TRCV_C」命令で長さが動的なデータを受信するには、アドホックモードを使用します。

  アドホックモードは、値「1」を ADHOC パラメータに割り当てることで設定できます。 アドホックモードを使用すると、データブロックに対して標準アクセスですべてのデータタイプを使用できます。 最適化されたアクセスのあるデータブロックには、ARRAY of BYTE または長さが 8 ビットのデータタイプのみを使用できます(CHAR、USINT、SINT など)。 実際の受信データ長は、RCVD_LEN パラメータに出力されます。

- **TCP (指定された長さのデータの受信)**

  指定された長さのデータの受信の場合、ADHOC パラメータに値「0」を割り当てます。 アドホックモードが無効の場合、LEN パラメータで指定された長さのデータが完全に受信されるまで、データの受信は完了しません。 受信領域でデータが使用可能になるのは、そのときだけです(DATA パラメータ)。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

- **ISO on TCP (メッセージ指向のデータ転送)**

  プロトコルのバリエーション ISO on TCP を使用する接続を経由したメッセージブロック全体が送信されます。これらは受信によってそのように認識されます。 受信領域は、LEN および DATA パラメータによって定義されます。 送信データに対して受信バッファ(DATA パラメータ)が小さすぎる場合、「TRCV_C」がエラーを信号で示します。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

次の表に、受信データがどのように受信領域に入力されるかを示します。

プロトコルの種類	受信領域内のデータの可用性	接続記述子の connection_type パラメータ	LEN パラメータ
TCP (アドホックモード)	データはすぐに使用可能です。	16 進数値: B#16#11 整数値: 17	1 から最大長(CPU によって異なる)
TCP (指定された長さのデータの受信)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#11 整数値: 17	1～8192
ISO on TCP (メッセージ指向のデータ転送)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#12 整数値: 18	1～8192

通信接続の終了

進行中のデータ送信がまだ完了していない場合でも、CONT パラメータが「0」にセットされると、通信接続が終了します。 ただし、設定した接続を使用している場合、これは適用されません。

パラメータ COM_RST を「1」にセットすることによって、接続をいつでもリセットできます。 これによって、既存の通信接続が終了し、新しい通信が確立されます。 この時にデータを転送中の場合、データが失われる恐れがあります。

パラメータ

以下の表に、「TRCV_C」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、T、C、または定数	受信可能です。
CONT	Input	BOOL	I、Q、M、D、L	通信接続を制御します。 • 0: 通信接続を切断します。 • 1: 通信接続を確立し、データ受信後も維持します。
LEN	Input	UDINT	I、Q、M、D、L、または定数	受信するデータの最大長。 DATA パラメータで最適化されたアクセスのある受信領域を使用する場合、LEN パラメータでは値「0」が使用される必要があります。
ADHOC	Input	BOOL	I、Q、M、D、L、または定数	オプションパラメータ(非表示)

				TCP プロトコルのバリエーションの場合は、アドホックモードを使用します。
CONNECT	InOut	VARIANT	D	接続記述子へのポインタ • プログラミングされた接続: ○ TCP または UDP の場合、構造体 TCON_IP_v4 を使用します。 詳細については、次を参照してください: TCON_IP_v4 に従う構造の接続パラメータ ○ ISO-on-TCP の場合、構造体 TCON_IP_RFC を使用します。 詳細については、次を参照してください: TCON_IP_RFC に従う構造の接続パラメータ ○ ISO の場合、構造体 TCON_ISO-native を使用します(CP1543-1 のみ) 詳細については、「TCON」命令 "TCON_ISOnative に基づいた接続記述子の構造」を参照してください。 • 設定した接続: ○ 既存の接続の場合、TCON_Configured システムデータタイプを使用します。 説明については、下の「設定した接続のシステムデータタイプ」を参照してください。
DATA	InOut	VARIANT	I、Q、M、D、L	受信領域へのポインタ。
ADDR	InOut	VARIANT	D	オプションパラメータ(非表示) 接続タイプが UDP の送信元アドレスへのポインタ。
COM_RST	InOut	BOOL	I、Q、M、D、L	オプションパラメータ(非表示) 接続のリセット: • 0: 対象外 • 1: 既存の接続がリセットされます。 COM_RST パラメータは、「TRCV_C」命令による評価の後にリセットされるため、スタティックに相互接続してはなりません。
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: 受信がまだ開始されていないか、または実行中です。 • 1: 受信がエラーなしで実行されました。 この状態は、1 つのサイクルに対してのみ常時されます。

				処理中(接続の確立、受信、接続の終了)に中間ステップが正常に完了した場合、および「TRCV_C」の実行が正常に完了した場合、出力パラメータ DONE が設定されます。
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: 受信がまだ開始されていないか、既に完了しています。 • 1: 受信がまだ完了していません。 新しい送信ジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: 接続確立、データ受信、または接続終了時にエラーが発生しました。 「TRCV_C」命令のエラーまたは内部で使用される通信命令のために、出力パラメータ ERROR が設定されることがあります。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス
RCVD_LEN	Output	UDINT	I、Q、M、D、L	実際に受信されたバイト単位のデータ量

有効なデータタイプに関する追加情報については、「**有効なデータタイプの概要**」を参照してください。

EN_R、CONT および COM_RST パラメータ

パラメータ CONT は、EN_R パラメータに関係なく、「TRCV_C」命令の接続の確立を制御します。CONT パラメータの動作は、プログラミングされた接続が使用されるか、または設定した接続が使用されるかに、部分的に依存します。

- CONT = "0"の場合: データは受信されません(プログラミングされた接続が使用されるか、または設定した接続が使用されるかに無関係)。
- CONT = "0"を"1"に変更する場合:
  - プログラミングされた接続を使用する場合、「TCON」で確立されます。
  - 設定した接続を使用する場合、「T_DIAG」でチェックされます。
- CONT = "1"の場合:
  - データが受信されない限り(EN_R="0")、接続は「T_DIAG」でチェックされます。
  - 内部で使用される通信命令が、接続エンドポイントが存在しないことを信号で示した場合、接続が「TCON」で自動的に再確立されます。
- CONT = "1"を"0"に変更する場合:
  - プログラミングされた接続を使用する場合、「TDISCON」で終了されます。
  - 設定した接続を使用する場合、「T_RESET」でリセットされます。

パラメータ COM_RST は、「0」から「1」に切り替わると、接続をリセットします。

- 接続が確立された場合、「T_RESET」でリセットされます(プログラミングされた接続が使用されるか、または設定した接続が使用されるかに無関係)。
- 接続が確立されない場合、パラメータの設定は影響を及ぼしません。

EN_R および COM_RST パラメータが影響を及ぼすのは、CONT が「1」にセットされている場合のみです。 次の表に、 EN_R、CONT 、 COM_RST パラメータの関係を示します。

EN_R	CONT	COM_RST	命令のステータス	説明
対象外	0	対象外	まだ実行されていません	動作中のジョブはありません(STATUS = 7000)。
対象外	0	対象外	初期化	接続を終了中です。 命令をリセット中です。
対象外	0 > 1	対象外	接続の確立	接続を確立中です。 データはまだ転送されていません。
0	1	0	接続が確立されました。	接続が確立され、命令「T_DIAG」でモニタされます。
対象外	1	0 > 1	接続が確立されました。	接続が「T_RESET」によって一時的に割り込まれ、リセットされます。
0 > 1	1	0	接続が確立されました。	命令が受信を開始します。
対象外	1	0 > 1	データを受信中です。	データ伝送が割り込みされます。 接続をリセットしています。

設定した接続のシステムデータタイプ

CONNECT パラメータで設定した接続の場合、接続の記述には、TCON_Configured に従って以下の構造を使用します。

Byte (バイト)	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceID	HW_ANY	-	ローカルインターフェースのハードウェア識別子(値の範囲: 0 ~ 65535)。
2 … 3	ID	CONN_OUC	-	接続への参照(値の範囲: 1 ~ 4095)。 既存の接続の接続 ID を入力します。
4	Connection-Type	BYTE	-	接続タイプ 設定した接続に対して 254 (10 進数)を選択します。

BUSY、DONE および ERROR パラメータ

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、ジョブが正常に実行されたかどうかをチェックできます。 「TRCV_C」の実行中にエラーが発生すると、ERROR パラメータが設定されます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

DONE	BUSY	ER-ROR	説明
0	0	0	命令はまだ実行されていません(EN_R パラメータに立ち上がりエッジなし)。
0	1	0	命令を実行中であり、内部で使用される通信命令を呼び出します。
1	0	0	受信が正常に完了しました。「0000」が、STATUS パラメータに出力されます。 DONE = "1"が、1 つのサイクルに対してのみ表示されます。
0	0	1	命令の実行または処理中の中間ステップが、エラーで終了されました。内部で使用される通信命令のために、その後のエラーが発生した場合は、処理中に最初に発生したエラーが表示されます。 この状態は、1 つのサイクルに対してのみ常時されます。

ERROR および STATUS パラメータ

ER-ROR	STATUS (W#16#...)	説明
0	0000	受信ジョブがエラーなしで実行されました。
0	0001	通信接続が確立されました。
0	0003	通信接続が終了しました。
0	7000	ジョブ処理が行われていません。
0	7001	接続を確立するための最初の呼び出し。
0	7002	接続を確立するための 2 回目の呼び出し。
0	7003	通信接続を終了中です。
0	7004	通信接続が確立されており、モニタされています。 受信ジョブの処理が行われていません。
0	7006	現在データを受信中です。
1	8085	• LEN パラメータが最大許容値よりも大きくなっています。 • 最初の呼び出しの後に、LEN または DATA パラメータの値が変更されました。
1	8086	ID パラメータが許容範囲外です。
1	8087	最大接続数に達しました。これ以上の接続は行えません。
1	8088	LEN パラメータの値が、DATA パラメータで設定された受信領域と一致していません。
1	8089	CONNECT パラメータがデータブロックをポイントしていません。
1	8091	最大ネスト深さを超えています。
1	809A	CONNECT パラメータが、接続記述子の長さと一致しないフィールドをポイントしています。
1	809B	InterfaceID が無効です。 ゼロであるか、ローカル CPU インターフェースまたは CP をポイントしていません。
1	80A1	• 接続またはポートが既にユーザーによって使用中です。 • 通信エラー: ○ 指定された通信がまだ確立されていません。 ○ 指定された接続を終了中です。 この接続による転送は行えません。

		○ インターフェースを再初期化中です。
1	80A3	ネストされた「T_DIAG」命令によって、接続が終了したことがレポートされました。
1	80A4	接続のリモートエンドポイントの IP アドレスが無効であるか、またはローカルパートナーの IP アドレスと一致します。
1	80A7	通信エラー: 送信ジョブの完了前に、COM_RST = 1 で命令を呼び出しました。
1	80AA	別のブロックによって同じ接続 ID を持つ接続が現在確立されています。REQ パラメータの新しい立ち上がりエッジでジョブを繰り返します。
1	80B4	ISO-on-TCP プロトコルの種類(connection_type = B#16#12)を使用するときに、パッシブ接続の確立(active_est = FALSE)について、次の条件の一方または両方に違反しました。 "local_tsap_id_len >= B#16#02"および/または"local_tsap_id[1] = B#16#E0"。
1	80B5	接続タイプ 13 = UDP の場合、パッシブ接続の確立以外は許可されていません。
1	80B6	接続記述子のデータブロックの connection_type パラメータにパラメータ割り当てエラーがあります。
1	80B7	接続記述のデータブロックの次のパラメータのいずれかにエラーがあります: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80C3	• すべての接続リソースが使用中です。 • この ID を持つブロックは、別の優先度グループで既に処理中です。
1	80C4	一時的な通信エラー: • 現時点では、接続を確立できません。 • インターフェースが新しいパラメータを受信中であるか、接続が確立中です。 • 設定した接続が現在「TDISCON」命令によって削除されています。 • 使用された接続が、COM_RST = 1 で呼び出されたことで終了しています。
1	80C6	リモートパートナーに到達できません(ネットワークエラー)。
1	8722	CONNECT パラメータでエラーが発生しました。 ソース領域(データブロックで宣言されていない領域)が無効です。
1	873A	CONNECT パラメータでエラーが発生しました。 接続記述へのアクセスが行えません(データブロックにアクセスできません)。
1	877F	CONNECT パラメータでエラーが発生しました。 内部エラー
1	8922	パラメータ DATA： ターゲット領域が無効です。領域が DB 内に存在していません。
1	8924	パラメータ DATA： VARIANT ポインタで領域エラーが発生しました。
1	8932	パラメータ DATA： DB 番号が大きすぎます。
1	893A	パラメータ CONNECT： 指定された接続データにアクセスできません(たとえば、DB が存在しないため)。
1	897F	パラメータ DATA： 内部エラー。たとえば、VARIANT 参照が無効です。
1	8A3A	パラメータ DATA： たとえば、データブロックが存在しないため、データ領域にアクセスできません。

注記

命令「TCON」、「TRCV」および「TDISCON」のエラーメッセージ

内部的には、TRV_C 命令は「TCON」、「TRCV」および「TDISCON」命令を使用します。 これらの命令のエラーメッセージは、それぞれの説明に含まれています。

TMAIL_C: 電子メールの転送

この章には下記に関する情報が記載されています：

TMAIL_C の説明

説明

「TMAIL_C」命令を使用し、S7-1500 CPU または V4.0 よりも新しい S7-1200、通信モジュール(CM)、または通信プロセッサ(CP)のイーサネットインターフェースを経由して電子メールを送信します。

この命令は、ハードウェアが構成済みであり、かつネットワークインフラがメールサーバーへの通信接続を許可している場合にのみ使用可能です。

以下のパラメータを使用し、電子メールの内容および通信データを定義します。

- パラメータ TO_S および CC を使用して、宛先のアドレスを定義します。
- パラメータ SUBJECT および TEXT を使用して、電子メールの内容を定義します。
- ATTACHMENT および ATTACHMENT_NAME パラメータで VARIANT ポインタを使用して、添付ファイルを定義できます。
- MAIL_ADDR_PARAM パラメータでシステムデータタイプ TMail_V4、TMail_V6 、または TMail_FQDN を使用して、通信データを定義し、メールサーバーのアドレス指定および認証を実行します。
  - S7-1500 CPU のインターフェースを使用している場合、システムデータタイプ TMail_V4 を使用する必要があります。 この場合、電子メールは SMTP によってのみ送信可能です。
  - CM/CP のインターフェースを使用している場合は、いずれのシステムデータタイプも使用可能です。この場合も、電子メールを SMTPS によって送信可能です。
- 電子メールの送信は、REQ パラメータの「0」から「1」へのエッジの切り替えで開始します。
- ジョブステータスは、出力パラメータ「BUSY」、「DONE」、「ERROR」、および「STATUS」で示されます。

「TMAIL_C」命令では、SMS を直接送信することはできません。 電子メールがメールサーバーによって SMS として転送されるかどうかは、通信プロバイダに依存します。

注記

送信する電子メールの数

PLC を使用して同時に複数の電子メールを送信できます。 CP 1243-8 または CP 1543-1 を使用する場合、各 CP につき 1 つの電子メールのみ送信できます。2 つの CP を使用する場合、2 つの電子メールの並列送信が可能です。

命令の操作

「TMAIL_C」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。「TMAIL_C」命令を呼び出すとき、インスタンスを指定する必要があります。

以下の場合、メールサーバへの接続が切断されます。

- 「TMAIL_C」がアクティブであるにもかかわらず、CPU が STOP に切り替わった場合。
- 産業用イーサネットバスで通信異常が発生した場合。

この場合、電子メールの転送が割り込みされ、宛先に到達しません。 命令が正常に実行され、電子メールが送信された場合も、通信が中止されます。

通知

ユーザープログラムの変更

使用しているユーザープログラムでは、「TMAIL_C」の呼び出しに直接影響を与える部分のみを変更できます。

- CPU が「STOP」モードの場合。
- 電子メールが送信されない場合(REQ = 0 および BUSY = 0)。

これは、特に、「TMAIL_C」呼び出しまたは「TMAIL_C」インスタンスの呼び出しを含むプログラムブロックの削除および置き換えに関連します。

この制限を無視すると、接続リソースが占有される場合があります。 産業用イーサネット経由で、オートメーションシステムの TCP/IP 通信ファンクションが未定義ステータスに変更されることがあります。

この切り替えが転送された後、CPU のウォームリスタートまたはコールドリスタートが必要になります。

データの整合性

TO_S、CC、SUBJECT、TEXT、ATTACHMENT、および MAIL_ADDR_PARAM パラメータは、「TMAIL_C」命令の実行中にこの命令によって適用されます。これは、これらのパラメータがジョブの完了後(BUSY = 0)以外には変更できないことを意味します。

SMTP 認証

認証は、ID の確認手順、たとえばパスワードの照会などで、これを参照します。

S7-1500 CPU インターフェースの場合、「TMAIL_C」命令は多くのメールサーバーが必要とする SMTP 認証手続き AUTH-LOGIN をサポートしています。 使用しているメールサーバーの認証手続きについての情報は、メールサーバーのマニュアルまたはインターネットサービスプロバイダの Web サイトを参照してください。

- 「TMAIL_C」命令がメールサーバーにログオンするためのユーザー名を要求するまで、AUTH-LOGIN 認証手続きは使用できません。 このユーザー名は、メールサーバーでメールアカウントを設定するために使ったユーザー名に対応します。 ユーザー名は UserName パラメータによって、パラメータ MAIL_ADDR_PARAM の構造体に転送されます。

  MAIL_ADDR_PARAM パラメータでユーザー名が指定されていない場合、AUTH-LOGIN 認証手続きは使用されません。 電子メールは、認証なしで送信されます。
- ログオンするために、「TMAIL_C」命令は、関連するパスワードを要求します。 このパスワードは、お使いのメールアカウントを設定したときに指定したパスワードに対応します。 パスワードは PassWord パラメータによって、パラメータ MAIL_ADDR_PARAM の構造体に転送されます。

パラメータ

以下の表に、「TMAIL_C」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、T、C、または定数	制御パラメータ REQUEST: 信号立ち上がりエッジで電子メールの送信を有効にします。
TO_S	Input	STRING	D、L、または定数	宛先アドレス 最大長 240 文字(バイト)の STRING。 電子メールアドレス書式については、パラメータの説明に記載されている例を参照してください。

CC	Input	STRING	D、L、または定数	CC 宛先アドレス(オプション) 最大長 240 文字(バイト)の STRING。 TO_S パラメータと同じ電子メールアドレス書式です。 ここに空の文字列が割り当てられている場合、電子メールは CC 受信者には送信されません。
SUBJECT	Input	STRING	D、L、または定数	電子メールの件名 最大長 240 文字(バイト)の STRING。
TEXT	Input	STRING	D、L、または定数	電子メールのテキスト(オプション) 最大長 240 文字(バイト)の STRING。 このパラメータに空の文字列が割り当てられている場合、電子メールはテキストなしで送信されます。
ATTACH-MENT	Input	VARIANT	D	電子メールの添付ファイル(オプション) 最大長 64KB のバイト/ワード/ダブルワードフィールド(ArrayOfByte、ArrayOfWord、または ArrayOfDWord)の参照。 値が割り当てられていない場合、電子メールは添付ファイルなしで送信されます。
ATTACH-MENT_NAME	Input	STRING	D、L、または定数	電子メールの添付ファイル名(オプション) 添付ファイル名を定義する最大長 50 文字の文字列の参照。 このパラメータに空の文字列が割り当てられている場合、電子メールの添付ファイルはファイル名「attach-ment.bin」で送信されます。
MAIL_ADDR_PARAM	Input	VARIANT	D	電子メールサーバーの接続パラメータおよびアドレス 接続パラメータを定義するには、構造体 TMail_V4、TMail_V6 、または TMail_FQDN を使用します(パラメータの説明を参照)。
DONE	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • DONE = 0: ジョブがまだ開始されていないか、または実行中です。 • DONE = 1: ジョブがエラーなしで実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • BUSY=0:「TMAIL_C」の処理が停止されました。 • BUSY = 1: 電子メールの送信が未完了です。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ • ERROR = 0: エラーは発生しませんでした。 • ERROR = 1: 処理中にエラーが発生しました。STATUS にエラーの種類に関する詳細情報を表示します。

STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ 「TMAIL_C」命令の戻り値またはエラー情報(パラメータの説明を参照)。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記

オプションのパラメータ

オプションのパラメータ CC、TEXT、および ATTACHMENT は、対応するパラメータが長さ> 0 の文字列を含む場合のみ電子メールと一緒に送信されます。

例

下記のリンクに TMAIL_C 命令を使用した電子メール送信のサンプルがあります。 例: TMAIL_C による電子メールの送信

TO_S および CC パラメータ

説明

TO_S および CC パラメータは、たとえば次の内容を持つ文字列です。

- <wenna@mydomain.com>, <ruby@mydomain.com>
- <admin@mydomain.com>, <judy@mydomain.com>

パラメータの入力時には、次のルールに注意してください。

- それぞれのアドレスの前に空白文字および右開きの括弧「<」を入力する必要があります。
- それぞれのアドレスの後に左開きの括弧「>」を入力する必要があります。
- TO および CC では、アドレスの間にカンマを入力する必要があります。

ランタイムおよびメモリ領域の理由で、「TMAIL_C」命令はパラメータ TO_S または CC の構文チェックを行いません。

MAIL_ADDR_PARAM パラメータ

説明

MAIL_ADDR_PARAM パラメータでは、構造体 TMail_V4、TMail_V6、または Tmail_FQDN で電子メール送信のための接続を定義し、電子メールサーバーのアドレスおよびログイン情報を保存します。

MAIL_ADDR_PARAM パラメータで使用する構造体は、電子メールサーバをアドレス指定する形式によって異なります。

- TMail_V4: IP アドレス(IPv4)によるアドレス指定。
- TMail_V6: IP アドレス(IPv6)によるアドレス指定。
- TMail_FQDN: 完全修飾ドメイン名(FQDN)によるアドレス指定。

どの構造体が使用可能であるかは、InterfaceId パラメータでアドレス指定されたインターフェースによって異なります。

- 内部インターフェースで「TMAIL_C」命令を使用する場合、TMail_V4 構造体を MAIL_ADDR_PARAM パラメータで使用する必要があります。
- 通信プロセッサ(CP)または通信モジュール(CM)を使用している場合は、これら 3 つすべてのアドレス指定方法(IPv4、IPv6、および FQDN)が使用可能です。

TMail_V4 : IP アドレス(IPv4)によるメールサーバーのアドレス指定

パラメータ	データタイプ	説明
TMail_V4	Struct	
InterfaceId	LADDR	インターフェースのハードウェア識別子
ID	CONN_OUC	接続 ID
ConnectionType	BYTE	接続タイプ。 IPv4 の接続タイプとして 16#20 を選択します。
ActiveEstablished	BOOL	ステータスビット。 接続が確立されると「1」にセットされます。
CertIndex	BYTE	• =0: SMTP を使用(**S**imple **M**ail **T**ransfer **P**rotocol)。電子メールを S7-1500 CPU のインターフェース経由で送信する場合は、SMTP を使用する必要があります。 • ≠0: SMTPS を使用すると、CP/CM との接続が確立前にセキュアになります。 CertIndex パラメータを使用し、使用する証明書を指定します([プロジェクトナビゲーション|グローバルセキュリティ設定|証明書マネージャ]を参照)。
WatchDogTime	TIME	実行ウォッチドッグ。 このパラメータを使用して、送信処理の最大実行時間を定義します。 注記: 接続が遅い場合、接続の確立により時間がかかります(約 1 分)。 WATCH_DOG_TIME パラメータを指定した場合、接続を確立するために必要な時間を考慮してください。 指定された時間が経過すると、接続が終了します。

MailServerAddress	IP_V4	メールサーバーの IP アドレス。 次の書式で表記された IPv4 です: XXX.XXX.XXX.XXX (10 進数)。 例: 192.142.131.237.
UserName	STRING[254]	メールサーバーのログイン名
PassWord	STRING[254]	メールサーバーのパスワード
From	EMAIL_ADDR	以下の 2 つの STRING パラメータを使用して定義される電子メール送信元アドレス。 たとえば、 "myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	送信元アドレスの@記号を含むローカル部分。 例: "myname@".
FullQualifiedDomainName	STRING[254]	メールサーバーの Fully Qualified Domain Name (略称 FQDN)。例: "mymailserver.com".

TMail_V6 : IP アドレス(IPv6)によるメールサーバーのアドレス指定

パラメータ	データタイプ	説明
TMail_V6	Struct	
InterfaceId	LADDR	インターフェースのハードウェア識別子
ID	CONN_OUC	接続 ID
ConnectionType	BYTE	接続タイプ。 IPv6 の接続タイプとして 16#21 を選択します。
ActiveEstablished	BOOL	ステータスビット。 接続が確立されると「1」にセットされます。
CertIndex	BYTE	• =0: SMTP を使用(**S**imple **M**ail **T**ransfer **P**rotocol)。電子メールを S7-1500 CPU のインターフェース経由で送信する場合は、SMTP を使用する必要があります。 • ≠0: SMTPS を使用すると、CP/CM との接続が確立前にセキュアになります。 CertIndex パラメータを使用し、使用する証明書を指定します([プロジェクトナビゲーション|グローバルセキュリティ設定|証明書マネージャ]を参照)。
WatchDogTime	TIME	実行ウォッチドッグ。 このパラメータを使用して、送信処理の最大実行時間を定義します。 注記: 接続が遅い場合、接続の確立により時間がかかります(約 1 分)。WATCH_DOG_TIME パラメータを指定した場合、接続を確立するために必要な時間を考慮してください。 指定された時間が経過すると、接続が終了します。
MailServerAddress	IP_V6	次の書式で表記されたメールサーバーの IP アドレス(IPv6)です: XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX.XXXX (16 進数)。 アドレスは、それぞれ 2 バイトの 8 つのブロックに分けられます(合計で 16 バイト)。 例: 2001:db8:1f11:08d3:290:27ff:0370:2093
UserName	STRING[254]	メールサーバーのログイン名
PassWord	STRING[254]	メールサーバーのパスワード

From	EMAIL_ADDR	以下の 2 つの STRING パラメータを使用して定義される電子メール送信元アドレス。たとえば、"myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	送信元アドレスの@記号を含むローカル部分。例:"myname@".
FullQualifiedDomainName	STRING[254]	メールサーバーの Fully Qualified Domain Name (略称 FQDN)。例: "mymailserver.com".

TMail_FQDN : FQDN によるメールサーバーのアドレス指定 FQDN

パラメータ	データタイプ	説明
TMail_V6	Struct	
TMail_FQDN	LADDR	インターフェースのハードウェア識別子
ID	CONN_OUC	接続 ID
ConnectionType	BYTE	接続タイプ。 FQDN の接続タイプとして 16#22 を選択します。
ActiveEstablished	BOOL	ステータスビット。 接続が確立されると「1」にセットされます。
CertIndex	BYTE	• =0: SMTP を使用(**S**imple **M**ail **T**ransfer **P**rotocol)。電子メールを S7-1500 CPU のインターフェース経由で送信する場合は、SMTP を使用する必要があります。 • ≠0: SMTPS を使用すると、CP/CM との接続が確立前にセキュアになります。 CertIndex パラメータを使用し、使用する証明書を指定します([プロジェクトナビゲーション|グローバルセキュリティ設定|証明書マネージャ]を参照)。
WatchDogTime	TIME	実行ウォッチドッグ。 このパラメータを使用して、送信処理の最大実行時間を定義します。 注記: 接続が遅い場合、接続の確立により時間がかかります(約 1 分)。 WATCH_DOG_TIME パラメータを指定した場合、接続を確立するために必要な時間を考慮してください。 指定された時間が経過すると、接続が終了します。
MailServerAddress	STRING[254]	メールサーバーの FQDN (**F**ully **Q**ualified **D**omain **N**ame)。完全修飾ドメイン名を使用してメールサーバーをアドレス指定します。 例: "www.mymailserver.com.".
UserName	STRING[254]	メールサーバーのログイン名
PassWord	STRING[254]	メールサーバーのパスワード
From	Struct	以下の 2 つの STRING パラメータを使用して定義される電子メール送信元アドレス。たとえば、"myname@mymailserver.com".
LocalPartPlusAtSign	STRING[64]	送信元アドレスの@記号を含むローカル部分。例:"myname@".
FullQualifiedDomainName	STRING[254]	メールサーバーの Fully Qualified Domain Name (略称 FQDN)。例: "mymailserver.com".

DONE、BUSY、および ERROR パラメータ

説明

出力パラメータ DONE、BUSY および ERROR は、それぞれ BUSY 出力パラメータのステータスが「1」から「0」に切り替わった場合に、1 サイクルのみ表示されます。

次の表に、DONE、BUSY、および ERROR の関係を示します。この表を使って、「TMAIL_C」命令の現在のステータスと、電子メールの送信がいつ完了するか決定できます。

DONE	BUSY	ER-ROR	説明
0	1	0	ジョブが処理中です。
1	0	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータで確認できます。
0	0	0	「TMAIL_C」命令に(新規)ジョブが割り当てられていません。

STATUS パラメータ

説明

次の表に、STATUS パラメータでの「TMAIL_C」の戻り値を示します。

戻り値 STATUS* (W#16#...):	説明	注記
0000	「TMAIL_C」の処理がエラーなしで完了されました。	「TMAIL_C」がエラーなしで完了しても、送信した電子メールが届いたことを意味しません。 宛先アドレスの入力が不正な場合でも、「TMAIL_C」命令のステータスエラーは生成されません。 この場合、これらが正しく入力されていても、電子メールの他の受信者への送信は保証されません。
7001	"TMAIL_C」が有効(BUSY = 1)です。	最初の呼び出し: ジョブがトリガされています。
7002	"TMAIL_C」が有効(BUSY = 1)です。	中間呼び出し ジョブが既にアクティブです。
8xxx	「TMAIL_C」の処理が、内部呼び出しの通信命令のエラーコードで完了しました。	詳細情報については、「TCON」、「TDISCON」、「TSEND」、および「TRCV」通信命令の STATUS パラメータの説明を参照してください。
8010	接続の確立中にエラーが発生しました。	評価に関する詳細情報については、インスタンスデータブロックの SFB_STATUS パラメータに記載されています。SFB_STATUS パラメータに表示されるエラーコードは、「TCON」命令の STATUS パラメータの説明に記載されています。
8011	データ送信のエラー	評価に関する詳細情報については、インスタンスデータブロックの SFB_STATUS パラメータに記載されています。SFB_STATUS パラメータに表示されるエラーコードは、「TSEND」命令の STATUS パラメータの説明に記載されています。
8012	データ受信のエラー	評価に関する詳細情報については、インスタンスデータブロックの SFB_STATUS パラメータに記載されています。SFB_STATUS パラメータに表示されるエラーコードは、「TRCV」命令の STATUS パラメータの説明に記載されています。
8013	接続の確立中にエラーが発生しました。	評価に関する詳細情報については、インスタンスデータブロックの SFB_STATUS パラメータに記載されています。SFB_STATUS パラメータに表示されるエラーコードは、「TCON」および「TDISCON」命令の STATUS パラメータの説明に記載されています。

8014	接続の確立が不可能です。	不正なメールサーバーの IP アドレス(MailServerAddress)、または接続を確立するためには短すぎる時間(WatchDogTime)を入力した可能性があります。 また、CPU がネットワークに接続していない、または CPU の設定が不正である可能性もあります。
8015	MAIL_ADDR_PARAM のデータタイプが不正です	有効なデータタイプは、システムデータタイプ (構造体) TMail_V4、TMail_V6、および TMail_FQDN のみです。
8016	ATTACHMENT パラメータのデータタイプが不正です	有効なデータタイプは、ArrayOfByte、ArrayOfWord、および ArrayOfDWord のみです。
8017	ATTACHMENT パラメータのデータ長が不正です	データ長は 65534 バイト以下である必要があります。
8401	使用可能なチャンネルがありません。考えられる原因: この CP を経由する電子メール接続が既に存在しています。2 番目の接続を並行して確立することはできません。	CP 1543 固有のエラー
8403	メールサーバーに TCP/IP 接続できませんでした。	CP 1543 固有のエラー
8405	サーバーがログイン要求を拒否しました。	CP 1543 固有のエラー
8406	SMTP クライアントが、内部的な SSL エラー、または証明書の構造の問題を検出しました。	CP 1543 固有のエラー
8407	SSL の使用要求が拒否されました。	CP 1543 固有のエラー
8408	メールサーバーへの TCP/IP 接続のためのソケットをクライアントが識別できません。	CP 1543 固有のエラー
8409	この接続を経由して書き込みできません。 考えられる原因: 通信パートナーが接続をリセットしたか、または接続が切断されました。	CP 1543 固有のエラー
8410	この接続を経由して読み取りできません。 考えられる原因: 通信パートナーが接続をリセットしたか、または接続が切断されました。	CP 1543 固有のエラー
8411	電子メールを送信できませんでした。原因: 送信処理の実行のためのメモリ領域が不足しています。	CP 1543 固有のエラー
8412	構成された DNS サーバーが、指定されたドメイン名を解決できませんでした。	CP 1543 固有のエラー
8413	DNS サブシステムの内部エラーにより、ドメイン名を解決できませんでした。	CP 1543 固有のエラー
8414	ドメイン名としてからの文字列が入力されています。	CP 1543 固有のエラー
8415	cURL モジュールで内部エラーが発生しました。 実行が停止されました。	CP 1543 固有のエラー

8416	SMTP モジュールで内部エラーが発生しました。 実行が停止されました。	CP 1543 固有のエラー
8417	SMTP に対する要求がチャンネルで既に使用されているか、チャンネル ID が無効です。実行は停止されました。	CP 1543 固有のエラー
8418	電子メール送信ジョブが中止されました。 考えられる原因: 実行時間 (WatchDogTime パラメータ)を超過したか、または CP が開始から停止に移行しました。	CP 1543 固有のエラー
8419	チャンネルに割り込みが発生したため、接続が閉じられるまで使用できません。	CP 1543 固有のエラー
8420	サーバー証明書の文字列を CP ルート証明書で検証できませんでした。	CP 1543 固有のエラー
8421	内部エラーが発生しました。 実行が停止されました。	CP 1543 固有のエラー
82xx, 84xx, OR 85xx	エラーメッセージがメールサーバーから発信され、SMTP プロトコルのエラー番号に対応します(「8」を除く)。 次の行に、発生する可能性のあるエラーコードを示します。	SMTP エラーコードおよびその他の SMTP プロトコルのエラーコードの詳細な情報は、インターネットまたはメールサーバーのエラー文書で参照してください。 メールサーバーからの最新のエラーメッセージは、BUFFER1 パラメータで使用しているインスタンス DB でも参照できます。 「TMAIL_C」命令によって最後に送信されたデータは、インスタンス DB の DATEN で確認できます。
8450	操作が実行されませんでした: メールボックスが使用できないか、到達できませんでした。	後で再度実行してみてください。
8451	操作が中止されました: ローカルの処理エラー	後で再度実行してみてください。
8500	シンタックスエラー: エラーが認識されませんでした. これには、コマンド文字列が長すぎる場合のエラーも含まれます。 これは、電子メールが LOGIN 認証手続きをサポートしない場合にも発生します。	「TMAIL_C」のパラメータをチェックします。電子メールを認証なしで送信してみてください。 これを行うには、UserName パラメータの内容を空の文字列で置き換えます。 ユーザー名が指定されていない場合、LOGIN 認証手続きは使用されません。
8501	シンタックスエラー: パラメータの入力が不正です	考えられる原因: TO_S または CC パラメータのアドレスが不正です(関連項目: TO_S および CC パラメータ).
8502	コマンドが不明または実装されていません	エントリ、特に FROM パラメータをチェックします。 このパラメータが不完全であったり、「@」または「.」が抜けていたりする場合があります (関連項目: TO_S および CC パラメータ).
8504	コマンドパラメータが実装されていません	エントリ、特に FROM パラメータをチェックします。 このパラメータが不完全であったり、「@」が抜けていたりする場合があります (関連項目: TO_S および CC パラメータ).
8535	SMTP 認証が不完全です	不正なユーザー名または不正なパスワードを入力した可能性があります。

8550	メールサーバーに到達できません。アクセス権がありません。	不正なユーザー名または不正なパスワードを入力した可能性があります。または、使用しているログイン方法をメールサーバーがサポートしていません。 エラーの別の原因としては、「@」の後のドメイン名の間違い、または、TO_S、CC、および FROM パラメータでの「.」の不足が考えられます(関連項目: TO_S および CC パラメータ).
8552	操作が中止されました: 割り当てられたメモリサイズを超えました。	後で再度実行してみてください。
8554	転送に失敗しました	後で再度実行してみてください。
8555	電子メールアドレスの不正な入力によるエラー。	複数のアドレスを入力した場合、「,」で区切る必要があります。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

例: TMAIL_C による電子メールの送信

概要

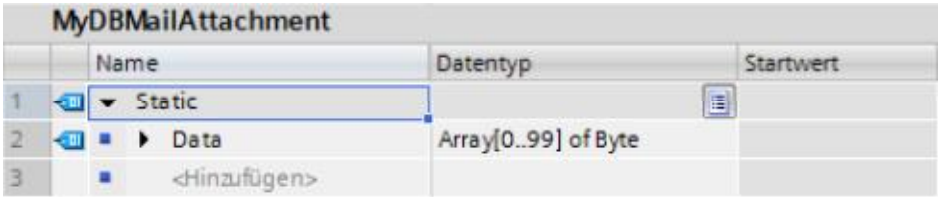
次の呼び出しの例で、TMAIL_C 命令がどのように機能するかを示します。

要件

電子メールを送信するにはハードウェアが事前に構成済みであり、かつネットワークインフラがメールサーバーへの通信接続を許可している必要があります。 さらに、TMAIL_C の下記の入力パラメータを提供するために 2 つの構造体を作成する必要があります。 1 つは電子メール添付に関する入力パラメータ、もう 1 つは接続情報とアドレスデータに関する入力パラメータです。

電子メール添付

電子メールは添付ファイル付きで送信されます。 添付ファイルのソースとしてグローバルデータブロックにファイル名「Data」の Array of Byte を作成します。 この配列は、後で ATTACHMENT 入力パラメータと相互接続します。

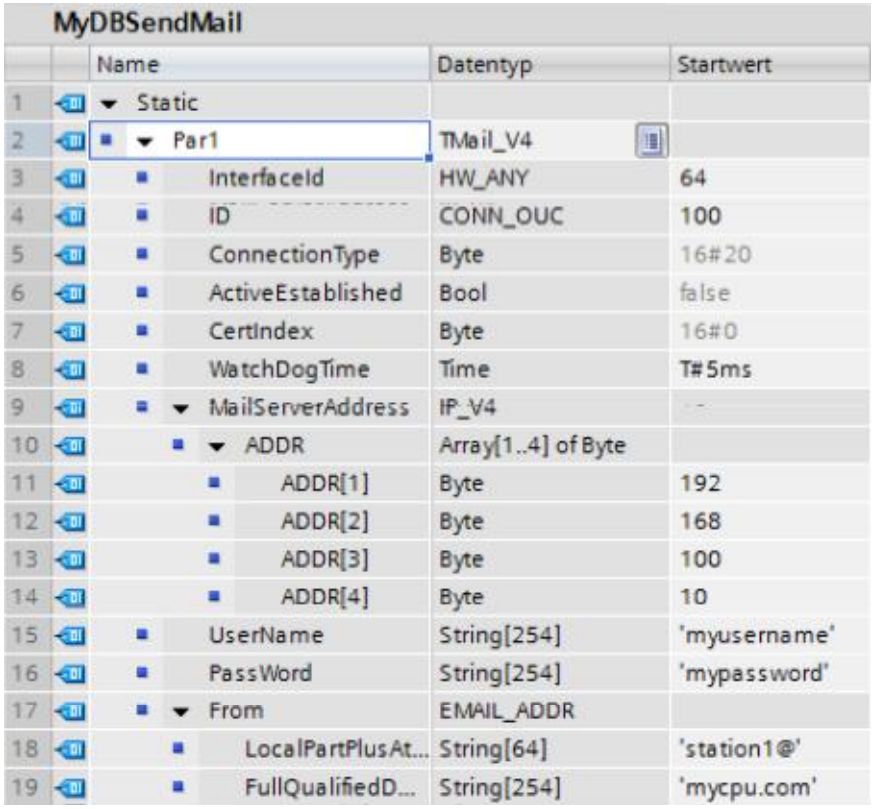


接続情報およびアドレスデータ

CPU の接続情報とメールサーバーのアドレスデータを TMail_V4 システムデータタイプで入力します。

- グローバルデータブロックに「TMail_V4」データタイプの「Par1」タグを作成します。
- 以下の構造体に、使用しているメールサーバーの設定と接続データに基づいて使用している CPU のパラメータを入力します。 詳細情報については、MAIL_ADDR_PARAM 入力パラメータの説明を参照してください。

「Par1」タグは、後で MAIL_ADDR_PARAM 入力パラメータと相互接続します。



MyDBSendMail

	Name	Datentyp	Startwert
1	Static		
2	Par1	TMail_V4	
3	InterfaceId	HW_ANY	64
4	ID	CONN_OUC	100
5	ConnectionType	Byte	16#20
6	ActiveEstablished	Bool	false
7	CertIndex	Byte	16#0
8	WatchDogTime	Time	T#5ms
9	MailServerAddress	IP_V4	
10	ADDR	Array[1..4] of Byte	
11	ADDR[1]	Byte	192
12	ADDR[2]	Byte	168
13	ADDR[3]	Byte	100
14	ADDR[4]	Byte	10
15	UserName	String[254]	'myusername'
16	PassWord	String[254]	'mypassword'
17	From	EMAIL_ADDR	
18	LocalPartPlusAt...	String[64]	'station1@'
19	FullQualifiedD...	String[254]	'mycpu.com'

命令の呼び出し

SCL 言語で以下のインターフェースを持つ新しいファンクションブロックを作成します。

- InOut
  - 名前:SendEMail
  - データタイプ:Bool
- Static
  - 名前:STATUS
  - データタイプ: Word

TMAIL_C の呼び出しとともに、下記のソースコードをプログラミングウィンドウにコピーします。

```
TMAIL_C を呼び出します。
BEGIN
    IF #SendEMail = true THEN
     "TMAIL_C_DB"(REQ := NOT "TMAIL_C_DB".BUSY,
                  TEXT := 'The cpu switched to run',
                  ATTACHMENT := "MyDBMailAttach-
ment".Data,
                  MAIL_ADDR_PARAM := "MyDBSend-
Mail".Par1);
     IF ("TMAIL_C_DB".BUSY = false) AND
("TMAIL_C_DB".DONE = false)
```

```
AND
      ("TMAIL_C_DB".ERROR = false) THEN
          #SendEMail := false;
      END_IF;
      IF ("TMAIL_C_DB".DONE = false) OR
("TMAIL_C_DB".ERROR = true) THEN
          #STATUS := "TMAIL_C_DB".STATUS;
        END_IF;
    END_IF;
```

結果

この命令は SendEMail = true で 1 回呼び出す必要があります。 以下の呼び出しでは、呼び出し時に SendEMail を割り当てないでください。電子メールが送信されるか送信中にエラーが発生するとすぐに、表示された命令で SendEMail がリセットされます。

電子メールが正常に送信されなかった場合、STATUS パラメータには TMAIL_C 命令の STATUS パラメータの対応する値が含まれます。

その他

この章には下記に関する情報が記載されています：

TCON: 通信接続の確立

この章には下記に関する情報が記載されています：

- TCON: 通信接続の確立 (S7-1200)
- TCON: 通信接続の確立 (S7-1500)

TCON: 通信接続の確立

説明

「TCON」命令を使用し、通信接続を設定して確立します。 接続の設定と確立が終了すると、CPU によって自動的に維持およびモニタされます。 "TCON」は非同期的に実行されます。

CONNECT および ID パラメータに対して指定された接続データは、通信接続の設定に使用されます。 接続を確立するには、REQ パラメータで信号立ち上がりエッジを検出する必要があります。 接続が正常に確立されると、DONE パラメータが「1」にセットされます。

実行可能な接続数

実行可能な通信接続の数については、使用している CPU の技術仕様を参照してください。

TCP および ISO-on-TCP での接続

通信パートナーの両方が、「TCON」命令を呼び出して通信接続を設定し、確立します。 パラメータ割り当て中に、アクティブ通信エンドポイントとなるパートナーとパッシブ通信エンドポイントとなるパートナーを指定します。

断線やリモート通信パートナーなどが原因で接続が中断された場合、アクティブパートナーは設定した接続の再確立を試みます。 「TCON」を再度呼び出す必要はありません。 これは、一度「TCON」が正常に確立された場合のみ適用されます(DONE = 1)。

「TDISCON」命令が実行されるか、CPU が STOP モードに切り替わると、既存の接続が終了し、設定された接続が削除されます。 接続を再度設定して確立するには、「TCON」を再実行する必要があります。

パラメータ

次の表に、「TCON」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	ID で指定された接続を確立するジョブを信号立ち上がりエッジで開始します。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	割り当てられた接続へのリファレンス 値の範囲 W#16#0001 ~ W#16#0FFF
CONNECT	InOut	TCON_Par-am	D	接続記述子へのポインタ 関連項目 TCON_Param に従う構造の接続パラメータ
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み

BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

BUSY、DONE および ERROR パラメータ

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータを使用し、ジョブが正常に実行されたかどうかをチェックします。 「TCON」の実行中にエラーが発生すると、ERROR パラメータがセットされます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ERROR	説明
1	0	0	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 STATUS パラメータにエラーの原因が出力されます。
0	0	0	新しいジョブが割り当てられませんでした。

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	接続が正常に確立されました。
0	7000	ジョブ処理が行われていません。
0	7001	ジョブの実行を開始し、接続を確立します。
0	7002	接続を確立中です(REQ 対象外)。
1	8085	接続 ID (ID パラメータ)は、設定した接続によって既に使用されています。
1	8086	ID パラメータが有効範囲外です。
1	8087	最大接続数に達しました。これ以上の接続は行えません。
1	8089	CONNECT パラメータがデータブロックをポイントしていません。
1	809A	CONNECT パラメータが、接続記述子の長さと一致しないフィールドをポイントしています。
1	809B	TCON_xxx 構造体内の工レメント InterfaceId が、CPU または CM/CP インターフェースのハードウェア識別子を参照しないか、値が「0」です。

1	80A0	エラーコード W#16#80A1 および W#16#80A2 のグループエラーです。
1	80A2	ローカルまたはリモートポートがシステムによって使用中です。
1	80A3	既存の接続を再確立しようとしています。
1	80A4	接続のリモートエンドポイントの IP アドレスが無効です。つまり、ローカルパートナーの IP アドレスと一致しています。
1	80A5	接続 ID が既に使用されています。
1	80A7	通信エラー: 「TCON」の完了前に「TDISCON」を実行しました。
1	80B2	CONNECT パラメータが、属性「ロードメモリにのみ保存」を使用して生成されたデータブロックをポイントしています。
1	80B4	ISO-on-TCP プロトコルの種類(connection_type = B#16#12)を使用したパッシブ接続の確立について、次の条件の 1 つ以上に違反しました。 • local_tsap_id_len >= B#16#02 • local_tsap_id[1] = B#16#E0 • local_tsap_id_len >= B#16#03 の場合、local_tsap_id[1]は ASCII 文字であること。 • local_tsap_id[1]は ASCII 文字であり、local_tsap_id_len >= B#16#03 であること。
1	80B5	接続タイプ 13 = UDP の場合、パッシブ接続の確立しか許可されてません。
1	80B6	SDT TCON_Param の connection_type パラメータでのパラメータ割り当てエラー。
1	80B7	接続記述のデータブロックの次のパラメータのどれかにエラーがあります: block_length, local_tsap_id_len, rem_subnet_id_len, rem_staddr_len, rem_tsap_id_len, next_staddr_len.
1	80B8	構造体エレメント ID およびブロックパラメータ ID の接続記述子が同一ではありません。
1	80C3	すべての接続リソースが使用中です。
1	80C4	一時的な通信エラー: • 現時点では、接続を確立できません。 • インターフェースが現在、新しいパラメータを受信中です。 • 設定した接続が、現在「TDISCON」命令によって削除されています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

TCON: 通信接続の確立

説明

「TCON」命令を使用し、通信接続を設定して確立します。 接続の設定と確立が終了すると、CPU によって自動的に維持およびモニタされます。 "TCON」は非同期的に実行されます。

CONNECT および ID パラメータに対して指定された接続データは、通信接続の設定に使用されます。

- CONNECT パラメータでは、可能な場合は、プログラムエディタのインスペクタウィンドウ内の接続パラメータ割り当てによって作成された状態の、あらかじめ定義された構造を使用してください。
- CONNECT パラメータがパラメータ表にリストされた構造体と相互接続されていない場合、または構造体にエラーが含まれている場合、エラーコード 8089 が STATUS パラメータに出力されます。

接続を確立するには、REQ パラメータで信号立ち上がりエッジを検出する必要があります。 接続が正常に確立されると、DONE パラメータが「1」にセットされます。

命令「"TCON」V3.0 以降は、CPU S7-1200 バージョン 4.0 以降でも使用可能です。

実行可能な接続数

実行可能な通信接続の数については、使用している CPU の技術仕様を参照してください。

TCP および ISO-on-TCP での接続

通信パートナーの両方が、「TCON」命令を呼び出して通信接続を設定し、確立します。 パラメータ割り当て中に、アクティブ通信エンドポイントとなるパートナーとパッシブ通信エンドポイントとなるパートナーを指定します。

断線やリモート通信パートナーなどが原因で接続が中断された場合、アクティブパートナーは設定した接続の再確立を試みます。 「TCON」を再度呼び出す必要はありません。 これは、一度「TCON」が正常に確立された場合のみ適用されます(DONE = 1)。

「TDISCON」命令が実行されるか、CPU が STOP モードに切り替わると、既存の接続が終了し、設定された接続が削除されます。 接続を再度設定して確立するには、「TCON」を再実行する必要があります。

パラメータ

以下の表に、「TCON」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	指定された接続を確立するジョブを信号立ち上がりエッジで開始します。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	割り当てられた接続へのリファレンス 値の範囲: W#16#0001 終了 W#16#0FFF
CONNECT	InOut	VARIANT	D	接続記述子へのポインタ • TCP または UDP の場合、構造体 TCON_IP_v4 を使用します。

				詳細については、次を参照してください: TCON_IP_v4 に従う構造の接続パラメータ • ISO-on-TCP の場合、構造体 TCON_IP_RFC を使用します。 詳細については、次を参照してください: TCON_IP_RFC に従う構造の接続パラメータ • ISO の場合、構造体 TCON_ISOnative を使用します(CP1543-1 のみ)。 詳細については、次を参照してください: "TCON_ISOnative に基づいた接続記述子の構造」を参照してください。
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

BUSY、DONE および ERROR パラメータ

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータを使用し、ジョブが正常に実行されたかどうかをチェックします。 「TCON」の実行中にエラーが発生すると、ERROR パラメータが設定されます。 エラー情報が STATUS パラメータに出力されます。

リモートパートナーへのアクティブ接続の確立が失敗した場合、バージョン 3.0 では命令「TCON」がエラーメッセージを生成します。 もう一度接続を確立するには、REQ パラメータで信号立ち上がりエッジを作成します。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ERROR	説明
1	0	0	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。

0	0	1	ジョブがエラーありで終了しました。 STATUS パラメータにエラーの原因が出力されます。
0	0	0	新しいジョブが割り当てられませんでした。

TCON_ISOnative に基づいた接続記述子の構造

TCON_ISOnative に基づいた構造の接続記述子 DB を使用して、ISO 用の通信接続パラメータを割り当てます。 TCON_ISOnative の既定のデータ構造体には、接続の確立に必要なすべてのパラメータが含まれています。

Byte (バイト)	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceId	HW_ANY	64	ローカルインターフェースのハードウェア識別子(値の範囲: 0～65535)。
2 … 3	ID	CONN_OUC	1	この接続の参照(次の値の範囲の一意となる ID: 1～4095)。
4	ConnectionType	BYTE	16#16	接続タイプ: ISO
5	ActiveEstablished	BOOL	TRUE	接続の確立方法の ID: • FALSE: パッシブ接続の確立 • TRUE: アクティブ接続の確立
8 … 13	RemoteMacAd-dress	ARRAY [1..6] of BYTE	-	パートナーエンドポイントの MAC アドレス。たとえば 00-74-41-FD-AE-84 の場合 • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84
14 ... 19	LocalMacAddress	ARRAY [1..6] of BYTE	-	ローカルエンドポイントの MAC アドレス。たとえば 00-74-41-FD-AE-84 の場合 • MacAddr[1] = 00 • MacAddr[2] = 74 • MacAddr[3] = 41 • MacAddr[4] = FD • MacAddr[5] = AE • MacAddr[6] = 84
20 ... 53	RemoteTSelector	Struct	-	リモート接続パートナーの TSelector。 • バイト 20～21 = TSelLength • バイト 22～53: = TSel[1-32]
	TSelLength	UINT	-	値の範囲は 0～32 です。

	TSel	ARRAY [1..32] of BYTE	-	どの場合も値の範囲は 0～255
54 ... 87	LocalTSelector	Struct	-	リモート接続パートナーの TSelector。 • バイト 20～21 = TSelLength • バイト 22～53: = TSel[1-32]
	TSelLength	UINT		値の範囲は 0～32 です。
	TSel	ARRAY [1..32] of BYTE	-	どの場合も値の範囲は 0～255
88 … 89	CrRetransmition-Time	UINT	-	接続試行が繰り返される時間(秒単位)。
90 ... 91	DataRetransmi-tionTime	UINT	100 ms	データ転送が繰り返される時間(ミリ秒単位)。
92 ... 93	MaxRetransmition-Count	UINT	-	繰り返しの最大数。
94 ... 95	InactivityTime	UINT	-	単位は秒
96 ... 97	WindowTime	UINT		単位は秒

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	接続が正常に確立されました。
0	7000	ジョブ処理が行われていません。
0	7001	ジョブの実行を開始し、接続を確立します。
0	7002	接続を確立中です(REQ 対象外)。
1	8085	接続 ID (ID パラメータ)は、設定した接続によって既に使用されています。
1	8086	ID パラメータが有効範囲外です。
1	8087	最大接続数に達しました。これ以上の接続は行えません。
1	8089	CONNECT パラメータが接続記述子をポイントしていないか、または接続記述子が手動で作成されています。
1	809A	CONNECT パラメータの構造がサポートされていないか、長さが無効です。
1	809B	TCON_xxx 構造体内の工レメント InterfaceId が、CPU または CM/CP インターフェースのハードウェア識別子を参照しないか、値が「0」です。
1	80A2	ローカルまたはリモートポートがシステムによって使用中です。 以下のポートがローカルで予約済みです。 20、21、80、102、135、161、162、443、34962、34963、34964 および 49152～65535 の領域。

1	80A3	ID が、CONNECT パラメータで同一の接続記述子を使用する、ユーザープログラムで作成された接続によって使用されています。
1	80A4	接続のリモートエンドポイントの IP アドレスが無効であるか、またはローカルパートナーの IP アドレスと同じです。
1	80A7	通信エラー: 「TCON」の完了前に「TDISCON」を実行しました。
1	80B4	TCON_IP_RFC のみ: ローカルの T セレクタが指定されていないか、最初のバイトが値 0x0E を含んでいない(T セレクタの長さ = 2 の場合のみ)、またはローカルの T セレクタが「SIMATIC-」で始まっています。
1	80B5	接続タイプ 13 = UDP の場合、パッシブ接続の確立以外は許可されていません(構造体 TCON_IP_v4 / TCON_PARAM のパラメータ ActiveEstablished の値が TRUE です)。
1	80B6	接続記述子のデータブロックの ConnectionType パラメータにパラメータ割り当てエラーがあります。 • TCON_IP_v4 の場合のみ有効: 0x11、0x0B、および 0x13。 • TCON_IP_RFC の場合のみ有効: 0x0C および 0x12。
1	80B7	TCON_IP_v4 の場合: • TCP (アクティブ接続の確立): リモートポートは「0」です。 • TCP (パッシブ接続の確立): ローカルポートは「0」です。 • UDP: ローカルポートは「0」です。 TCON_IP_RFC の場合: • ローカル(LocalTSelector)またはリモート(RemoteTSelector)の T セレクタが、32 バイトを超える長さで指定されています。 • T セレクタ(ローカルまたはリモート)の TSelLength に対して、32 を超える長さが入力されています。 • 特定の接続パートナーのアドレス長のエラー。
1	80B8	ローカル接続記述子のパラメータ ID(CONNECT パラメータの構造体)と命令のパラメータ ID が異なります。
1	80C3	すべての接続リソースが使用中です。
1	80C4	一時的な通信エラー: • 現時点では、接続を確立できません。 • インターフェースが現在、新しいパラメータを受信中です。 • 構成された接続が、現在「TDISCON」命令によって削除されています。
1	80C5	接続パートナーが接続の確立を拒否したか、接続を終了したか、または能動的に接続を切断しました。
1	80C6	接続パートナーに到達できません(ネットワークエラー)。
1	80C7	実行がタイムアウトしました。
1	80C8	ID パラメータの値が、ユーザープログラムを使用して作成された接続によって既に使用されています。 接続は同じ ID を使用していますが、パラメータ CONNECT での接続設定が異なっています。
1	80C9	接続パートナーの検証に失敗しました。 接続を確立しようとしている接続パートナーが、CONNECT パラメータの構造の指定されたパートナーと一致していません。
1	80CE	ローカルインターフェースの IP アドレスが 0.0.0.0 です。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

TDISCON: 通信接続の終了

説明

「TDISCON」命令は、CPU から接続パートナーへの通信接続を終了します。

ファンクションの説明

「TDISCON」命令は、非同期で動作します。つまり、複数の呼び出しにわたってジョブが処理されます。 REQ = 1 で「TDISCON」命令を呼び出すことによって、接続を終了するためのジョブを開始します。

「TDISCON」が正常に実行された後、「TCON」に対して指定された ID は無効になるため、送受信に使用できません。

ジョブステータスは、出力パラメータ BUSY および STATUS によって示されます。 ここでは、STATUS は非同期命令の出力パラメータ RET_VAL に対応します(関連項目:**非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味**).

次の表に、BUSY、DONE、ERROR の関係を示します。 この表を使用して、「TDISCON」の現在のステータスや、接続の確立がいつ完了したかを認識できます。

BUSY	DONE	ERROR	説明
1	0	0	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータで確認できます。
0	0	0	命令に(新しい)ジョブが割り当てられませんでした。

パラメータ

次の表に、「TDISCON」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域		説明
			S7-1200	S7-1500	
REQ	Input	BOOL	I、Q、M、D、L、または定数	I、Q、M、D、L、または定数	制御パラメータ REQUEST は、IDによって指定された接続を終了するためのジョブを開始します。このジョブは信号立ち上がりエッジで開始します。
ID	Input	CONN_OUC (WORD)	D、L、または定数	I、Q、M、D、L、または定数	終了する接続の参照(接続 ID) 値の範囲 W#16#0001 ~ W#16#0FFF
DONE	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	ステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました

BUSY	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	ステータスパラメータ: • BUSY = 1: ジョブがまだ完了していません。 • BUSY = 0: ジョブがまだ完了していないか、開始していません。
ERROR	Output	BOOL	I、Q、M、D、L	I、Q、M、D、L	ステータスパラメータ: • ERROR=0: エラーはありません。 • ERROR=1: 処理中にエラーが発生しました。STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	I、Q、M、D、L	ステータスパラメータ: エラー情報(「パラメータ ERROR および STATUS」を参照)

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

パラメータ ERROR および STATUS

ERROR	STATUS*(W#16#...)	説明
0	0000	接続が正常に終了しました。
0	7000	ジョブ処理が行われていません。
0	7001	ジョブ処理を開始します。接続を終了中です。
0	7002	中間呼び出し(REQ は対象外)。接続を終了中です。
1	8086	ID パラメータが許容値の範囲内にありません。
1	80A3	存在していない接続を終了しようとしているか、接続が既に終了しています。
1	80C4	一時的な通信エラー: インターフェースに新しいパラメータを割り当て中であるか、接続が現在確立中です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

TSEND: 通信接続経由のデータ送信

この章には下記に関する情報が記載されています：

TSEND: 通信接続経由のデータ送信

説明

「TSEND」命令の以下の説明は、CPU S7-1200 のバージョン 3.0 以下に対して有効です。

「TSEND」命令を使用し、既存の通信接続を介してデータを送信します。 "TSEND」は非同期的に実行されます。

DATA パラメータで送信領域を指定します。 これには、アドレスと送信データの長さが含まれます。 BOOL および Array of BOOL を除くすべてのデータタイプを送信データに使用できます。

送信ジョブは、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。

LEN パラメータを使用して、送信ジョブで送信する最大バイト数を指定します。

- TCP(ストリーミングプロトコル)を使用してデータが転送されるとき、「TSEND」命令は、「TRCV」に送信されるデータの長さに関する情報を提供しません。
- ISO-on-TCP(メッセージ指向プロトコル)を使用してデータが転送されるとき、送信データの長さが「TRCV」に通知されます。 「TSEND」でパケットとして送信される量のデータが、受信側でも再度受信される必要があります(「TRCV」)。
  - 送信データに対して受信バッファが小さすぎる場合、受信側でエラーが発生します。
  - 受信バッファが十分に大きくない場合、データパケットが受信されると直ちに、「TRCV」は DONE=1 を返します。

送信データは、送信ジョブが完了するまで編集しないでください。 送信ジョブが正常に実行された場合、DONE パラメータが「1」にセットされます。DONE パラメータの信号状態が「1」であっても、送信データが通信パートナーに読み取られているとは限りません。

パラメータ

次の表に、「TSEND」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで送信ジョブを開始します。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	「TCON」で確立された接続への参照。 値の範囲:W#16#0001 ～W#16#0FFF
LEN	Input	UINT	I、Q、M、D、L、または定数	ジョブで送信する最大バイト数。
DATA	InOut	VARIANT	I、Q、M、D	アドレスと送信データの長さの入っている送信領域へのポインタ アドレスは、以下を参照します。 • 入力のプロセスイメージ • 出力のプロセスイメージ • ビットメモリ • データブロック
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ:

				• 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

LEN および DATA パラメータ

- LEN = 0 の場合、DATA パラメータで指定されたすべてのデータが送信されます。
- LEN パラメータでのバイト数が、DATA パラメータで定義された送信データの長さよりも大きい場合、エラーコード 8088 が STATUS パラメータに出力されます(以下の STATUS パラメータの説明を参照)。
- 構造体(Struct)が DATA パラメータを介して参照される場合、LEN をこの構造体よりも短くすることができます。 この場合、LEN パラメータの長さまでのデータしか転送されません。
- データタイプ STRING および WSTRING では、パラメータ LEN = 0 の場合にすべてのデータが転送されます。 LEN > 0 の場合、長さは少なくとも最大バイト数に、長さ情報を含む 2 バイトを追加したものである必要があります。 データタイプの構造体に関する詳細情報については、次の項目を参照してください。 "有効なデータタイプの概要".
- 送信できる最大バイト数は 65534 です。
- 最適化された DB から構造体タグを使用する場合、構造体タグのアドレスは DATA パラメータおよびパラメータ LEN = 0 で相互接続される必要があります。これにより、受信側で同じ構造体が使用された場合に、構造体全体のタイプセーフな伝送が保証されます。

BUSY、DONE および ERROR パラメータ

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、ジョブが正常に実行されたかどうかをチェックできます。 「TSEND」の実行中にエラーが発生すると、ERROR パラメータがセットされます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ER-ROR	説明
1	0	0	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータに明記されています。

0	0	0	新しいジョブが割り当てられませんでした。

注記
「TSEND」は非同期的に実行されるため、DONE パラメータまたは ERROR パラメータが値「1」に変更されるまで、送信領域内のデータの整合性を保持する必要があります。

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	送信ジョブがエラーなしで完了しました。
0	7000	ジョブ処理が行われていません。
0	7001	ジョブの実行が開始され、データが送信中です。 このジョブの処理時に、オペレーティングシステムが DATA 送信領域内のデータにアクセスします。
0	7002	ジョブを実行中です(REQ は対象外)。 このジョブの処理時に、オペレーティングシステムが DATA 送信領域内のデータにアクセスします。
1	8085	• LEN パラメータが最大許容値よりも大きくなっています(65536)。 • DATA および LEN パラメータの値が両方とも「0」です。
1	8086	ID パラメータが許容アドレス範囲(1～0xFFF)外です。
1	8088	LEN パラメータが、DATA で指定された領域よりも大きくなっています。
1	80A1	通信エラー: • 指定された通信がまだ確立されていません。 • 指定された接続を終了中です。 この接続による転送は行えません。 • インターフェースを再初期化中です。
1	80B3	プロトコルの種類(接続記述子内の ConnectionType パラメータ)が UDP に設定されています。 UDP 接続では命令「TUSEND」を使用します。
1	80C3	• この ID を持つブロックは、別の優先度グループで既に処理中です。 • リソースが内部的に不足しています。
1	80C4	一時的な通信エラー: • 現時点では、パートナーに対する接続を確立できません。 • インターフェースが新しいパラメータの設定を受信中であるか、接続が確立中です。
1	80C5	接続が通信パートナーによって終了されました。
1	80C6	ネットワークエラー。 通信パートナーに到達できません。
1	80C7	実行がタイムアウトしました。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

TSEND: 通信接続経由のデータ送信

説明

「TSEND」命令の以下の説明は、CPU S7-1500 および S7-1200 V4.0 以降に対して有効です。

「TSEND」命令を使用し、既存の通信接続を経由してデータを送信します。 "TSEND」は非同期的に実行されます。

DATA パラメータで送信領域を指定します。 これには、アドレスと送信データの長さが含まれます。 BOOL および Array of BOOL を除くすべてのデータタイプを送信データに使用できます。

送信ジョブは、REQ パラメータで信号立ち上がりエッジが検出されたときに実行されます。

LEN パラメータを使用して、送信ジョブで送信する最大バイト数を指定します。

- TCP(ストリーミングプロトコル)を使用してデータが転送されるとき、「TSEND」命令は、「TRCV」に送信されるデータの長さに関する情報を提供しません。
- ISO-on-TCP(メッセージ指向プロトコル)を使用してデータが転送されるとき、送信データの長さが「TRCV」に通知されます。 「TSEND」でパケットとして送信される量のデータが、受信側でも再度受信される必要があります(「TRCV」)。
  - 送信データに対して受信バッファが小さすぎる場合、受信側でエラーが発生します。
  - 受信バッファが十分に大きくない場合、データパケットが受信されると直ちに、「TRCV」は DONE=1 を返します。

送信データは、送信ジョブが完了するまで編集しないでください。 送信ジョブが正常に実行された場合、DONE パラメータが「1」にセットされます。 DONE パラメータのシグナル状態が「1」であっても、送信データが通信パートナーに読み取られているとは限りません。

パラメータ

以下の表に、「TSEND」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで送信ジョブを開始します。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	「TCON」で確立された接続への参照。 値の範囲: W#16#0001 終了 W#16#0FFF
LEN	Input	UDINT	I、Q、M、D、L、または定数	ジョブで送信される最大バイト数(S7-1200 の最大許容値: 8192、S7-1500 の場合の最大値:65536)。
DATA	InOut	VARIANT	I、Q、M、D、L	アドレスと送信データの長さの入っている送信領域へのポインタ アドレスは、以下を参照します。 • 入力のプロセスイメージ • 出力のプロセスイメージ • ビットメモリ • データブロック • ローカルデータ

DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

LEN および DATA パラメータ

- LEN = 0 の場合、DATA パラメータで指定されたすべてのデータが送信されます。
- LEN パラメータでのバイト数が、DATA パラメータで定義された送信データの長さよりも大きい場合、エラーコード 8088 が STATUS パラメータに出力されます(以下の STATUS パラメータの説明を参照)。
- 構造体(Struct)が DATA パラメータによって参照される場合、LEN をこの構造体よりも短くすることができます。 この場合、LEN パラメータの長さまでのデータ以外は転送されません。
- データタイプ STRING および WSTRING では、パラメータ LEN = 0 の場合にすべてのデータが転送されます。 LEN > 0 の場合、長さは少なくとも最大バイト数に、長さ情報を含む 2 バイトを追加したものである必要があります。 データタイプの構造体に関する詳細情報については、次の項目を参照してください。 "有効なデータタイプの概要".
- 送信できる最大バイト数は、デバイスに依存します。
- 最適化された DB から構造体タグを使用する場合、構造体タグのアドレスは DATA パラメータおよびパラメータ LEN = 0 で相互接続される必要があります。これにより、受信側で同じ構造体が使用された場合に、構造体全体のタイプセーフな伝送が保証されます。

パラメータ BUSY、DONE および ERROR

BUSY、DONE、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、ジョブが正常に実行されたかどうかをチェックできます。「TSEND」の実行中にエラーが発生すると、ERROR パラメータがセットされます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ER-ROR	説明
1	0	0	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。

0	0	1	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータに明記されています。
0	0	0	新しいジョブが割り当てられませんでした。

注記
「TSEND」は非同期的に実行されるため、DONE パラメータまたは ERROR パラメータが値「1」に変更されるまで、送信領域内のデータの整合性を保持する必要があります。

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	送信ジョブがエラーなしで完了しました。
0	7000	ジョブ処理が行われていません。
0	7001	ジョブの実行が開始され、データが送信中です。 このジョブの処理時に、オペレーティングシステムが DATA 送信領域内のデータにアクセスします。
0	7002	ジョブを実行中です(REQ は対象外)。 このジョブの処理時に、オペレーティングシステムが DATA 送信領域内のデータにアクセスします。
1	8085	• LEN パラメータの値が最大許容値よりも大きくなっています(S7-1200 の場合: 8192、S7-1500 の場合: 65536). • DATA および LEN パラメータの値が両方とも「0」です。
1	8086	ID パラメータが許容アドレス範囲(1～0xFFF)外です。
1	8088	LEN パラメータが、DATA で指定された領域よりも大きくなっています。
1	80A1	通信エラー: • 指定された通信がまだ確立されていません。 • 指定された接続を終了中です。 この接続による転送は行えません。 • インターフェースを再初期化中です。
1	80B1	現在のジョブを終了する前に DATA パラメータが変更されました。
1	80B3	プロトコルの種類(接続記述子内の ConnectionType パラメータ)が UDP に設定されています。 UDP 接続では命令「TUSEND」を使用します。
1	80C3	• この ID を持つブロックは、別の優先度グループで既に処理中です。 • リソースが内部的に不足しています。
1	80C4	一時的な通信エラー: • 現時点では、パートナーに対する接続を確立できません。 • インターフェースが新しいパラメータの設定を受信中であるか、接続が確立中です。
1	80C5	接続が通信パートナーによって終了されました。
1	80C6	ネットワークエラー。 通信パートナーに到達できません。
1	80C7	実行がタイムアウトしました。

* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。

TRCV: 通信接続経由のデータ受信

この章には下記に関する情報が記載されています：

- TRCV: 通信接続経由のデータ受信 (S7-1200)
- TRCV: 通信接続経由のデータ受信 (S7-1200, S7-1500)

TRCV: 通信接続経由のデータ受信

説明

「TRCV」命令の以下の説明は、CPU S7-1200 のバージョン 3.0 以下に対して有効です。

「TRCV」命令を使用し、既存の通信接続を介してデータを受信します。 "TRCV」は非同期的に実行されます。

EN_R パラメータが値「1」にセットされると、データの受信が可能になります。 受信データは、受信領域に入力されます。 使用されているプロトコル変数に応じて、受信エリアの長さを LEN パラメータ(LEN <> 0 の場合)、または DATA パラメータの長さの情報(LEN = 0 の場合)で指定します。

データを受信している間、受信されたデータの整合性を確保するために、DATA パラメータまたは定義済み受信領域に対して変更を行うことはできません。

データが正常に受信されると、NDR パラメータが値「1」にセットされます。 RCVD_LEN パラメータで、実際に受信したデータの量を問い合わせることができます。

「TRCV」の受信モード

次の表に、受信データがどのように受信領域に入力されるかを示します。

プロトコルの種類	受信領域内のデータの可用性	接続記述子の connection_type*パラメータ	LEN パラメータ
TCP (アドホックモード)	データはすぐに使用可能です。	16 進数値: B#16#11 整数値: 17	0
TCP (指定された長さのデータの受信)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#11 整数値: 17	1 ~ 8192
ISO on TCP (メッセージ指向のデータ転送)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#12 整数値: 18	• 1 ~ 1452、CP を使用する場合。 • 1 ~ 8192、CP を使用しない場合。
* 「TCON_Param に従う構造の接続パラメータ」を参照してください。			

TCP (アドホックモード)

アドホックモードは、TCP プロトコル種類でのみ使用可能です。 「TRCV」命令で長さが動的なデータを受信するには、アドホックモードを使用します。

アドホックモードは、値「0」を LEN パラメータに割り当てることで設定できます。 アドホックモードを使用すると、データブロックに対して標準アクセスですべてのデータタイプを使用できます。 最適化されたアクセスのあるデータブロックには、ARRAY of BYTE または長さが 8 ビットのデータタイプのみを使用できます(CHAR、USINT、SINT など)。 アドホックモードが有効の場合、データの受信は、NDR パラメータの受信バイトの後に、既に信号で示されています。

TCP (指定された長さのデータの受信)

指定された長さのデータの受信の場合、LEN パラメータにデータの長さを入力します。 LEN パラメータで指定された長さのデータが完全に受信されるまで、データの受信は完了しません。 受信領域で

データが使用可能になるのは、そのときだけです(DATA パラメータ)。 データの受信は、NDR 出力パラメータによって信号で示されます。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

ISO on TCP (メッセージ指向のデータ転送)

プロトコルのバリエーション ISO on TCP を使用する接続を介して、メッセージブロック全体が送信されます。これらは受信によってそのように認識されます。 ISO on TCP を使用している場合、メッセージブロックが完全に受信されると直ちに、「TRCV」がデータの受信を信号で示します。 受信領域は、LEN および DATA パラメータによって定義されます。 送信データに対して受信バッファ(DATA パラメータ)が小さすぎる場合、「TRCV」がエラーを信号で示します。 データの正常な受信は、NDR 出力パラメータによって信号で示されます。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

パラメータ

次の表に、「TRCV」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、または定数	受信可能です。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	「TCON」で確立された接続への参照。 値の範囲 W#16#0001 (1) ~ W#16#0FFF (4095)
LEN	Input	UDINT	I、Q、M、D、L、または定数	受信領域のバイト単位の長さ(非表示)。 DATA パラメータで最適化されたアクセスのあるメモリ領域を使用する場合、LEN パラメータの値は「0」である必要があります。
DATA	InOut	VARIANT	I、Q、M、D	受信領域へのポインタ
NDR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ(New Data Received): • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブがエラーなしで完了済み
BUSY	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: エラーは発生していません。 • 1: 命令の実行中にエラーが発生しました。 詳細情報は、STATUS パラメータで出力されます。
STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ:

				ステータスおよびエラー情報の出力。
RCVD_LEN	Output	UDINT	I、Q、M、D、L	実際に受信されたバイト単位のデータ量

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

LEN、DATA および RCVD_LEN パラメータ

- LEN = 0 の場合、受信データは、DATA パラメータで指定された受信領域に保存されます。 受信したバイト数は、RCVD_LEN パラメータで示されます。
- LEN パラメータで指定された長さが、DATA パラメータで受信したデータの長さよりも大きい場合、エラーコード 8088 が STATUS パラメータに出力されます(以下の STATUS パラメータの説明を参照)。
- 構造体(Struct)が DATA パラメータを介して参照される場合、LEN をこの構造体よりも短くすることができます。 この場合、LEN パラメータの長さまでのデータしか転送されません。
- DATA パラメータが最適化されたアクセスでデータブロックを参照する場合、LEN パラメータを「0」にセットする必要があります。
- STRING データタイプが DATA パラメータを介して参照される場合、LEN パラメータで指定される長さを 0 または 2 以上にする必要があります(LEN = 1 は許可されません)。
- WSTRING データタイプが DATA パラメータを介して参照される場合、LEN パラメータで指定される長さを 0 または 5 以上にする必要があります。

BUSY、NDR および ERROR パラメータ

BUSY、NDR、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 NDR パラメータで、ジョブが正常に実行されたかどうかをチェックできます。 TRCV の実行中にエラーが発生すると、ERROR パラメータがセットされます。エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、NDR、ERROR パラメータの関係を示します。

BUSY	NDR	ER-ROR	説明
1	-	-	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 STATUS パラメータにエラーの原因が出力されます。
0	0	0	新しいジョブが割り当てられませんでした。

注記

「TRCV」は非同期的に実行されるため、受信領域内のデータは、NDR パラメータが値「1」にセットされたときにのみ整合性があります。

ERROR および STATUS パラメータ

ERROR	STATUS* (W#16#...)	説明
0	0000	ジョブが正常に完了しました。 受信データの現在の長さが RCVD_LEN パラメータに出力されます。
0	7000	ブロックの受信準備ができていません。
0	7001	ブロックの受信準備が完了し、受信ジョブが有効になりました。
0	7002	中間呼び出し。受信ジョブを実行中です。 注記: ジョブが処理されている間に、データが受信領域に書き込まれます。 この間の受信領域へのアクセスでは、矛盾したデータが提供される場合があります。
1	8085	• LEN パラメータが最大許容値よりも大きくなっています。 • 最初の呼び出しの後に、LEN または DATA パラメータの値が変更されました。 • LEN パラメータおよび DATA パラメータの値が「0」であるか、または LEN が最大許容値(65536)よりも大きな値です。
1	8086	ID パラメータが許容アドレス範囲(1 ~ 0x0FFF)外です。
1	8088	• 受信領域が小さすぎます。 • LEN パラメータの値が、DATA パラメータで設定された受信領域よりも大きくなっています。
1	80A1	通信エラー: • 指定された通信がまだ確立されていません。 • 指定された接続を終了中です。 この接続による受信ジョブは行えません。 • 接続を最初期化中です。
1	80B3	プロトコルの種類(接続記述子内の connection_type パラメータ)が UDP に設定されています。 UDP 接続では命令「TURCV」を使用します。
1	80C3	• この ID を持つブロックは、別の優先度グループで既に処理中です。 • リソースが内部的に不足しています。
1	80C4	一時的な通信エラー: • 現時点では、パートナーに対する接続を確立できません。 • インターフェースが新しいパラメータの設定を受信中であるか、接続が確立中です。
1	80C5	リモートパートナーによって接続が終了しました。
1	80C6	リモートパートナーに到達できません(ネットワークエラー)。
1	80C7	実行がタイムアウトしました。
1	80C9	受信領域の長さが送信データの長さよりも小さくなっています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

TRCV: 通信接続経由のデータ受信

説明

「TRCV」命令の以下の説明は、CPU S7-1500 および S7-1200 V4.0 以降に対して有効です。

「TRCV」命令を使用し、既存の通信接続を経由してデータを受信します。 "TRCV」は非同期的に実行されます。

EN_R パラメータが値「1」にセットされると、データの受信が可能になります。 受信データは、受信領域に入力されます。 使用されているプロトコル変数に応じて、受信エリアの長さを LEN パラメータ(LEN <> 0 の場合)、または DATA パラメータの長さの情報(LEN = 0 の場合)で指定します。

データを受信している間、受信されたデータの整合性を確保するために、DATA パラメータまたは定義済み受信領域に対して変更を行うことはできません。

データが正常に受信されると、NDR パラメータが値「1」にセットされます。 RCVD_LEN パラメータで、実際に受信したデータの量を照会できます。

「TRCV」の受信モード

次の表に、受信データがどのように受信領域に入力されるかを示します。

プロトコルの種類	ADHOC パラメータ	受信領域内のデータの可用性	接続記述子の connection_type パラメータ	パラメータ LEN
TCP (アドホックモード)	1 (アドホック有効)	データはすぐに使用可能です。	16 進数値: B#16#11 整数値: 17	1 から最大長(CPU によって異なる)
TCP (指定された長さのデータの受信)	0 (アドホック無効)	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#11 整数値: 17	1 ~ 8192
ISO on TCP (メッセージ指向のデータ転送)	-	LEN パラメータで指定されたデータ長が完全に受信されるとすぐに、データが使用可能になります。	16 進数値: B#16#12 整数値: 18	• 1 ~ 1452、CP を使用する場合。 • 1 ~ 8192、CP を使用しない場合。

TCP (アドホックモード)

アドホックモードは、TCP プロトコル種類でのみ使用可能です。 「TRCV」命令で長さが動的なデータを受信するには、アドホックモードを使用します。

アドホックモードは、値「1」を ADHOC パラメータに割り当てることで設定できます。 アドホックモードを使用すると、データブロックに対して標準アクセスですべてのデータタイプを使用できます。 最適化されたアクセスのあるデータブロックには、ARRAY of BYTE または長さが 8 ビットのデータタイプのみを使用できます(CHAR、USINT、SINT など)。 アドホックモードが有効の場合、データの受信は、NDR パラメータの受信バイトの後に、既に信号で示されています。

TCP (指定された長さのデータの受信)

指定された長さのデータの受信の場合、ADHOC パラメータに値「0」を割り当てます。 アドホックモードが無効の場合、LEN パラメータで指定された長さのデータが完全に受信されるまで、データの受信は完了しません。 受信領域でデータが使用可能になるのは、そのときだけです(DATA パラメータ)。 データの正常な受信は、NDR 出力パラメータによって信号で示されます。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

ISO on TCP (メッセージ指向のデータ転送)

プロトコルのバリエーション ISO on TCP を使用する接続を経由して、メッセージブロック全体が送信されます。これらは受信によってそのように認識されます。 ISO on TCP を使用している場合、メッセージブロックが完全に受信されると直ちに、「TRCV」がデータの受信を信号で示します。 受信領域は、LEN および DATA パラメータによって定義されます。 送信データに対して受信バッファ(DATA パラメータ)が小さすぎる場合、「TRCV」がエラーを信号で示します。 データの正常な受信は、NDR 出力パラメータによって信号で示されます。 RCVD_LEN パラメータの実際に受信されたデータ長(バイト単位)は、受信後の LEN パラメータのデータ長に相当します。

パラメータ

以下の表に、「TRCV」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、または定数	受信可能です。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	「TCON」で確立された接続への参照。 値の範囲:W#16#0001 終了 W#16#0FFF
LEN	Input	UDINT	I、Q、M、D、L、または定数	受信領域のバイト単位の長さ(非表示)(S7-1200 の場合の最大値: 8192、S7-1500 の場合の最大値: 65536). DATA パラメータで最適化されたアクセスのある受信領域を使用する場合、LEN パラメータの値は「0」である必要があります。
ADHOC	Input	BOOL	I、Q、M、D、L、または定数	TCP プロトコルのバリエーションの場合は、アドホックモードを使用します(非表示)。
DATA	InOut	VARIANT	I、Q、M、D、L	受信領域へのポインタ
NDR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ(New Data Received): • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: 新しいデータが受信されました
BUSY	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または既に完了しています。 • 1: ジョブがまだ完了していません。 新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR: • 0: エラーは発生していません。

				• 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス
RCVD_LEN	Output	UDINT	I、Q、M、D、L	実際に受信されたバイト単位のデータ量

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ LEN、DATA および RCVD_LEN

- LEN = 0 の場合、受信データは、DATA パラメータで指定された受信領域に保存されます。 受信したバイト数は、RCVD_LEN パラメータで示されます。
- LEN パラメータで指定された長さが、DATA パラメータで受信したデータの長さよりも大きい場合、エラーコード 8088 が STATUS パラメータに出力されます(以下の STATUS パラメータの説明を参照)。
- 構造体(Struct)が DATA パラメータによって参照される場合、LEN をこの構造体よりも短くすることができます。 この場合、LEN パラメータの長さまでのデータ以外は転送されません。
- DATA パラメータが最適化されたアクセスでデータブロックを参照する場合、LEN パラメータを「0」にセットする必要があります。 基本データタイプのデータの長さが一致しない場合、データは受信されず、エラーコード 8088 が STATUS パラメータに出力されます。
- STRING データタイプが DATA パラメータによって参照される場合、LEN パラメータで指定される長さを 0 または 2 以上にする必要があります(LEN = 1 は許可されません)。
- WSTRING データタイプが DATA パラメータによって参照される場合、LEN パラメータで指定される長さを 0 または 5 以上にする必要があります。

パラメータ BUSY、NDR および ERROR

BUSY、NDR、ERROR および STATUS パラメータを使用してジョブのステータスを確認できます。 BUSY パラメータは、処理ステータスを示します。 NDR パラメータで、ジョブが正常に実行されたかどうかをチェックできます。 TRCV の実行中にエラーが発生すると、ERROR パラメータがセットされます。 エラー情報が STATUS パラメータに出力されます。

次の表に、BUSY、NDR、ERROR パラメータの関係を示します。

BUSY	NDR	ERROR	説明
1	-	-	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 STATUS パラメータにエラーの原因が出力されます。
0	0	0	新しいジョブが割り当てられませんでした。

注記
「TRCV」は非同期的に実行されるため、受信領域内のデータは、NDR パラメータが値「1」にセットされたときにのみ整合性があります。

ERROR および STATUS パラメータ

ERROR	STATUS*	説明

	(W#16#...)	
0	0000	ジョブが正常に完了しました。 受信データの現在の長さが RCVD_LEN パラメータに出力されます。
0	7000	ブロックの受信準備ができていません。
0	7001	ブロックの受信準備が完了し、受信ジョブが有効になりました。
0	7002	中間呼び出し。受信ジョブを実行中です。 注記: ジョブが処理されている間に、データが受信領域に書き込まれます。 この間の受信領域へのアクセスでは、矛盾したデータが提供される場合があります。
1	8085	• LEN パラメータの値が最大許容値よりも大きくなっています(S7-1200 の場合: 8192 バイト、S7-1500 の場合: 65536 バイト)。 • 最初の呼び出しの後に、LEN または DATA パラメータの値が変更されました。 • LEN パラメータおよび DATA パラメータの値が「0」であるか、または LEN が最大許容値(S7-1200 の場合)よりも大きな値です。 8192 バイト、S7-1500 の場合: 65536 バイト)。
1	8086	ID パラメータが許容値の範囲(1 .. 0x0FFF)外です。
1	8088	• 受信領域が小さすぎます。 • LEN パラメータの値が、DATA パラメータで設定された受信領域よりも大きくなっています。
1	80A1	通信エラー: • 指定された通信がまだ確立されていません。 • 指定された接続を終了中です。 この接続による受信ジョブは行えません。 • 接続を最初期化中です。
1	80B1	現在のジョブを終了する前に DATA パラメータが変更されました。
1	80B3	プロトコルの種類(接続記述子内の connection_type パラメータ)が UDP に設定されています。 UDP 接続では命令「TURCV」を使用します。
1	80C3	• この ID を持つブロックは、別の優先度グループで既に処理中です。 • リソースが内部的に不足しています。
1	80C4	一時的な通信エラー: • 現時点では、パートナーに対する接続を確立できません。 • インターフェースが新しいパラメータの設定を受信中であるか、接続が確立中です。
1	80C5	リモートパートナーによって接続が終了しました。
1	80C6	リモートパートナーに到達できません(ネットワークエラー)。
1	80C7	実行がタイムアウトしました。
1	80C9	受信領域の長さが送信データの長さよりも小さくなっています。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。		

UDP でのリモートパートナーのアドレス情報の構造体

概要

UDP 接続の使用時、リモートパートナーのアドレス情報が、システムデータタイプ TADDR_Param に格納されます。

- 命令「TUSEND」を使用して、宛先のアドレス情報が、TADDR_Param 経由でパラメータ ADDR に割り当てられます。

  リモートパートナーの格納されたアドレスデータが、この命令によって、システムデータタイプから読み出されます。
- 命令「TURCV」を使用して、送信元のアドレスが、TADDR_Param 経由で、パラメータ ADDR で受信されます。

  アドレスデータが、この命令によって、システムデータタイプに書き込まれます。

TADDR_Param に従ったアドレス情報の構造

システムデータタイプ TADDR_Param は、リモートパートナーのアドレス情報を収納し、IP アドレスとポート番号から構成されます。

システムデータタイプ TADDR_Param の構造を以下に示します。

バイト	パラメータ	データタイプ	開始値	説明
0 ~ 3	rem_ip_addr	ARRAY [1..4] of USINT	B#16#00 ...	リモートパートナーの IP アドレス。たとえば 192.168.002.003 の場合: • rem_ip_addr[1] = B#16#C0 (192) • rem_ip_addr[2] = B#16#A8 (168) • rem_ip_addr[3] = B#16#02 (002) • rem_ip_addr[4] = B#16#03 (003) IP アドレスは、リモートパートナーのインターフェースプロパティのデバイス & ネットワークビューから取得できます。 別の方法として、IP アドレスは、UDP 接続のプロパティのアドレス詳細にも表示されます。
4 ~ 5	rem_port_nr	UINT	B#16#00 ...	リモートポート番号(指定可能な値については「ポート番号の割り当て): • 」を参照 rem_port_nr[1] = 16 進数のポート番号の上位バイト • rem_port_nr[2] = 16 進数のポート番号の下位バイト ポート番号は、UDP 接続プロパティのデバイス & ネットワークビューから取得できます。 ポート番号は、アドレス詳細に 10 進数値として表示されます。 例: ポート番号 = 2000 (10 進数) / W#16#07D0 (16 進数)

				• rem_port_nr[1] = 07 (上位バイト) • rem_port_nr[2] = D0 (下位バイト)
6 ~ 7	reserved	WORD	B#16#00 ...	未使用。 このパラメータの値は「0」のままです。

データブロックの TADDR_Param の作成

TADDR_Param を作成する場合、以下のオプションがあります。

- 新しいデータブロックを作成し、ダイアログ[新しいデータブロックの追加]で、タイプとして TADDR_Param を選択する。
- 既存のデータブロックを開き、新しいタグを作成して、列データタイプ TADDR_Param に入力する。

データブロックには、複数のシステムデータタイプ TADDR_Param を収納できます。

TUSEND: イーサネット(UDP)経由のデータ送信

説明

「TUSEND」命令は、UDP を使用して、ADDR パラメータによってアドレス指定されたリモートパートナーにデータを送信します。

警告

UDP 経由でのデータ転送

RFC 768 規格の UDP 経由でリモートパートナーに転送されるデータは確認なしで送信されるため、セキュアではありません。 つまり、ブロックで示されずにデータが失われる可能性があります。

注記

異なるパートナーに対する順次送信操作の場合、「TUSEND」を呼び出すときに ADDR パラメータを調整するだけで済みます。 「TCON」および「TDISCON」命令を再度呼び出す必要はありません。 特定のリモートパートナーがデータを受信するためには、このパートナーで UDP ポートを設定する必要があります。

ファンクションの説明

「TUSEND」命令は、非同期で動作します。つまり、複数の呼び出しにわたってジョブが処理されます。 もう一度接続を確立するには、REQ パラメータで信号立ち上がりエッジを作成します。

出力パラメータ BUSY、DONE、ERROR および STATUS は、ジョブのステータスを示します。

関連項目: 非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味.

次の表に、BUSY、DONE、ERROR の関係を示します。 この表を使用して、「TUSEND」の現在のステータスや、送信プロセスがいつ終了したかを判別できます。

BUSY	DONE	ERROR	説明
TRUE	FALSE	FALSE	ジョブが処理中です。
FALSE	TRUE	FALSE	ジョブが正常に完了しました。
FALSE	FALSE	TRUE	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータで確認できます。
FALSE	FALSE	FALSE	命令に(新しい)ジョブが割り当てられませんでした。

注記

「TUSEND」の非同期操作が行われるため、DONE パラメータまたは ERROR パラメータの値が TRUE になるまで、送信領域内のデータの整合性を保持してください。

パラメータ

以下の表に、「TUSEND」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ REQUEST は、信号立ち上がりエッジで送信ジョブを開始します。 DATA および LEN によって指定された領域からデータが転送されます。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	ユーザープログラムとオペレーティングシステムの通信レイヤ間の関連する接続への参照。 ID は、命令「TCON」を使用したローカル接続記述子の関連するパラメータ ID と同じである必要があります。 値の範囲:W#16#0001 終了 W#16#0FFF
LEN	Input	UINT	I、Q、M、D、L、または定数	ジョブで送信するバイト数 値の範囲: 1 ~ 1472
DONE	Output	BOOL	I、Q、M、D、L	ステータスパラメータ DONE: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました。 この値は、1 つのサイクルに対してのみ表示されます。
BUSY	Output	BOOL	I、Q、M、D、L	• BUSY = 1: ジョブがまだ完了していません。 新しいジョブをトリガできません。 • BUSY = 0: ジョブが完了しました。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR: • ERROR=1: 処理中にエラーが発生しました。STATUS にエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ STATUS: エラー情報
DATA	InOut	VARIANT	I、Q、M、D	送信領域にはアドレスと長さが含まれます。アドレスは以下を参照します。 • 入力のプロセスイメージ • 出力のプロセスイメージ • ビットメモリ • データブロック
ADDR	InOut	VARIANT	D	システムデータタイプ TADDR_Param へのポインタ。 リモートパートナーのアドレス情報(IP アドレスおよびポート番号)をシステムデータタイプ TADDR_Param のデータブロックに格納します。 関連項目: UDP でのリモートパートナーのアドレス情報の構造体

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

ERROR	STATUS*(W#16#...)	説明
0	0000	送信ジョブがエラーなしで完了しました。
0	7000	ジョブ処理が行われていません。
0	7001	ジョブ処理が開始され、データを送信中です。 注記: この処理フェーズ中に、オペレーティングシステムが DATA 送信領域内のデータにアクセスします。
0	7002	中間呼び出し(REQ は対象外)。ジョブを処理中です。 注記: この処理フェーズ中に、オペレーティングシステムが DATA 送信領域内のデータにアクセスします。
1	8085	LEN パラメータの値が「0」であるか、最大許容値よりも大きくなっています。
1	8086	ID パラメータが許容値の範囲内にありません。
0	8088	LEN パラメータが、DATA で指定されたメモリ領域よりも大きくなっています。
1	8089	ADDR パラメータが、TADDR_Param 構造のデータブロックをポイントしていません。
1	80A1	通信エラー: • ユーザープログラムとオペレーティングシステムの通信レイヤ間の指定された接続がまだ確立されていません。 • ユーザープログラムとオペレーティングシステムの通信レイヤ間の指定された接続を現在終了中です。 この接続を経由した送信はできません。 • インターフェースを再初期化中です。
1	80B1	現在のジョブを終了する前に DATA パラメータが変更されました。
1	80A4	リモート接続エンドポイントの IP アドレス(ADDR パラメータ)が無効です。ローカルパートナー自身の IP アドレスと同一である可能性があります。
1	80B3	• プロトコルの種類(接続記述子内の connection_type パラメータ)が UDP に設定されていません。 「TSEND」を使用してください。 • ADDR パラメータ。 ポート番号に関する情報が無効です。
1	80B7	パラメータ ADDR によって参照される構造体の長さは 8 バイトではありません。
1	80C3	• この ID を持つブロックは、別の優先度クラスで既に処理中です。 • リソースが内部的に不足しています。
1	80C4	一時的な通信エラー: • 現時点では、ユーザープログラムとオペレーティングシステムの通信レイヤ間の接続を確立できません。 • 新しいパラメータ設定がインターフェースに割り当てられています。
1	80C5	リモートパートナーによって接続が終了しました。
1	80C6	リモートパートナーに到達できません(ネットワークエラー)。
1	80C7	実行がタイムアウトしました。
-	一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得

* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。

TURCV: イーサネット(UDP)を介したデータ受信

説明

「TURCV」命令は、UDP を経由してデータを受信します。 「TURCV」が正常に完了すると、ADDR パラメータはリモートパートナー(送信者)のアドレスを表示します。

	**警告**
セキュアでないデータ転送 RFC 768 規格の UDP 経由でリモートパートナーに転送されるデータは確認なしで送信されるため、セキュアではありません。 つまり、ブロックで示されずにデータが失われる可能性があります。	

ファンクションの説明

「TURCV」命令は、非同期で動作します。つまり、複数の呼び出しにわたってジョブが処理されます。 EN_R = 1 で「TURCV」を呼び出すことによって、受信ジョブを開始します。

出力パラメータ BUSY、DONE、ERROR および STATUS は、ジョブのステータスを示します。

関連項目: **非同期命令でのパラメータ REQ、RET_VAL および BUSY の意味**.

次の表に、BUSY、NDR、ERROR の関係を示します。 この表を使用して、TURCV の現在のステータスや、受信プロセスがいつ終了したかを判別できます。

BUSY	NDR	ERROR	説明
TRUE	FALSE	FALSE	ジョブが処理中です。
FALSE	TRUE	FALSE	ジョブが正常に完了しました。 新しいデータが受信されました。
FALSE	FALSE	TRUE	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータで確認できます。
FALSE	FALSE	FALSE	命令に(新しい)ジョブが割り当てられませんでした。

注記
「TURCV」の非同期操作が行われるため、受信領域内のデータは、NDR パラメータの値が TRUE であるときにのみ整合性があります。

パラメータ

以下の表に、「TURCV」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ enabled to receive EN_R = 1 の場合、「TURCV」は受信する準備ができています。 受信ジョブを処理中です。
ID	Input	WORD	I、Q、M、D、L、または定数	ユーザープログラムとオペレーティングシステムの通信レイヤ間の関連する接続への参照。

				ID は、ローカル接続記述子の対応するパラメータ ID と同じである必要があります。 値の範囲:W#16#0001 終了 W#16#0FFF
LEN	Input	UINT	I、Q、M、D、L、または定数	受信領域のバイト単位の長さ: 0 (推奨)または 1 ～1472
NDR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ NDR: • NDR = 0: ジョブがまだ開始されていないか、または実行中です。 • NDR = 1: ジョブが正常に完了しました。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ ERROR: • ERROR=1: 処理中にエラーが発生しました。 STATUS にエラーの種類に関する詳細情報を表示します。
BUSY	Output	BOOL	I、Q、M、D、L	• BUSY = 1: ジョブがまだ完了していません。新しいジョブをトリガできません。 • BUSY = 0: ジョブが完了しました。
STATUS	Output	WORD	I、Q、M、D、L	ステータスパラメータ STATUS: エラー情報
RCVD_LEN	Output	UINT	I、Q、M、D、L	実際に受信されたバイト単位のデータ量
DATA	InOut	VAR-IANT	I、Q、M、D、L	受信領域 アドレスは、以下を参照します。 • 入力のプロセスイメージ • 出力のプロセスイメージ • ビットメモリ • データブロック
ADDR	InOut	VAR-IANT	D	システムデータタイプ TADDR_Param へのポインタ。 リモートパートナーのアドレス情報(IP アドレスおよびポート番号)がシステムデータタイプ TADDR_Param のデータブロックに書き込まれます。 関連項目: UDP でのリモートパートナーのアドレス情報の構造体

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

ERROR および STATUS パラメータ

ER-ROR	STA-TUS*(W#16#...)	説明
0	0000	新しいデータが受け入れられました。 受信したデータの現在の長さは、RCVD_LEN に表示されます。
0	7000	ブロックの受信準備ができていません。
0	7001	ブロックの受信準備が完了し、受信ジョブが有効になりました。

0	7002	中間呼び出し。受信ジョブを処理中です。 注記: この処理フェーズ中に、「TURCV」が受信領域にデータを書き込みます。そのため、エラーによって受信領域内のデータの不一致が起こる恐れがあります。
1	8085	LEN パラメータが最大許容値よりも大きいか、最初の呼び出し以降に LEN または DATA パラメータの値を変更しました。
1	8086	ID パラメータが許容値の範囲内にありません
1	8088	• 受信領域が小さすぎます • LEN の値が、DATA によって指定された受信領域よりも大きくなっています。
1	8089	ADDR パラメータが、TADDR_Param 構造のデータブロックをポイントしていません。
1	80A1	通信エラー: • ユーザープログラムとオペレーティングシステムの通信レイヤ間の指定された接続がまだ確立されていません。 • ユーザープログラムとオペレーティングシステムの通信レイヤ間の指定された接続を現在終了中です。 この接続による受信ジョブは行えません。 • 新しいパラメータ設定がインターフェースに割り当てられています。
1	80B1	現在のジョブを終了する前に DATA パラメータが変更されました。
1	80B3	プロトコルの種類(接続記述子内の connection_type パラメータ)が UDP に設定されていません。 「TRCV」を使用してください。
1	80B7	ADDR パラメータの長さが 8 バイトではありません。
1	80C3	• この ID を持つブロックは、別の優先度クラスで既に処理中です。 • リソースが内部的に不足しています。
1	80C4	一時的な通信エラー: 新しいパラメータ設定がインターフェースに割り当てられています。
1	80C5	リモートパートナーによって接続が終了しました。
1	80C7	実行がタイムアウトしました。
1	80C9	RFC1006 / UDP の場合: 指定されている長さよりも長い受信データです(受信バッファのサイズを超えています)。
-	一般エラー情報	関連項目: GET_ERR_ID: ローカルでエラー ID を取得
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

T_RESET: 接続のリセット

説明

「T_RESET」命令は、既存の接続を終了した後で、再度確立します。

接続のローカルエンドポイントは保持されます。 次の場合、ローカルエンドポイントは自動的に生成されます。

- 接続が設定されており、CPU に読み込み済みである場合。
- 接続が、ユーザープログラム(たとえば「TCON」命令の呼び出し)によって生成済みである場合。

命令「T_RESET」は、すべての接続タイプ(TCP、UDP、ISO-on-TCP など)に対して実行することができます。 この接続に CPU のローカルインターフェースまたは CM/CP のインターフェースが使用されたかどうかは重要ではありません。

REQ パラメータを使用して「T_RESET」命令が呼び出されると、ID パラメータで指定された接続が終了し、必要に応じてデータ送信および受信バッファがクリアされます。 接続をキャンセルすると、進行中のすべてのデータ転送もキャンセルされます。 このため、データ転送が進行中の場合はデータが喪失するリスクがあります。 その後、アクティブな接続パートナーとして定義された CPU は、中断された通信接続の復元を自動的に試行します。 このため、「TCON」命令を呼び出して通信接続を再確立する必要はありません。

出力パラメータ DONE、BUSY、および STATUS は、ジョブのステータスを示します。

パラメータ

次の表に、「T_RESET」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	制御パラメータ REQUEST は、ID によって指定された接続を終了するためのジョブを開始します。 このジョブは信号立ち上がりエッジで開始します。
ID	Input	CONN_OUC	I、Q、M、D、L、または定数	中断されるパッシブパートナーへの接続の参照 ID は、ローカル接続記述子内の対応するパラメータ ID と同一であることが必要です。 値の範囲 W#16#0001 ~ W#16#0FFF
DONE	Output	BOOL	I、Q、M、D、L	STATUS パラメータ DONE • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
BUSY	Output	BOOL	I、Q、M、D、L	STATUS パラメータ BUSY • 0: ジョブが完了しました。

				• 1: ジョブがまだ完了していません。
ERROR	Output	BOOL	I、Q、M、D、L	STATUS パラメータ ERROR • 0: エラーは発生しませんでした。 • 1: 処理中にエラーが発生しました。 STATUS パラメータにエラーの種類に関する詳細情報を表示します。
STATUS	Output	WORD	I、Q、M、D、L	STATUS パラメータ STATUS エラー情報(「STATUS パラメータ」の表を参照)。

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

STATUS* (W#16#...)	説明
0000	エラーなし。
0001	接続が確立されていません
7001	接続終了処理が開始しました。
7002	接続を終了しています。
8081	ID パラメータで不明な接続が指定されています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

T_DIAG: 接続のチェック

説明

「T_DIAG」命令を使用し、接続のステータスをチェックし、この接続のローカルエンドポイントに関する詳細情報を読み出します。

- 接続は ID パラメータによって参照されます。 接続エディタで設定した接続エンドポイント、および、たとえば「TCON」命令によってプログラミングされた接続エンドポイントの両方を読み出すことが可能です。

  一時接続エンドポイント(たとえばエンジニアリングステーションに接続する際に作成されたエンドポイント)は、この処理では接続 ID が生成されないため、診断することはできません。
- 読み出された接続情報は、RESULT パラメータが参照する構造体に格納されます。
- 出力パラメータ STATUS は、接続情報を読み出せたかどうかを示します。 RESULT パラメータの構造体内の接続情報は、「T_DIAG」命令が STATUS = W#16#0000、および ERROR = FALSE で完了した場合のみ有効です。

  エラーが発生する場合、接続情報は評価できません。

接続情報に含まれる情報

RESULT パラメータでの接続情報の読み出しには、2 つの異なる構造体が使用できます。

- 「TDiag_Status」構造体には、使用されるプロトコル、接続ステータス、送信または受信されるデータのバイト数といった、接続エンドポイントに関する最も重要な情報のみが含まれます。
- 構造体「TDiag_StatusExt」には最も重要な情報だけではなく、接続試行の回数や接続中止の理由なども含まれます。

この 2 つの構造体の構造およびパラメータを以下に記載します(「TDIAG_Status および TDIAG_StatusExt 構造体」の表を参照)。

パラメータ

以下の表に、「T_DIAG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジが存在する場合、命令を開始して ID パラメータで指定された接続をチェックします。
ID	Input	CONN_OUC (WORD)	I、Q、M、D、L、または定数	割り当てられた接続へのリファレンス 値の範囲: W#16#0001 終了 W#16#0FFF
RESULT	InOut	VARIANT	D	接続情報が格納されている構造体へのポインタ。 構造体 TDiag_Status または TDiag_StatusExt は、RESULT パラメータで使用可能です(詳細については、「TDIAG_Status および TDIAG_StatusExt 構造体」の表を参照)。

DONE	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: 命令がまだ開始されていないか、または実行中です。 • 1: 命令がエラーなしで実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: 命令がまだ開始されていないか、既に完了しています。 • 1: 命令がまだ完了していません。新しいジョブを開始することはできません。
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ BUSY、DONE および ERROR

BUSY、DONE、ERROR、および STATUS パラメータを使用して、「T_DIAG」命令の実行ステータスをチェックできます。 BUSY パラメータは、処理ステータスを示します。 DONE パラメータを使用し、命令が正常に実行されたかどうかをチェックします。 「T_DIAG」の実行中にエラーが発生すると、ERROR パラメータがセットされます。

次の表に、BUSY、DONE、ERROR パラメータの関係を示します。

BUSY	DONE	ERROR	説明
1	-	-	命令が処理中です。
0	1	0	命令は正常に実行されました。 RESULT が参照する構造体内のデータは、この場合のみ有効です。
0	0	1	命令完了(エラーあり)。 STATUS パラメータにエラーの原因が出力されます。
0	0	0	新しい命令が割り当てられていません。

STATUS パラメータ

次の表に、STATUS パラメータの値の意味を示します。

STATUS* (W#16#...)	説明
0000	「T_DIAG」命令は正常に実行されました。 RESULT パラメータで参照される構造体内のデータを評価できます。
7000	命令の処理が行われていません。
7001	命令の処理が開始されました。
7002	接続情報が読み出されています(REQ パラメータは対象外)。

8086	ID パラメータの値が有効な範囲(W#16#0001 ~ W#16#0FFF)外にあります。
8089	RESULT パラメータが不正なデータタイプをポイントしています(構造体 TDIAG_Status および TDIAG_StatusExt のみ)。
80A3	ID パラメータが、存在していない接続エンドポイントを参照しています。 プログラミングされた接続の場合、このエラーは「TDISCON」命令の呼び出し後も発生する可能性があります。
80C4	内部エラー 一時的に接続エンドポイントにアクセスできません。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

構造体 TDIAG_Status および TDIAG_StatusExt

以下の表で、TDIAG_Status および TDIAG_StatusExt 構造体の形式について説明します。

- InterfaceID から ReceivedBytes パラメータまでは、「TDIAG_StatusExt」と「TDIAG_Status」命令で同じです。
- 構造体 TDIAG_StatusExt には、パラメータ ConnTrials ~ LastDisconnTimeStamp も含まれています。

各エレメントの値は、命令がエラーなく実行された場合にのみ有効です。 エラーが発生する場合、パラメータの内容は変化しません。

名前	データタイプ	説明
以下のパラメータは、TDIAG_Status および TDIAG_StatusExt 構造体の両方で使用されます。		
InterfaceID	HW_ANY	CPU または CM/CP のインターフェース ID (LADDR)
ID	CONN_OUC	診断される接続 ID。 呼び出しに成功すると、このエレメントの値は「T_DIAG」命令のパラメータ ID と同じになります。
Connection-Type	BYTE	接続に使用されるプロトコルタイプ: • 0x01: 未使用。 • ... • 0x0B: TCP プロトコル(IP_v4) • 0x0C: ISO-on-TCP プロトコル(RFC1006) • 0x0D: TCP プロトコル(DNS) • 0x0E: ダイヤルインプロトコル • 0x0F: WDC プロトコル • 0x10: SMTP プロトコル • 0x11: TCP プロトコル • 0x12: TCP および ISO-on-TCP プロトコル(RFC1006) • 0x13: UDP プロトコル • 0x14: 予約済み • 0x15: PROFIBUS バスアクセスプロトコル(FDL) • 0x16: ISO 8073 転送プロトコル(ISO ネイティブ) • ... • 0x20: IPv4 に基づく SMTP または SMTPS プロトコル

		• 0x21: IPv6 に基づく SMTP または SMTPS プロトコル • 0x22: FQDN (**F**ully **Q**ualified **D**omain **N**ame)に基づく SMTP または SMTPS プロトコル • ... • 0x70: S7 コネクション • その他: 予約済み
ActiveEstab-lished	BOOL	• FALSE: ローカル、パッシブ接続エンドポイント • TRUE: ローカル、アクティブ接続エンドポイント
State	BYTE	接続エンドポイントの現在のステータス • 0x00: 未使用。 • 0x01: 接続が終了しました。「T_RESET」命令呼び出し後などの、一時的なステータス。 この後、システムは接続の再確立を自動的に試行します。 • 0x02: アクティブ接続エンドポイントが、リモート通信パートナーへの接続確立を試行しています。 • 0x03: パッシブ接続エンドポイントが、リモート通信パートナーへの接続確立を待機しています。 • 0x04: 接続が確立されました。 • 0x05: 接続を終了中です。 これは、"T_RESET"または"T_DISCON」命令が呼び出されたためである可能性があります。 Other possible reasons are protocol errors and line breaks. • 0x06 ~ 0xFF: 未使用。
Kind	BYTE	接続エンドポイントのモード: • 0x00: 未使用。 • 0x01: 設定済み、スタティック接続が設定され CPU に読み込まれています。 • 0x02: 設定済み、ダイナミック接続が設定され CPU に読み込まれています(現在はサポートされていません)。 • 0x03: 「TCON"命令を使用してユーザープログラムによって生成された、プログラミングされた接続です。「TDISCON」命令の呼び出し、または CPU の STOP ステータスへの移行によって、接続エンドポイントが破棄されました。 • 0x04: エンジニアリングステーション(ES)またはオペレータステーション(OS)などによってダイナミック接続が一時的に確立されています。 (この接続タイプは、ID が存在しないため現在診断できません)。 • 0x05 ~ 0xFF: 未使用。
SentBytes	UDINT	送信済みデータバイト数。
ReceivedBytes	UDINT	受信済みデータバイト数。
以下のパラメータは、TDiag_StatusExt 構造体でのみ使用します。		
ConnTrials	UDINT	接続試行の回数。 接続が確立されると、ConnTrials の値は「0」です。このエレメントが「0」ではない場合、接続に問題があります。 注記: パッシブ接続エンドポイントの場合、この値は「1」よりも大きくなることはありません。
ConnTrials-Success	UDINT	成功した接続試行の回数。 このエレメントは接続エンドポイントのライフサイクル中にリセットされることはなく、0xFFFF FFFF に到達すると 0 に戻ります。

		注記: この接続に問題が発生していない場合、このパラメータは「1」です。
LastConnErr-Reason	UDINT	エラーが発生した最後の接続試行中の出力されたエラー ID(エラーメッセージは LastDisconnReason パラメータのものと同じです): • 0x4F01: リモート接続エンドポイントに到達できません(通常、このエラーは接続の確立中に発生します。)。 • 0x4F02: 接続がローカルで終了しました。 • 0x4F03: 接続がリモート通信パートナーによって終了されました。 • 0x4F04: 接続がプロトコルエラーよって終了されました。 • 0x4F05: 接続がローカルで検出されたネットワークの問題よって終了されました。 • 0x4F06: 接続がリモートで検出されたネットワークの問題よって終了されました。 • 0x4F07: 接続がプロトコルのタイムアウトによって終了されました。 • 0x4F08: 不正なパラメータ割り当て: ローカルパートナー自身のアドレスに接続を確立しようとしています。 • 0x4F09: 接続が「T_RESET」命令の呼び出しによって一時的にリセットされました。 • 0x4F0A: 使用可能な接続リソースが不足しています(数が多すぎます)。 • 0x4F0B: 内部エラー: 不正なアドレスパラメータ • 0x4F0C: 内部 CPU 通信エラー • 0x4F0D: CPU と CM/CP 間の内部 AS 通信エラー • 0x4F0E: 指定されたローカル TCP/UDP ポート(または RFC1006-T セレクタ)が、既に使用されています。
LastConnErr-TimeStamp	LDT	エラーが発生した最後の接続試行の時間。
LastDisconn-Reason	UDINT	最後の接続が終了する原因となったエラー ID(エラーメッセージは LastConnErrReason パラメータのものと同じです): • 0x4F01: リモート接続エンドポイントに到達できません(通常、このエラーは接続の確立中に発生します。)。 • 0x4F02: 接続がローカルで終了しました。 • 0x4F03: 接続がリモート通信パートナーによって終了されました。 • 0x4F04: 接続がプロトコルエラーよって終了されました。 • 0x4F05: 接続がローカルで検出されたネットワークの問題よって終了されました。 • 0x4F06: 接続がリモートで検出されたネットワークの問題よって終了されました。 • 0x4F07: 接続がプロトコルのタイムアウトによって終了されました。 • 0x4F08: 不正なパラメータ割り当て: ローカルパートナー自身のアドレスに接続を確立しようとしています。 • 0x4F09: 接続が「T_RESET」命令の呼び出しによって一時的にリセットされました。 • 0x4F0A: 使用可能な接続リソースが不足しています(数が多すぎます)。 • 0x4F0B: 内部エラー: 不正なアドレスパラメータ • 0x4F0C: 内部 CPU 通信エラー

		• 0x4F0D: CPU と CM/CP 間の内部 AS 通信エラー • 0x4F0E: 指定されたローカル TCP/UDP ポート(または RFC1006-T セレクタ)が、既に使用されています。
LastDisconnTi-meStamp	LDT	最後の接続終了の時間。

T_CONFIG: インターフェースの設定

この章には下記に関する情報が記載されています：

T_CONFIG の説明

説明

「T_CONFIG」命令は、CPU の統合 PROFINET インターフェース、または、CP/CM のインターフェースのプログラム制御設定に使用されます。

ユーザープログラムからこの命令を使用して、PROFINET デバイス名のイーサネットアドレスを変更できます。 既存の構成データが上書きされます。

この命令「T_CONFIG」を使用して、以下の変更を行うことができます。

- IP プロトコルの設定
  - IP アドレス
  - サブネットマスク
  - ルータアドレス
- PROFINET の設定
  - PROFINET デバイス名の割り当て

これらの設定は、ダイアログ[イーサネットアドレス]の[IP プロトコル]および[PROFINET]の設定オプションと対応しています。 これは、ビュー[デバイスとネットワーク]の PROFINET インターフェースのプロパティに表示されます。

警告

「T_CONFIG」命令実行後の CPU の再起動

CPU は、この命令を実行して IP パラメータを変更した後に、再起動されます。 CPU が STOP モードに移行し、ウォームリスタートが実行されてから、CPU が再起動します(RUN モード)。

この命令の実行に続いて CPU が再起動された後に、制御プロセスが安全な動作モードであることを確認します。 制御されていない操作は、たとえば、誤動作またはプログラミングエラーのために、重大な機器の損傷または人身傷害を招く場合があります。 非保持データは失われる場合があります。

要件

この命令を使用するには、ハードウェアコンフィグレーションで、ユーザープログラムが IP アドレスパラメータおよびデバイス名の割り当てを行う必要があることを明示的に指定する必要があります。

- このために、デバイスビューで、PROFINET インターフェースのプロパティを開きます。 ダイアログ[イーサネットアドレス]で、以下のオプションを有効にします。
  - 「T_CONFIG」を使用して IP アドレスパラメータを変更するには: [IP プロトコル]の設定[IP アドレスをデバイスに直接設定する]を選択します。
  - 「T_CONFIG」を使用して PROFINET デバイスネームを変更するには: [PROFINET]の設定[PROFINET デバイスネームをデバイスに直接設定する]を選択します。
- 構成データを以下のシステムデータタイプに格納し、パラメータ CONF_DATA に割り当てる必要があります。
  - IP アドレス、サブネットマスク、およびルータアドレスは、システムデータタイプ IF_CONF_V4 に格納します。
  - デバイス名は、システムデータタイプ IF_CONF_NOS に格納します。 デバイス名の割り当てに関する制限を遵守してください(パラメータ CONF_DATA を参照)。

ファンクションの説明

「T_CONFIG」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。REQ = 1 で「T_CONFIG」を呼び出すことによって、設定プロセスを開始します。常に 1 つのジョブ以外はアクティブにできません。

このブロックはエッジでトリガされます。すなわち、BUSY = FALSE の後に、REQ = FALSE でブロックを再び呼び出して、インスタンスを有効にする必要があります。

パラメータ

以下の表に、「T_CONFIG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	命令の呼び出しが REQ = 1 の場合、命令の処理を開始します。
INTER-FACE	Input	HW_INTER-FACE	I、Q、M、D、L、または定数	インターフェースのハードウェア識別子 ハードウェア ID は、デバイスビューのインターフェースのプロパティ、および、PLC タグのシステム定数に表示されます。
CONF_DA-TA	Input	VARIANT	D、L	システムデータタイプ IF_CONF_HEAD-ER、IF_CONF_V4、および IF_CONF_NOS が収納されている親構造体へのポインタ (パラメータ CONF_DATA の説明を参照)。
DONE	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: 処理がまだ完了していません。 • 1: 命令の処理が正常に終了しました。
BUSY	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: 命令の処理がまだ開始されていないか、完了したか、またはキャンセルされました。 • 1: 命令の処理が進行中です
ERROR	Output	BOOL	I、Q、M、D、L	ステータスパラメータ: • 0: エラーは発生していません。 • 1: エラー
STATUS	Output	DWORD	I、Q、M、D、L	詳細なステータス情報 エラーコードの形式での詳細なエラーおよびステータス情報が、パラメータ STA-TUS に出力されます。
ERR_LOC	Output	DWORD	I、Q、M、D、L	エラーの場所: • 0: 命令の実行またはパラメータの割り当て時のエラー。 • > 0: CONF_DATA パラメータの設定データの構造体または内容のエラー。

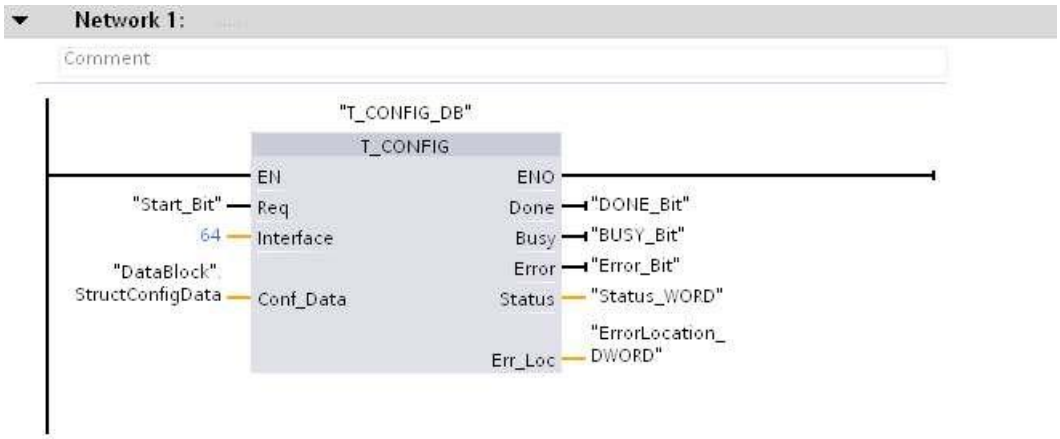
有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

例

次の例では、命令「T_CONFIG」を使用して、IO デバイスのデバイス名を変更します。

命令の呼び出し

- 命令の実行は、REQ = 1 で開始されます。
- PROFINET インターフェースのハードウェア ID をパラメータ Interface で指定します。
- 構造体「StructConfigData」は、パラメータ CONF_DATA で参照されます。 この構造体は、グローバルデータブロック「DataBlock」で作成されました。



パラメータ CONF_DATA

CONF_DATA パラメータの構造体「StructConfigData」には、以下の情報が収納されています。

- Header (IF_CONF_HEADER)内

  SubfieldCount = 1: 以降で、1 つの追加構造体(nos)のみが使用されることを示します。
- 構造体「nos」(システムデータタイプ IF_CONF_NOS)内
  - Lenght = 11: 構造体 NOS の全体の長さに関する情報(デバイス名「myplc」用の 5 バイト + パラメータ Id、Length、および Mode 用の 6 バイト)

    注記: 絶対長の代わりに、動的長さの既定開始値(Lenght = 0)を使用できます。
  - Mode = 1: デバイス名「myplc」の常時変更。
  - NOS[1] ... NOS[5]: デバイス名(1 文字/バイト)

DataBlock

	Name	Data type	Start value
1	Static		
2	StructConfigData	Struct	
3	Header	IF_CONF_Header	
4	FieldType	UInt	0
5	FieldId	UInt	0
6	SubfieldCount	UInt	1
7	nos	IF_CONF_NOS	
8	Id	UInt	40
9	Length	UInt	11
10	Mode	UInt	1
11	NOS	array [1..240] of Byte	
12	NOS[1]	Byte	'm'
13	NOS[2]	Byte	'y'
14	NOS[3]	Byte	'p'
15	NOS[4]	Byte	'l'
16	NOS[5]	Byte	'c'
17	NOS[6]	Byte	16#0
18	NOS[7]	Byte	16#0
19	NOS[8]	Byte	16#0
20	NOS[9]	Byte	16#0
21	NOS[10]	Byte	16#0
22	NOS[11]	Byte	16#0

パラメータ CONF_DATA

構成データの構造体

CONF_DATA パラメータの構成データは、グローバルデータブロック、または、ブロックインターフェースのセクション「Static」に格納できます。

構成データは、以下の構造体に従って、格納する必要があります。

名前		データタイプ	説明
ConfData		Struct	CONF_DATA パラメータに割り当てられる親構造体。
	Header	IF_CONF_HEADER	ヘッダーは、システムデータタイプの数を定義するために使用されます。 システムデータタイプ IF_CONF_HEADER は常に含まれます。
	IPData	IF_CONF_V4	IP アドレス、サブネットマスク、およびルータアドレスは、このシステムデータタイプに格納されます。 IF_CONF_V4 は、「T_CONFIG」を使用してイーサネットアドレスを変更する場合のみ、作成します。
	NoS	IF_CONF_NOS	PROFINET デバイスネームは、このシステムデータタイプに格納されます。 IF_CONF_NOS は、「T_CONFIG」を使用してデバイス名を変更する場合のみ、作成します。

システムデータタイプ IF_CONF_HEADER、IF_CONF_V4、および IF_CONF_NOS は、データブロックまたはブロックインターフェースの列[データタイプ]にシステムデータタイプの名前を入力して作成します。 システムデータタイプの名前は、自由に割り当てることができます。

システムデータタイプ IF_CONF_Header

システムデータタイプ IF_CONF_Header を使用して、「T_CONFIG」の実行中に使用されるシステムデータタイプ IF_CONF_V4 および IF_CONF_NOS の数を指定します。

バイト	パラメータ	データタイプ	開始値	説明
0 ... 1	FieldType	UINT	0	フィールドタイプ: 常に値が「0」。
2 ... 3	FieldId	UINT	0	エラー ID: 常に値が「0」。
4 ... 5	SubfieldCount	UINT	0	システムデータタイプ IF_CONF_V4 および IF_CONF_NOS の数: • 1: これらのシステムデータタイプの 1 つだけが使用されます。 • 2: 両方のシステムデータタイプが使用されます。

システムデータタイプ IF_CONF_V4

IP アドレス、サブネットマスク、およびルータアドレスは、システムデータタイプ IF_CONF_V4 で定義されます。

バイト	パラメータ	データタイプ	開始値	説明
0 ... 1	Id	UINT	30	システムデータタイプ ID。このパラメータの開始値は変更できません。
2 ... 3	Length	UINT	18	システムデータタイプ IF_CONF_V4 の長さ IF_CONF_V4 の複数のパラメータは固定長および構造体を持つため、長さ指定で開始値を使用する必要があります。
4 ... 5	Mode	UINT	0	アドレス指定の有効性 • 1: 構成データが固定的に有効 • 2: 既存の固定構成データの削除を含めて、構成データが一時的に有効
6 ... 9	InterfaceAddress	IP_V4 *	0.0.0.0	IP アドレス
10 ... 12	SubnetMask	IP_V4 *	0.0.0.0	サブネットマスク
14 ... 16	DefaultRouter	IP_V4 *	0.0.0.0	ルーターアドレス
*データタイプ IP_V4 は 4BYTE の構造体です。この構造体には、それぞれのパラメータのそれぞれのアドレス(たとえばパラメータ SubnetMask では、IP プロトコルのサブネットマスクの 4 桁のアドレス)が含まれています。				

サブフィールド IF_CONF_NOS

サブフィールド IF_CONF_NOS を使用して、「T_CONFIG」命令の実行時に割り当てるステーション名を指定します。

バイト	パラメータ	データタイプ	開始値	説明
0 ... 1	Id	UINT	40	システムデータタイプ ID。このパラメータの開始値は変更できません。
2 ... 3	Length	UINT	246	システムデータタイプ IF_CONF_NOS の長さ(バイト単位)。 • 絶対長の場合、Lenght パラメータの値は以下から構成されます: ○ Id、Length、および Mode パラメータの 6 バイト。 ○ デバイス名(パラメータ NOS)用の最大 240 バイト。 例: 長さ全体が 10 は、デバイス名「plc1」が長さ 4 文字(= 4 バイト)の場合の結果です。 • 動的な長さの場合、Lenght パラメータで既定開始値 246 を使用します。 名前の後に必ず値「0」を入力してください(NOS パラメータの説明を参照)。
4 ... 5	Mode	UINT	0	デバイス名の変更の有効性:

				• 1: デバイス名の永続的な有効性。 • 2: デバイス名の一時的な有効性。
6 ... 244	NOS	ARRAY [1...240] of Byte	0	デバイス名(Name of Station) • ARRAY を最初のバイトから占有する必要があります。 最初のバイトとして「0」が割り当てられた場合、ステーション名は削除されます。 • この名前の最小長さは 2 バイトです。名前の最大長は、240 文字です。 • デバイス名がパラメータ Length で指定された長さよりも短い場合は、現在のステーション名の後にゼロバイト(16#0 16 進数)を入力する必要があります(IEC 61185-6-10 に準拠)。 ゼロバイトを入力しないと、NOS は拒否され、命令「T_CONFIG」がパラメータ STATUS にエラーコード DW#16#C0809400 を出力します。 • デバイス名がパラメータ Length で指定された長さよりも長い場合は、指定された長さのみデバイス名が書き込まれます。 デバイス名では、以下の制限が適用されます。 • 名前は ASCII コードで指定する必要があります。 • 名前では、小文字、数字、ハイフン、または点のみを使用できます。 ○ 名前は、ハイフンで開始/終了してはいけません。 ○ 名前は数字で開始してはいけません。 ○ 名前の形式が、n.n.n.n (n = 0, ... 999)であってはいけません。 ○ 名前は、文字列「port-xyz」または「port-xyz-abcde」(a、b、c、d、e、x、y、z = 0、... 9)で開始してはいけません。 • 2 つの点の間の名前要素は、最大 63 文字の長さまで可能です。 • ウムラウト、括弧、アンダースコア、スラッシュ、空白などの特殊文字は使用できません。 無効な文字が使用されると、エラーコード C080_9400 がパラメータ STATUS に出力されます。

DONE、BUSY、および ERROR パラメータ

説明

次の表に、BUSY、DONE、ERROR の関係を示します。この表を使用して、命令の現在のステータスと、構成データの転送がいつ終了したかを判別できます。

BUSY	DONE	ERROR	説明
TRUE	FALSE	FALSE	ジョブが処理中です。
FALSE	TRUE	FALSE	ジョブが正常に完了しました。
FALSE	FALSE	TRUE	ジョブがエラーありで終了しました。 エラーの原因は、STATUS パラメータで確認できます。
FALSE	FALSE	FALSE	命令に(新しい)ジョブが割り当てられませんでした。

STATUS および ERR_LOC パラメータ

説明

命令「T_CONFIG」のステータスおよびエラーメッセージは、パラメータ STATUS および ERR_LOC に出力されます。

- パラメータにエラーの原因が出力されます。STATUS
- 発生したエラーの場所は、パラメータ ERR_LOC に出力されます。 この場合、以下のオプションが使用できます。
  - 16#0000_0000: 命令呼び出し時のエラー(たとえば、命令へのパラメータ割り当て時のエラー、または、PROFINET インターフェースとの通信でのエラー)。
  - 16#0001_0000: システムデータタイプ IF_CONF_HEADER のパラメータの構成データエラー。
  - 16#0001_0001: システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のパラメータの構成データエラー。

次の表に、パラメータ STATUS および ERR_LOC の可能な値を示します。

STATUS*	ERR_LOC*	説明
0000_0000	0000_0000	命令処理がエラーなしで完了しました。
0070_0000	0000_0000	ジョブ処理が行われていません。
0070_0100	0000_0000	命令処理の開始。
0070_0200	0000_0000	中間呼び出し(REQ は対象外)。
C08x_yy00	0000_0000	一般エラー情報関連項目 GET_ERR_ID: ローカルでエラー ID を取得
C080_8000	0000_0000	命令呼び出し時のエラー: Interface パラメータのハードウェア ID が無効です。
C080_8100	0000_0000	命令呼び出し時のエラー: Interface パラメータのハードウェア ID が、PROFINET インターフェースをアドレス指定していません。
C080_8700	0000_0000	命令呼び出し時のエラー: パラメータでのデータブロックの長さが正しくありません。CONF_DATA
C080_8800	0001_0000	システムデータタイプ IF_CONF_HEADER のエラー: FieldType パラメータの値が無効です。 FieldType で値「0」を使用します。
C080_8900	0001_0000	システムデータタイプ IF_CONF_HEADER のエラー: パラメータ FieldId の値が無効であるか、複数回使用されています。 FieldId で値「0」を使用します。
C080_8A00	0001_0000	システムデータタイプ IF_CONF_HEADER のエラー:

		SubfieldCount パラメータの番号が正しくありません。 システムデータタイプ IF_CONF_V4 および IF_CONF_NOS の使用中の正しい番号を入力します。
C080_8B00	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: パラメータ Id の値が無効です。 IF_CONF_NOS の場合「30」を使用し、IF_CONF_V4 の場合「40」を使用します。
C080_8C00	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: 不正なシステムデータタイプが使用されているか、順序が間違っているか、またはシステムデータタイプが複数使用されています。
C080_8D00	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: パラメータ Length の値が不正または無効です。
C080_8E00	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: パラメータ Mode の値が不正または無効です。 値「1」(永続的)または値「2」(一時的)のみが有効です。
C080_9000	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: 構成データを適用できません。考えられる原因: 設定「デバイスの IP アドレスのセットまたは「デバイスの PROFINET デバイスネームのセット」がハードウェアコンフィグレーションで選択されていません。
C080_9400	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: パラメータの値が未定義または無効です。
C080_9500	0001_0001	システムデータタイプ IF_CONF_V4 または IF_CONF_NOS のエラー: 2 つのパラメータの値が矛盾しています。
C080_C200	0000_0000	命令呼び出し時のエラー: 構成データを転送できません。考えられる原因: PROFINET インターフェースにアクセスできません。
C080_C300	0000_0000	命令呼び出し時のエラー: リソースが不足しています(たとえば、異なるパラメータで「T_CONFIG」を複数回呼び出すなど)
C080_C400	0000_0000	命令呼び出し時のエラー: 一時的な通信エラー。 サマータイムへの切り替えの時刻指示。
C080_D200	0000_0000	命令呼び出し時のエラー: 呼び出しができません。 命令が、選択された PROFINET インターフェースによってサポートされていません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

開放型ユーザー間通信ライブラリ V3.1 と V4.0 の相違点

この章には下記に関する情報が記載されています：

- 新しいライブラリバージョン OUC V4.0 (S7-1200, S7-1500)
- 命令の動作変更 (S7-1200, S7-1500)

新しいライブラリバージョン OUC V4.0

概要

ファームウェアバージョン 4.1 以降の CPU S7-1200 は、開放型ユーザー間通信(OUC)用の新しい命令バージョンをサポートしています。 新しい命令には、バージョン 4.0 のライブラリが含まれています。

このライブラリを使用する場合は、開放型ユーザー間通信用の新しい命令の動作が従来のものと異なるため、アプリケーションプログラムに変更が必要となります。

このセクションでは詳細な相違点、特に命令の呼び出し動作について説明します。

注記

この説明は、CPU S7-1200 V4.0 以前から S7-1200 V4.1 以降にアップグレードする場合のみ適用されます。

S7-1500 CPU を使用している場合は、ライブラリバージョン間の相違はありません。 また、S7-1200 バージョン 4.1 以降を使用していて、開放型ユーザー間通信用のライブラリをバージョン 4.0 変換しない場合も同様です。

開放型ユーザー間通信ライブラリのバージョン 3.1 と 4.0 の相違点

以下の表に、バージョン 3.1 と 4.0 で異なる開放型ユーザー間通信ライブラリの命令を示します。 詳細情報については、命令の名前をクリックしてください。

命令	ライブラリのバージョン V3.1 (V4.0 以前の CPU FW)	ライブラリのバージョン V4.0 (V4.1 以降の CPU FW)
TSEND_C	V2.1	V3.0
TRCV_C	V2.1	V3.0
TMAIL_C *	V2.1	V3.0
TCON	V3.0	V4.0
TDISCON	V2.1	V2.1 (ライブラリ V3.1 と同一)
TSEND	V3.0	V4.0
TRCV	V3.0	V4.0
TUSEND	V3.0	V4.0
TURCV	V3.0	V4.0
T_RESET *	V1.1	V1.2
T_DIAG *	V1.1	V1.2
T_CONFIG	V1.0	V1.0 (ライブラリ V3.1 と同一)
MB_CLIENT	V3.1	V4.0
MB_SERVER	V3.1	V4.0
* ユーザープログラムに影響を及ぼすバージョンの相違はありません。		

命令の動作変更

この章には下記に関する情報が記載されています：

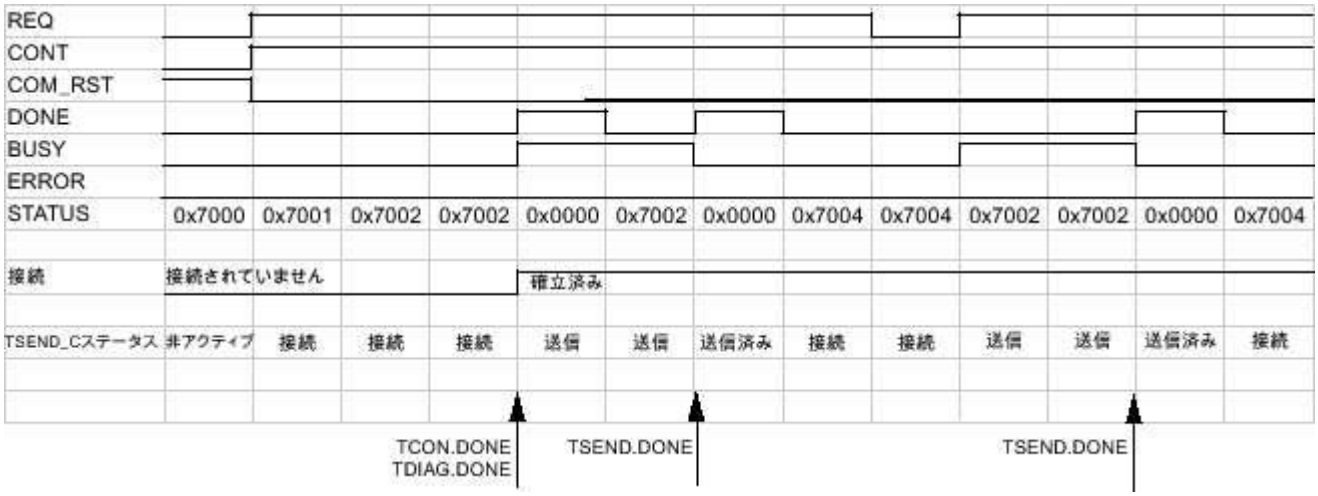
- TSEND_C 命令での変更 (S7-1200, S7-1500)
- TRCV_C 命令での変更 (S7-1200, S7-1500)
- TCON 命令での変更 (S7-1200, S7-1500)
- TSEND/TUSEND 命令での変更 (S7-1200, S7-1500)
- TRCV/TURCV 命令での変更 (S7-1200, S7-1500)
- MB_SERVER/MB_CLIENT 命令での変更 (S7-1200, S7-1500)

TSEND_C 命令での変更

TSEND_C の呼び出し動作(V3.0 より前)

TSEND_C 命令のバージョン 2.1 までは、DONE 出力パラメータは 2 回設定されます。 内部的に使用される TCON 命令によって接続が確立されるときに 1 回、次に内部的に使用される TSEND 命令によって送信操作が行われた後に 1 回設定されます。

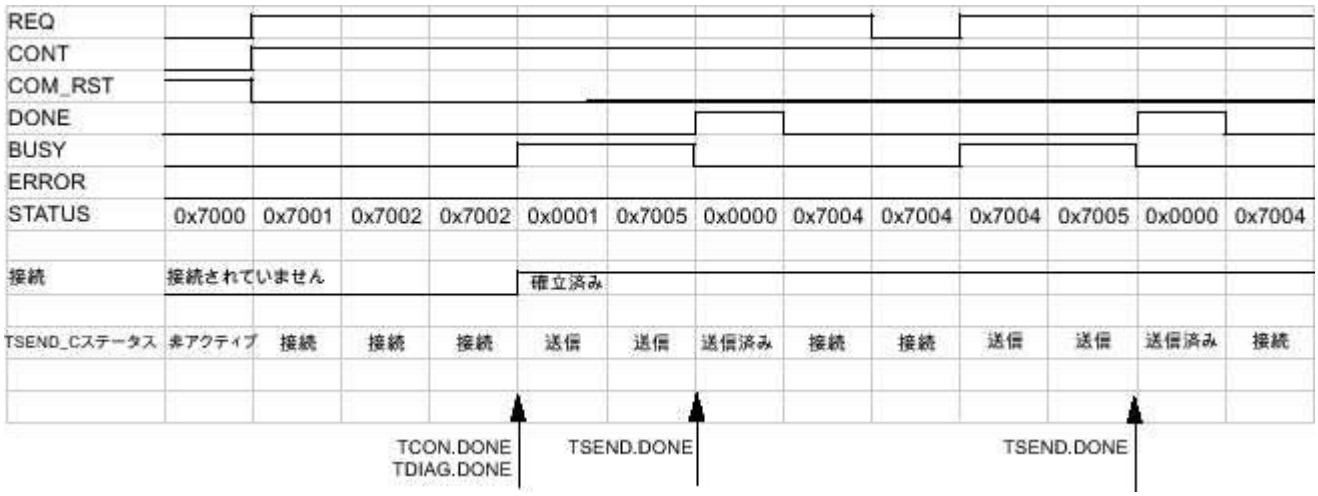
次の図に、TSEND_C V2.1 による接続確立およびデータの送信を示します。



COM_RST パラメータを「1」にセットすると、現在確立されている接続または現在のデータ伝送をいつでもリセットできます。これによって、既存の通信接続が終了し、新しい通信が確立されます。

TSEND_C の呼び出し動作(V3.0 以降)

TSEND_C バージョン以降、DONE パラメータは、内部的に使用される TSEND 命令(STATUS = 0000)によってデータ転送が完了した場合にのみ設定されます。



COM_RST パラメータが「1」にセットされると、既存の接続が一時的に切断されリセットされます。 ただし、TSEND_C V2.1 とは異なり、接続エンドポイントは保持されます。

注記

バージョン 3.0 以降の TSEND_C での追加プロトコル

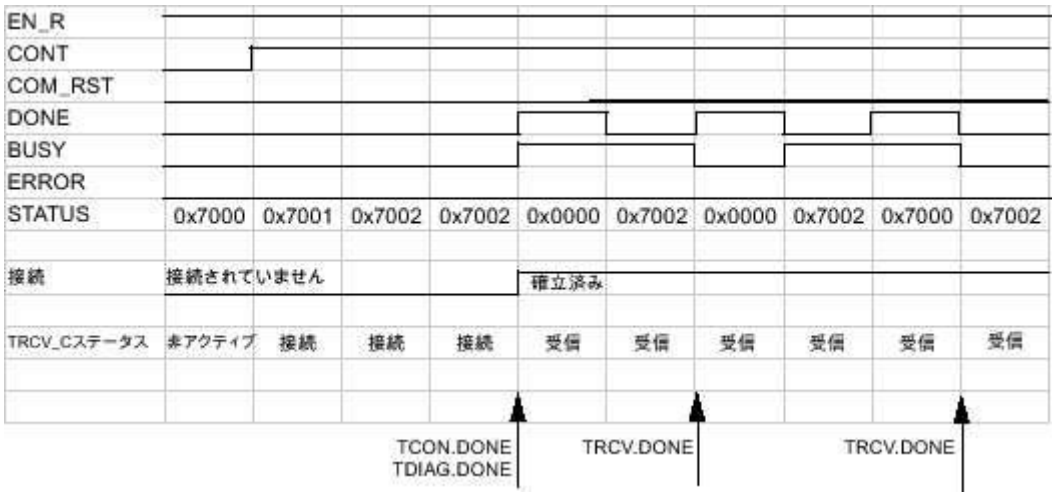
TSEND_C 命令のバージョン 3.0 でも、CPU のインターフェース経由および CM/CP 経由の UDP および UDP ブロードキャストをサポートしています。

TRCV_C 命令での変更

TRCV_C の呼び出し動作(V3.0 より前)

TRCV_C 命令のバージョン 2.1 までは、DONE 出力パラメータは接続確立後に設定されます。 STATUS 出力パラメータは、接続またはデータ転送が完了したかどうかは区別しません。

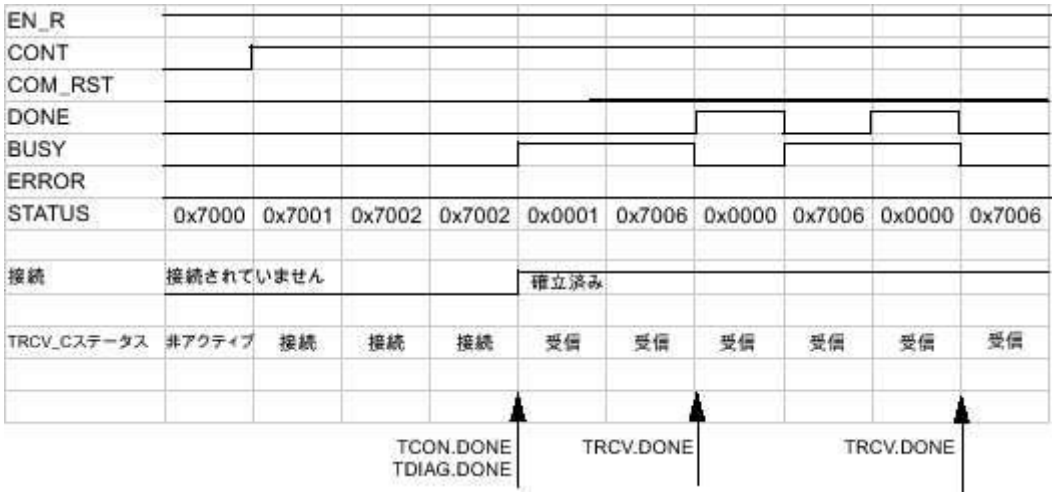
次の図に、TRCV_C V2.1 による接続確立およびデータの送信を示します。



COM_RST パラメータを「1」にセットすると、現在確立されている接続または現在のデータ伝送をいつでもリセットできます。これによって、既存の通信接続が終了し、新しい通信が確立されます。

TRCV_C の呼び出し動作(V3.0 以降)

TRCV_C バージョン以降、DONE パラメータは、内部的に使用される TRCV 命令(STATUS = 0000)によってデータ転送が完了した場合にのみ設定されます。 内部的に使用される TCON 命令による接続確立の完了は、STATUS パラメータでの出力値 0x0001 によって表示されます。



COM_RST パラメータが「1」にセットされると、既存の接続が一時的に切断されリセットされます。 ただし、TSEND_C V2.1 とは異なり、接続エンドポイントは保持されます。

注記

バージョン 3.0 以降の TRCV_C での追加プロトコル

TRCV_C 命令のバージョン 3.0 でも、CPU のインターフェース経由および CM/CP 経由の UDP および UDP ブロードキャストをサポートしています。

注記

TCP プロトコルのバリエーションの場合は ADHOC モードを使用

TRCV 命令に従って、バージョン 2.1 までの TRCV_C を使用して LEN パラメータに値「0」を割り当てると、ADHOC モードが有効になります。 この命令のバージョン 3.0 以降では、このために ADHOC パラメータを使用します。 詳細は、命令の記述を参照してください。

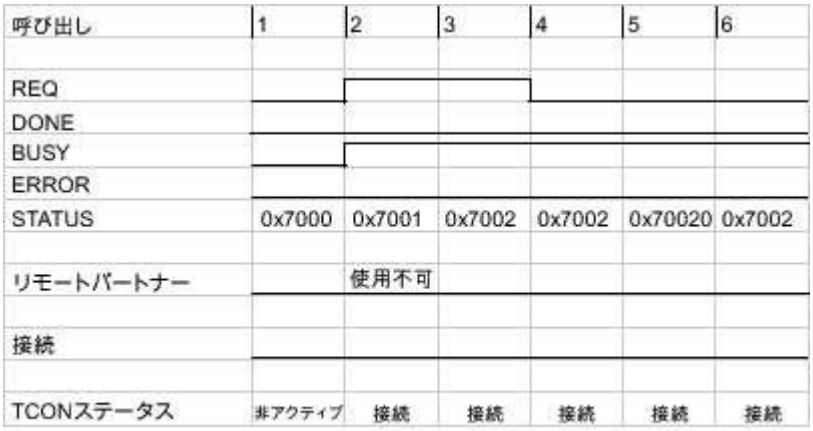
TCON 命令での変更

接続エラーがある場合の呼び出し動作の変更

TCON の以前の呼び出し動作(V4.0 より前)

- 以前の TCON 命令(V4.0 より前)は、 REQ 入力パラメータの立ち上がりエッジで有効になります。
- リモート通信パートナーが接続できない場合、命令は BUSY 出力パラメータを設定します。
- エラーメッセージは出力されません。

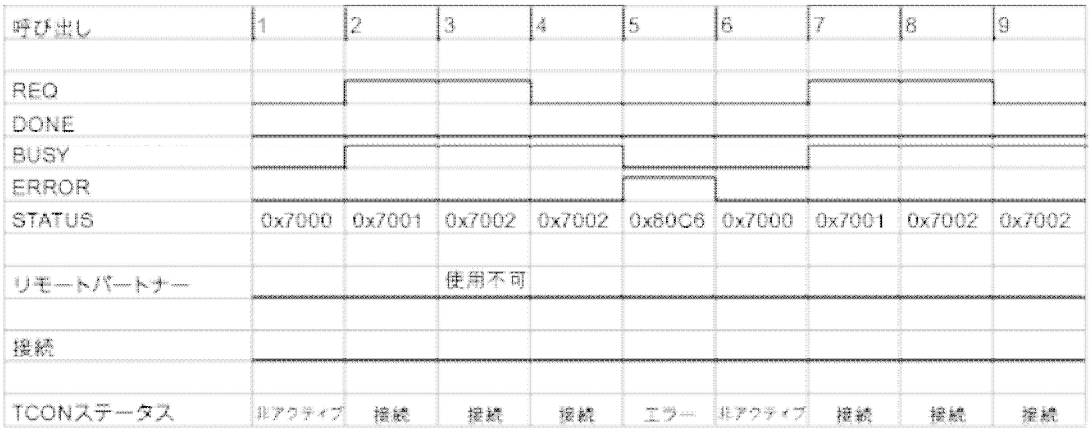
次の図に、アクティブ接続パートナー部分での TCON の動作を示します。 接続が確立しない場合、命令は再び呼び出されません。



TCON の新しい呼び出し動作(V4.0 以降)

- 新しい TCON 命令も、REQ 入力パラメータの立ち上がりエッジで有効になります。
- 以前とは異なり、リモート通信パートナーが接続できない場合はエラーメッセージが出力されます。 このエラーは照会でき、呼び出しは REQ パラメータの新しいエッジにより再び開始できます。
- ユーザープログラムでは、以下の点が変更されました。
  - 評価することが可能な TCON の追加エラーメッセージ(TCON の説明を参照)。
  - 接続確立は、新しい立ち上がりエッジにより再初期化することができます。 これを行うには、DONE および ERROR パラメータを照会してエラーが存在することを確認します。

次の図に、アクティブ接続パートナー部分での TCON の動作を示します。 命令は、ネットワークエラー(エラーコード 80C6)の後に再び呼び出されます。



通信エラーなしの呼び出し動作

接続エラーがない場合、TCON バージョンは同じ動作を行います。

次の図に、アクティブ接続パートナー部分での TCON の動作を示します。

TSEND/TUSEND 命令での変更

接続確立での送信操作

送信中に遅延や割り込みが発生しない場合、TSEND/TUSEND 命令のバージョン 3.0 と 4.0 は同じ動作を行います。

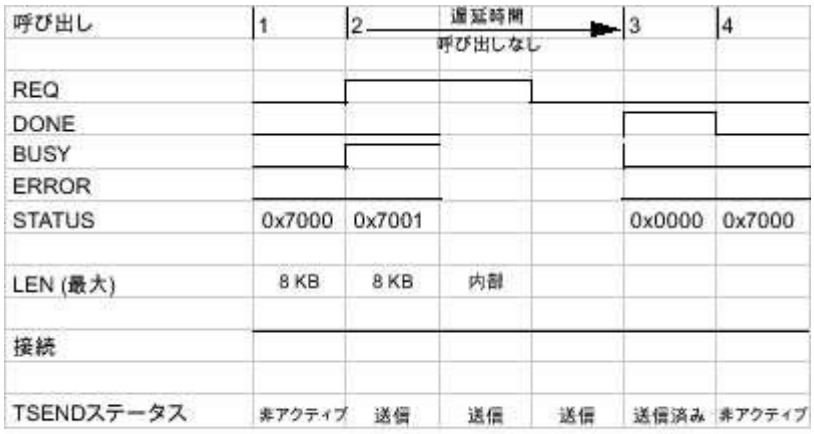
- 命令は、「REQ」パラメータの立ち上がりエッジで開始します。 命令は非同期で実行されます。つまり、完全な実行が DONE パラメータによって示されるまで、呼び出しを行う必要があります。
- TSEND を使用して、最大で 8 KB のデータを送信できます。 TUSEND を使用して、最大で 1472 バイトのデータを送信できます。 このデータ量は命令が完全に実行された場合の数値で、DONE パラメータが設定されるまで必要なすべての呼び出しが行われます。
- データ伝送は 3 段階で行われます。
  - データはオペランド領域から内部バッファ(STATUS=7001 によって表示)に書き込まれます。
  - 次にリモート通信パートナーへの伝送を行います。
  - 伝送が正常に完了した場合、DONE に「1」が設定され、STATUS が「0」にリセットされます(下図の呼び出し 5 を参照)。

呼び出し	1	2	3	4	5	6
REQ						
DONE						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
LEN (最大)	8 KB	8 KB	内部			
接続						
TSENDステータス	非アクティブ	送信	送信	送信	送信済み	非アクティブ

遅延呼び出しでの送信操作

送信操作が通信モジュール(CM)または 通信プロセッサ(CP)経由で実行されない場合、TSEND/TUSEND 命令のバージョン 3.0 と 4.0 は同じ動作を行います。

- 次の例では、TSEND または TUSEND が REQ パラメータの立ち上がりエッジで 1 回だけ呼び出されます(下図の呼び出し 2 を参照)。
- 遅延が十分に長く、データの転送が可能である場合、次の呼び出しで DONE パラメータが直接「1」に設定されます(ここでは呼び出し 3)。



送信操作が CM/CP 経由で実行される場合、TSEND/TUSEND 命令のバージョン 4.0 は異なる動作を行います。 この場合、TRCV/TURCV 命令の NDR パラメータによってデータの受信が確認されるまで、複数回にわたり命令を呼び出す必要があります。

割り込み接続での送信操作

送信操作中に接続が中断する場合、ERROR および STATUS パラメータがエラーとその原因を表示します。 次の例では、呼び出し番号 5 で接続が中断し、その結果、このエラーが STATUS パラメータで表示されます。

TSEND/TUSEND 命令のバージョン 4.0 以降では、その他の STATUS アラームが使用可能で、それに応じて評価を行うことができます(TSEND/TUSEND の説明を参照)。

TRCV/TURCV 命令での変更

接続確立での受信操作

送信中に遅延や割り込みが発生しない場合、TRCV/TURCV 命令のバージョン 3.0 と 4.0 は同じ動作を行います。

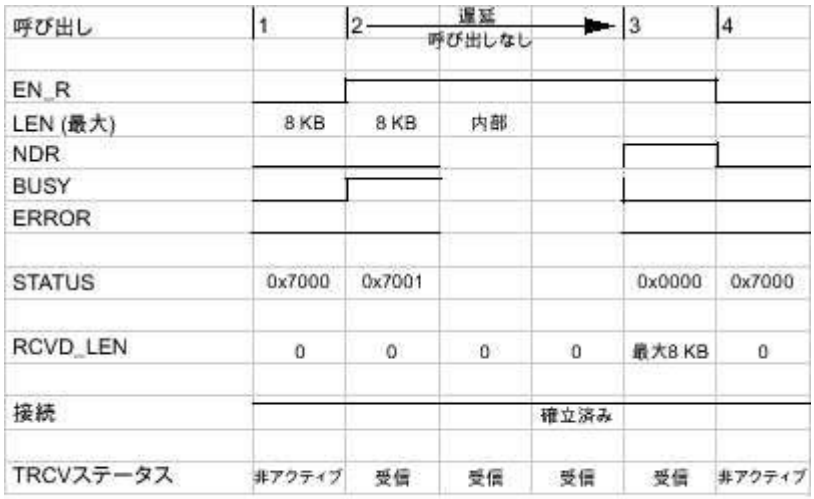
- TRCV を使用して、最大で 8 KB のデータを送信できます。 TURCV を使用して、最大で 1472 バイトを受信できます。 このデータ量は命令が完全に実行された場合の数値で、DONE パラメータが設定されるまで必要なすべての呼び出しが行われます。
- EN_R パラメータが「1」に設定される場合、命令はデータを受信します。
- LEN パラメータで指定された長さのデータが完全に受信されるまで、データの受信は完了しません。 受信データの長さが、RCVD_LEN 出力パラメータに出力されます。 その時に初めて、DATA パラメータで定義された領域のデータが使用可能になります。

呼び出し	1	2	3	4	5	6
EN_R						
LEN (最大)	8 KB	8 KB	内部			
NDR						
BUSY						
ERROR						
STATUS	0x7000	0x7001	0x7002	0x7002	0x0000	0x7000
RCVD_LEN	0	0	0	0	最大8 KB	0
接続				確立済み		
TRCVステータス	非アクティブ	受信	受信	受信	受信	非アクティブ

遅延呼び出しでの受信操作

送信操作が通信モジュール(CM)または 通信プロセッサ(CP)経由で実行されない場合、TRCV/TURCV 命令のバージョン 3.0 と 4.0 は同じ動作を行います。

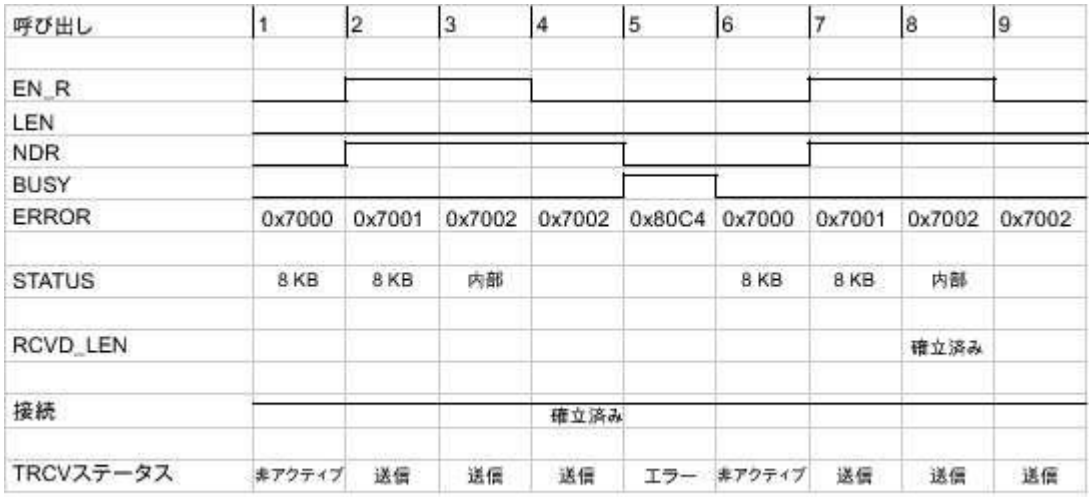
- 次の例では、TRCV または TURCV が EN_R パラメータの立ち上がりエッジで 1 回だけ呼び出されます(下図の呼び出し 2 を参照)。
- 遅延が十分に長く、データの転送が可能である場合、次の呼び出しで NDR パラメータが直接「1」に設定されます(ここでは呼び出し 3)。



送信操作が CM/CP 経由で実行される場合、TSEND/TUSEND 命令のバージョン 4.0 は異なる動作を行います。 この場合、NDR パラメータによってデータの受信が確認されるまで、複数回にわたり命令を呼び出す必要があります。

割り込み接続での受信操作

送信操作中に接続が中断する場合、ERROR および STATUS パラメータがエラーとその原因を表示します。 次の例では、呼び出し番号 5 で一時的な通信エラーが発生し、その結果が STATUS パラメータによって表示されます。



TRCV/TURCV 命令のバージョン 4.0 以降では、その他の STATUS アラームが使用可能で、それに応じて評価を行うことができます(TRCV/TURCV の説明を参照)。

ADHOC モードを使用した受信操作

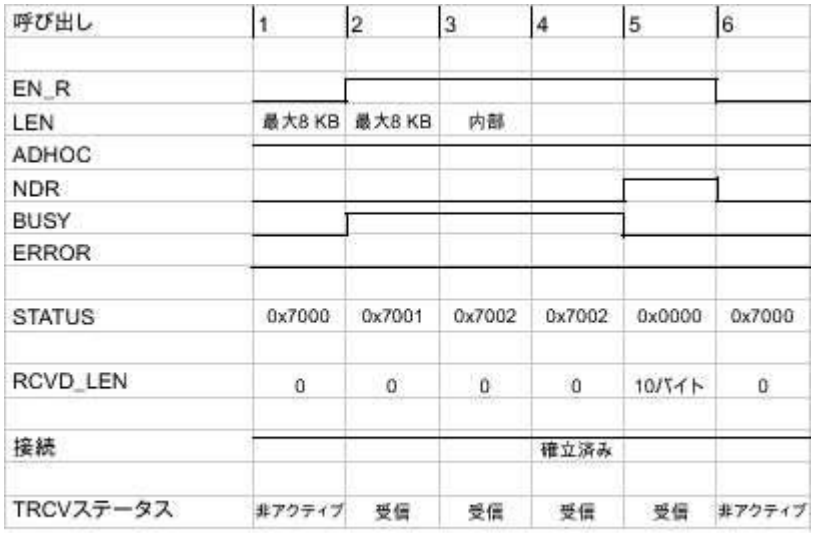
ADHOC モードは、TCP プロトコルのバリエーションでのみ使用可能です。 ADHOC モードを使用して、TRCV/TURCV 命令で可変長データを受信することができます。 ADHOC モードがアクティブである場合、少なくとも 1 バイトが転送された時に NDR パラメータでデータの受信が確認されます。

TRCV 3.0 以前(S7-1200 V4.0 以前)による ADHOC モードのデータ受信

TRCV の以前のバージョンでは、LEN パラメータを「0」に設定すると ADHOC モードが有効になりました。 次の例では、10 バイトのデータが呼び出し 5 で転送されます。

TRCV 3.0 以降(S7-1200 V4.0 以降または S7-1500)による ADHOC モードのデータ受信

TRCV 命令のバージョン 3.0 以降では、ADHOC モードは独自のパラメータ(ADHOC)で有効になります。



実行中にエラーが発生した場合、TRCV 命令のバージョン 4.0 以降では、その他の STATUS アラームが使用可能で、それに応じて評価を行うことができます。

送信操作が CM/CP 経由で実行される場合、TRCV/TURCV 命令のバージョン 4.0 は異なる動作を行います。 この場合、NDR パラメータによってデータの受信が確認されるまで、複数回にわたり命令を呼び出す必要があります。

MB_SERVER/MB_CLIENT 命令での変更

Modbus 命令のバージョン 3.1 と 4.0 の相違点

MB_SERVER/MB_CLIENT MODBUS 命令では、バージョン間で以下の相違が存在します。

- アドレスパラメータ
  - バージョン 3.1 では、Modbus TCP サーバーのアドレスデータは入力パラメータ「IP_x」で指定されます。
  - バージョン 4.0 は、この目的のために CONNECT 入力パラメータで TCON_IP_V4 および TCON_Configured システムデータタイプを使用します。
- 実行中にエラーが発生した場合、Modbus 命令のバージョン 4.0 以降では、その他の STATUS アラームが使用可能で、それに応じて評価を行うことができます。

詳細な情報は、MB_CLIENT (V3.1)および MB_CLIENT (V4.0)の命令説明や MB_SERVER (V3.1)および MB_SERVER (V4.0)の説明を参照してください。

Web サーバー

この章には下記に関する情報が記載されています：

- WWW: ユーザー定義の Web ページの同期化 (S7-1200, S7-1500)

WWW: ユーザー定義の Web ページの同期化

説明

命令 WWW は、CPU の Web サーバーを初期化し、ユーザー定義の Web ページを CPU のユーザープログラムと同期します。

ユーザー定義の Web ページと Web サーバーを一緒に使用すると、ブラウザによって CPU の設計済み Web ページに自由にアクセスできます。

ユーザー定義の Web ページでスクリプト命令(JavaScript など)と HTML コードを使用し、さらに処理を行うために Web ブラウザを経由して CPU にデータを転送し、CPU のオペランド領域からのデータを Web ブラウザで表示します。 ユーザープログラムの WWW 命令を呼び出し、ユーザープログラムと Web サーバーを同期し、初期化します。

初期化

ユーザー定義の Web ページは、CPU によって処理するためデータブロック内に「包括」されます。構成中に、ソースファイル(HTML ファイル、画面、JavaScript ファイル、...)から適切なデータブロックを生成する必要があります。Web コントロール DB には特殊なロールがあります(既定: DB 333)。これはステータス情報や制御情報が格納している上に、コード化された Web ページがある追加のデータブロックにリンクしています。 コード化された Web ページを有するデータブロックではフラグメント DB が呼び出されます。

データブロックが CPU にダウンロードされている場合、CPU はユーザー定義の Web ページがその内部にコード化されていることを「認識」していません。 たとえば、スタートアップ OB の命令「WWW」は CPU に対し、どの DB が Web コントロール DB であるかを通知します。 この初期化後に Web ブラウザを介してユーザー定義の Web ページにアクセスできます。

同期

ユーザープログラムとユーザー定義の Web ページを相互に対話させたい場合、命令 WWW を循環プログラム部分で使用する必要があります。

ユーザープログラムと Web ページ間の操作例

- 受信されたデータのチェック
- データをアセンブルし、Web ブラウザに戻して要求を作成する

この場合、現在のステータス情報を評価できなければならず、Web サーバーは Web ブラウザによって要求された Web ページのリリースなど、コントロール情報を受信する必要があります。

パラメータ

次の表に、「WWW」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
CTRL_DB	Input	DB_WWW	I、Q、M、D、L、または定数	ユーザー定義の Web ページを表すデータブロック(Web コントロール DB)
RET_VAL	Output	INT	I、Q、M、D、L	エラー情報

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

RET_VAL パラメータ

エラーコード (W#16#...)	説明
0000	エラーは発生しませんでした。 ユーザープログラムによってリリースする必要がある Web ページ要求はありません。
00xy	x: Web コントロール DB (CTRL_DB)の初期化中にエラーが発生したかどうかを示します。 x=0: エラーは発生しませんでした。 x=1: エラーが発生しました。 エラーは Web コントロール DB のバイト「CTRL_DB.last_error」でコード化されます。Web コントロール DB の説明を参照してください。 y: 保留中の要求数。 複数の要求が可能です(たとえば、要求「0」と「1」が保留中: y="3". y="1": 要求「0」 y="2": 要求「1」 y="4": 要求「2」 y="8": 要求「3」
803A	CPU に指定した Web コントロール DB がありません。
8081	Web コントロール DB のバージョンおよび形式が不正。
80C1	2 または 4 個の Web アプリケーションのみを実行できるため、たとえば Web アプリケーションを初期化するリソースがありません。

通信プロセッサ

この章には下記に関する情報が記載されています：

- SIMATIC NET CM/CP (S7-1200, S7-1500)
- (S7-1200, S7-1500)
- (S7-1200, S7-1500)
- MODBUS (RTU) (S7-1200, S7-1500)
- ポイントツーポイント (S7-1200)
- USS (S7-1200)
- MODBUS (RTU) (S7-1200)
- MODBUS (TCP) (S7-1200, S7-1500)

SIMATIC NET CM/CP

この章には下記に関する情報が記載されています：

- S7-1500 CM/CP
- S7-1200 CM/CP

S7-1500 CM/CP

この章には下記に関する情報が記載されています：

- 産業用イーサネット

産業用イーサネット

この章には下記に関する情報が記載されています：

- FTP サービスの命令

FTP サービスの命令

この章には下記に関する情報が記載されています：

- FTP サービスの FTP_CMD
- ファイル DB の構造と使用 - FTP クライアントモード

FTP サービスの FTP_CMD

この章には下記に関する情報が記載されています：

FTP_CMD の概要

意味

FTP_CMD 命令を使用して FTP 接続を確立し、FTP サーバーとの間でファイルを転送します。

データ転送は FTP または FTPS (セキュアな SSL 接続)を使用して実行できます。

注記

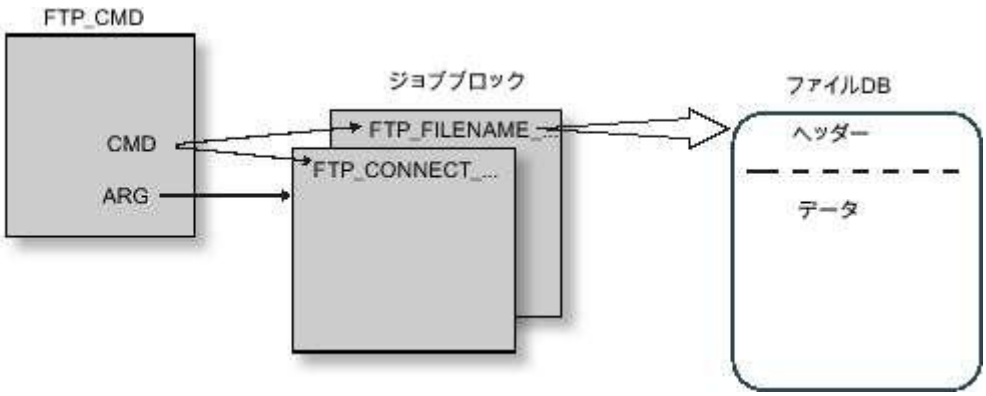
FTPS: 証明書の比較

FTPS は、FTP サーバーと FTP クライアントの間で証明書を比較する必要があります。 FTP サーバーが FTP クライアントの STEP 7 プロジェクト外で設定されている場合、証明書は FTP サーバーからインポートする必要があります。 FTP サーバーの証明書を信頼できる証明書として証明書マネージャにインポートします。

動作原理:

FTP_CMD 命令は、FTP コマンドが指定されているジョブブロック(ARG)を参照します。 このブロックは、FTP コマンド(CMD)のタイプに応じてパラメータ割り当てに異なるデータ構造体を使用します。 これらの異なる構造体に対して、最適なデータタイプ(UDT)が用意されています。

以下の図に、呼び出し構造体を示します。



ジョブブロック

以下のデータ構造体はジョブブロックとして使用されます。

- 接続の確立

  以下のアクセスのタイプを使用する接続確立については、さまざまなデータ構造体が用意されています。
  - FTP_CONNECT_IPV4: IPv4 準拠の IP アドレスを使用した接続の確立
  - FTP_CONNECT_IPV6: IPv6 準拠の IP アドレスを使用した接続の確立
  - FTP_CONNECT_NAME: サーバー名(DNS)を使用した接続の確立
- データ転送

  データ転送については、2 つの異なるデータ構造体が用意されています。
  - FTP_FILENAME: 完全なファイルにアクセスするためのデータ構造体
  - FTP_FILENAME_PART: データ領域に読み取りアクセスするためのデータ構造体

File_DB 内でのデータ転送

ジョブデータのヘッダーおよびユーザーデータの領域を含むデータブロックを使用して、データ転送を実行できます。 データブロックは、ジョブバッファ内で指定します。

CPU コンフィグレーションの要件

FTP アクセスを許可するには、以下の設定を使用します。

- [プロパティ|全般|保護]の CPU の設定データ内で、 [PUT/GET 通信の無効化]オプションを無効にします。
- ファイル DB として使用されているすべてのデータブロックに対して、「最適化したブロックアクセス」属性を無効にします。

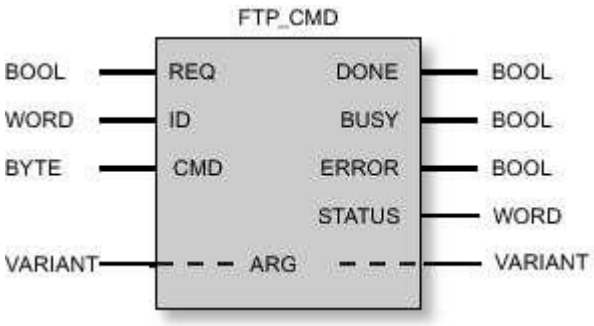
有効性

FTP_CMD 命令は以下のモジュールタイプと共に使用することができます。

- CP 1543-1

呼び出しインターフェース

FBD 表記での呼び出しインターフェース



入力パラメータ - FTP_CMD

入力パラメータの説明

「FTP_CMD」命令には、次の入力パラメータを指定します。

「FTP_CMD」命令の仮パラメータ - 入力パラメータ

パラメータ	宣言	データタイプ	メモリ領域	意味/備考
REQ	Input	BOOL	I、Q、M、D、L	信号立ち上がりエッジで送信ジョブを開始します。
ID *	INPUT	INT	1, 2 ... 64	FTP ジョブは FTP 接続で処理されます。 このパラメータは使用中の接続を識別します。
CMD *	INPUT	BYTE	次の表「コマンド」を参照してください。	命令が呼び出された際に実行される FTP コマンド。 表の後に、FTP コマンドタイプごとの値の範囲を記載します。 注記: ここで指定された FTP コマンドは、ジョブブロック(ARG パラメータ)内でも同じく指定する必要があります。 コマンドが CP ファームウェアでサポートされていない場合、STATUS = $8F6B_H$ によってエラーメッセージが出力されます。
ARG *	INPUT	VARIANT	次の表「コマンド」を参照してください。	ジョブブロック 該当する FTP コマンドに最適な実行パラメータを使用してデータ領域を参照します。 データタイプ(UDT)は FTP コマンドに応じて使い分けられます。 以下に、UDT を示します。 ANY データタイプは、ここで指定するポインタには使用できません。
* 入力パラメータ「ID」および「CMD」の値は、入力パラメータ「ARG」の値を上書きします。				

「CMD」パラメータ内の FTP コマンド

次の表に、「CMD」パラメータのコマンドの重要性、およびジョブブロックにどの UDT を指定するかを示します。

コマンドタイプ

CMD (コマンドタイプ)	関連するジョブブロック/UDT	意味/処理
0 (NOOP)	*	呼び出された FC がアクションを実行しません。 このパラメータが指定された場合、ステータスコードは以下のように設定されます。 DONE=1; ERROR=0; STATUS=0
1 (CONNECT)	FTP_CON-NECT_IPV4 FTP_CON-NECT_IPV6 FTP_CON-NECT_NAME	FTP 接続確立 このコマンドを使用して、FTP クライアントは FTP サーバーへの FTP 接続を確立します(ポート 21)。 この接続は、すべての以後の FTP コマンドに対して、ここで指定する接続 ID で使用可能です。 その後データはこのユーザーに対して指定された FTP サーバーによって交換されます。
2 (STORE)	FTP_FILENAME	このファンクション呼び出しは、データブロック(ファイル DB)を FTP クライアント(S7-CPU)から FTP サーバーに転送します。 注意: ファイル(ファイル DB)が FTP サーバーに既に存在する場合、ファイルは上書きされます。
3 (RETRIEVE)	FTP_FILENAME	このファンクション呼び出しは、ファイルを FTP サーバーから FTP クライアント(S7-CPU)に転送します。 注意: データブロック(ファイル DB)が FTP クライアントに既に存在する場合、データブロックは上書きされます。
4 (DELETE)	FTP_FILENAME	このファンクション呼び出しを使用して、FTP サーバー上のファイルを削除します。
5 (QUIT)	*	このファンクション呼び出しを使用して、この ID によって選択した FTP 接続を確立します。
6 (APPEND)	FTP_FILENAME	「STORE」と同様に、「APPEND」コマンドはファイルを FTP サーバーに保存します。 ただし「APPEND」を使用した場合、FTP サーバー上のファイルは上書きされません。 新しい内容が既存のファイルに追加されます。 ファイル(ファイル DB)が FTP サーバーに存在しない場合、ファイルは作成されます。
7 (RETR_PART)	FTP_FILE-NAME_PART	「RETR_PART」コマンド(一部を取得)を使用して、ファイルの一部を FTP サーバーから要求することができます。 非常に大きなファイルが関係する場合、これによって現在必要な部分のみを読み取るように制限することができます。 これを実行するには、ファイルの構造が分かっている必要があります。 FB40 で 2 つのパラメータ「OFFSET」および「LEN」を使用して必要な部分を入力します。
* コマンドタイプ 0 (NOOP)および 5 (QUIT)の場合、自由に選択可能なジョブブロック(UDT)を指定する必要があります。 これは評価されません。		

FTP_CMD のジョブブロック

意味

ARG パラメータを使用して「FTP_CMD」命令にジョブブロックを指定します。 構造体は FTP コマンドタイプによって異なります。 既定のデータタイプ(UDT)を使用することで、命令はジョブブロックのタイプを認識します。 以下に、次のジョブブロックに関連するデータタイプ(UDT)を記載します。

- IPv4 準拠の IP アドレスを使用した FTP 接続の確立
- IPv6 準拠の IP アドレスを使用した FTP 接続の確立
- サーバー名を使用した FTP 接続の確立
- 書き込み/読み取りアクセスおよびその他の FTP コマンド
- FTP コマンド「RETR_PART」

IPv4 準拠の IP アドレスを使用した FTP 接続確立用のジョブブロック

IPv4 準拠の IP アドレスを使用した FTP 接続の確立には、以下のデータ構造体を使用します。

FTP_CONNECT_IPV4

パラメータ	タイプ	値の範囲	意味/備考
InterfaceID	HW_ANY		モジュールの開始アドレス 命令を呼び出すとき、LADDR パラメータの CP のモジュール開始アドレスを転送します。 CP のモジュール開始アドレスは、 [プロパティ|アドレス|入力]の CPU の設定内に表示されています。
ID	CONN_OUC	1, 2...64	FTP ジョブは FTP 接続で処理されます。 このパラメータは使用中の接続を識別します。
ConnectionType	BYTE	0	接続タイプ「FTP」
ActiveEstablish-ment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP コマンド "CONNECT" 命令が呼び出された際に実行される FTP コマンド。 値の範囲は表 コマンドタイプに記載されています。 注記: ここで指定された FTP コマンドは、CMD 入力パラメータ内でも同じく指定する必要があります。
CertIndex	BYTE	0 = FTP 1 = FTPS	ここでは、プロトコルタイプ FTP または FTPS を選択します。 FTPS に関する注:

			FTP サーバーが FTP クライアントの STEP 7 プロジェクト外で設定されている場合、証明書は FTP サーバーからインポートする必要があります。
UserName	STRING[32]	'benutzer'	FTP サーバーへのログイン用ユーザー名
Password	STRING[32]	'passwort'	FTP サーバーへのログイン用パスワード
FTPserverI-Paddr	IP_V4	ADDR(1) ... ADDR(4)	Array[1..4] of Byte としての FTP サーバーの IP アドレス。1 バイトでアドレスの 1 ブロックを指定。 例: ADDR(1)は 1 番目のアドレスブロック(アドレスの 1 番目のバイト)を指定します。

IPv6 準拠の IP アドレスを使用した FTP 接続確立用のジョブブロック

IPv6 準拠の IP アドレスを使用した FTP 接続の確立には、以下のデータ構造体を使用します。

FTP_CONNECT_IPV6

パラメータ	タイプ	値の範囲	意味/備考
InterfaceID	HW_ANY		モジュールの開始アドレス 命令を呼び出すとき、LADDR パラメータの CP のモジュール開始アドレスを転送します。 CP のモジュール開始アドレスは、[プロパティ|アドレス|入力]の CPU の設定内に表示されています。
ID	CONN_OUC	1, 2...64	FTP ジョブは FTP 接続で処理されます。このパラメータは使用中の接続を識別します。
ConnectionType	BYTE	0	接続タイプ「FTP」
ActiveEstablish-ment	BOOL	TRUE	
FTPCmd	BYTE	1	FTP コマンド "CONNECT" 命令が呼び出された際に実行される FTP コマンド。 値の範囲は表 コマンドタイプに記載されています。 注記: ここで指定された FTP コマンドは、CMD 入力パラメータ内でも同じく指定する必要があります。
CertIndex	BYTE	0 = FTP 1 = FTPS	ここでは、プロトコルタイプ FTP または FTPS を選択します。 FTPS に関する注: FTP サーバーが FTP クライアントの STEP 7 プロジェクト外で設定されている場合、証明書は FTP サーバーからインポートする必要があります。
UserName	STRING[32]	'user'	FTP サーバーへのログイン用ユーザー名
Password	STRING[32]	'password'	FTP サーバーへのログイン用パスワード

FTPserverI-Paddr	IP_V6	ADDR(1) ... ADDR(16)	Array[1..16] of Byte としての FTP サーバーの IP アドレス。2 バイトでアドレスの 1 ブロックを指定。 例: ADDR(1) + ADDR(2)で 1 番目のアドレスブロックを指定します。

サーバー名を使用した FTP 接続確立用のジョブブロック

サーバー名を指定する FTP 接続の確立には、以下のデータ構造体を使用します。 サーバー名は、DNS を使用して IP アドレスに割り当てられます。

FTP_CONNECT_NAME

パラメータ	タイプ	値の範囲	意味/備考
InterfaceID	HW_ANY		モジュールの開始アドレス 命令を呼び出すとき、LADDR パラメータの CP のモジュール開始アドレスを転送します。 CP のモジュール開始アドレスは、[プロパティ|アドレス|入力]の CPU の設定内に表示されています。
ID	CONN_OUC	1, 2...64	FTP ジョブは FTP 接続で処理されます。このパラメータは使用中の接続を識別します。
ConnectionType	BYTE	0	接続タイプ「FTP」
ActiveEstablish-ment	BOOL	TRUE	
FTPcmd	BYTE	1	FTP コマンド "CONNECT" 命令が呼び出された際に実行される FTP コマンド。 値の範囲は表 コマンドタイプに記載されています。 注記: ここで指定された FTP コマンドは、CMD 入力パラメータ内でも同じく指定する必要があります。
CertIndex	BYTE	0 = FTP 1 = FTPS	ここでは、プロトコルタイプ FTP または FTPS を選択します。 FTPS に関する注: FTP サーバーが FTP クライアントの STEP 7 プロジェクト外で設定されている場合、証明書は FTP サーバーからインポートする必要があります。
UserName	STRING[32]	'benutzer'	FTP サーバーへのログイン用ユーザー名
Password	STRING[32]	'passwort'	FTP サーバーへのログイン用パスワード
FTPserverName	STRING[254]		FTP サーバーの IP アドレス

書き込み/読み取りアクセスおよびその他の FTP コマンドのためのジョブブロック

保存、取得、削除、および追加の FTP コマンドには、以下のデータ構造体を使用します。

FTP_FILENAME

パラメータ	タイプ	値の範囲	意味/備考
InterfaceID	HW_ANY		モジュールの開始アドレス 命令を呼び出すとき、LADDR パラメータの CP のモジュール開始アドレスを転送します。 CP のモジュール開始アドレスは、[プロパティ|アドレス|入力]の CPU の設定内に表示されています。
ID	CONN_OUC	1, 2...64	FTP ジョブは FTP 接続で処理されます。このパラメータは使用中の接続を識別します。
ConnectionType	BYTE	0	接続タイプ「FTP」
ActiveEstablish-ment	BOOL	TRUE	
FTPcmd	BYTE	2, 3, 4, 6	FTP コマンド "STORE / RETRIEVE / DELETE / APPEND" 命令が呼び出された際に実行される FTP コマンド。 値の範囲は表 コマンドタイプに記載されています。 注記: ここで指定された FTP コマンドは、CMD 入力パラメータ内でも同じく指定する必要があります。
CertIndex	BYTE	0 = FTP 1 = FTPS	ここでは、プロトコルタイプ FTP または FTPS を選択します。 FTPS に関する注: FTP サーバーが FTP クライアントの STEP 7 プロジェクト外で設定されている場合、証明書は FTP サーバーからインポートする必要があります。
DataBlockNum-ber	UINT		ここで指定するデータブロックには、読み取り/書き込みされる DB が含まれます。
LenFilename	UINT	0...1000	ファイル名の合計長を指定する「LenFilename」パラメータは評価されず、 「Filename」パラメータの文字列の長さ情報が評価されます。
Filename	ARRAY[0..3] OF STRING[220]		宛先またはソースファイルのファイル名。

「RETR_PART」FTP コマンド用のジョブブロック

「RETR_PART」FTP コマンドには、以下のデータ構造体を使用します。

FTP_FILENAME_PART

パラメータ	タイプ	値の範囲	意味/備考
InterfaceID	HW_ANY		モジュールの開始アドレス 命令を呼び出すとき、LADDR パラメータの CP のモジュール開始アドレスを転送します。 CP のモジュール開始アドレスは、[プロパティ|アドレス|入力]の CPU の設定内に表示されています。
ID	CONN_OUC	1, 2...64	FTP ジョブは FTP 接続で処理されます。このパラメータは使用中の接続を識別します。
ConnectionType	BYTE	0	接続タイプ「FTP」
ActiveEstablish-ment	BOOL	TRUE	
FTPcmd	BYTE	7	FTP コマンド "RETR_PART" 命令が呼び出された際に実行される FTP コマンド。 値の範囲は表 コマンドタイプに記載されています。 注記: ここで指定された FTP コマンドは、CMD 入力パラメータ内でも同じく指定する必要があります。
CertIndex	BYTE	0 = FTP 1 = FTPS	ここでは、プロトコルタイプ FTP または FTPS を選択します。 FTPS に関する注: FTP サーバーが FTP クライアントの STEP 7 プロジェクト外で設定されている場合、証明書は FTP サーバーからインポートする必要があります。
Offset	DWORD		ファイルの読み取りが開始するバイト単位のオフセット。
Length	DWORD		「OFFSET」で指定された値で読み取りを開始するバイト単位の下位長さ。 特殊機能: • 「DW#16#FFFFFFFF」が指定されている場合、ファイルの使用可能な残りが読み取られます。 その他のエラーが発生しない場合は結果 OK (DONE = 1, STATUS = 0) • OFFSET > オリジナルファイルの長さの場合: 宛先ファイルの長さ(ファイル DB で ACT_LENGTH): CPU で 0 バイト。 その他のエラーが発生しない場合は結果 OK (DONE = 1, STATUS = 0) • OFFSET + LEN > オリジナルファイルの長さ(かつ LEN ≠ 0xFFFFFFFF)の場合:

			宛先ファイルの長さ(ファイル DB で ACT_LENGTH):「OFFSET」で開始する使用可能なバイト。 その他のエラーが発生しない場合は結果 OK (DONE = 1, STATUS = 0)
DataBlockNum-ber	UINT		ここで指定するデータブロックには、読み取り/書き込みされる DB が含まれます。
LenFilename	UINT	0...1000	ファイル名の合計長を指定する「LenFilename」パラメータは評価されず、 「Filename」パラメータの文字列の長さ情報が評価されます。
Filename	ARRAY[0..3] OF STRING[220]		宛先またはソースファイルのファイル名。

出力パラメータおよびステータス情報 FTP_CMD

パラメータ BUSY、DONE および ERROR

パラメータ BUSY、DONE、ERROR、および STATUS を使用して実行のステータスを制御します。BUSY パラメータは、処理ステータスを示します。 DONE パラメータで、ジョブが正常に実行されたかどうかをチェックします。 "FTP_CMD"の実行中にエラーが発生すると、ERROR パラメータがセットされます。 エラー情報は、STATUS パラメータに出力されます。

次の表に、パラメータ BUSY、DONE、および ERROR パラメータの関係を示します。

BUSY	DONE	ER-ROR	説明
1	-	-	ジョブが処理中です。
0	1	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーのため終了されました。 エラーの原因は、STATUS パラメータに明記されています。
0	0	0	新しいジョブが割り当てられませんでした。

ステータスコードの評価

注記
STATUS で 8FxxH によってコーディングされるエントリについては、『STEP 7 標準ファンクションおよびシステムファンクションリファレンスマニュアル』の情報を参照してください。
RET_VAL 出力パラメータを使用したエラー評価を説明している章には、詳細情報が記載されています。

FTP_CMD: DONE および ERROR との関連での STATUS パラメータの意味

DONE	ERROR	STATUS	意味
0	0	0000_H	実行中のジョブはありません。
1	0	0000_H	ジョブがエラーなしで完了しました。
0	0	7001_H	ジョブが初めてトリガされました(BUSY=1)
0	0	7002_H	ジョブがまだ実行中です(BUSY=1)
0	0	$80C4_H$	通信エラー(一時的に発生し、通常ユーザープログラムでジョブを繰り返すことが最善です)。
0	0	8183_H	設定がジョブパラメータに一致しません。
0	1	8401_H	不明なエラー 考えられる原因: • 接続でタイムアウトが検出されました。 • FTP サーバーが接続を中止しました。 対策:

			• 接続を再確立するために、QUIT コマンドおよび CONNECT コマンドを送信します。
0	1	8402_H	接続にエラーステータスがあります。 接続のタイムアウトが超過したか、FTP サーバーが接続を終了した可能性があります。 対策: 接続を再確立するために、QUIT コマンドおよび CONNECT コマンドを再送信します。
0	1	8403_H	ログインに失敗しました。
0	1	8404_H	FTP サーバーを取得できません。
0	1	8405_H	転送に失敗しました。
0	1	8406_H	現在の操作がタイムアウトしました。
0	1	8407_H	ファイルが FTP サーバー上に見つかりませんでした。
0	1	8408_H	転送できません。
0	1	8409_H	ファイルをフェッチできませんでした。
0	1	8410_H	TCP ポートのデータ接続用の設定に失敗しました。
0	1	8411_H	オフセット情報が一致しません。
0	1	8412_H	指定されたディレクトリ変更のエラー
0	1	8413_H	データ受信のエラー
0	1	8414_H	データ送信のエラー
0	1	8415_H	指定された CMD (コマンドタイプ)がクライアントによって拒否されました。
0	1	8416_H	接続が FTP サーバーによって閉じました。
0	1	8418_H	ユーザーデータのエラー。 考えられる原因: • ファイル名が空です。 • データ長が「0」 • など
0	1	8419_H	データ接続を開くためのソケットリソースがありません。
0	1	8420_H	制御接続を開くためのソケットリソースがありません。
0	1	8421_H	読み出すファイル DB を開く際のエラー。
0	1	8422_H	書き込むファイル DB を開く際のエラー。
0	1	8423_H	FTP サーバーへの接続の確立に失敗しました。
0	1	8424_H	内部エラー
0	1	8425_H	ドメイン名のフォーマットエラー
0	1	8426_H	保留中 DNS 照会が多すぎます。
0	1	8427_H	指定した DNS サーバーが、指定したドメイン名を割り当てることができませんでした。
0	1	8428_H	接続リソースが使用できません。

0	1	8429_H	不明なチャンネル ID
0	1	8430_H	ファイル DB が短すぎます。
0	1	8431_H	ファイル DB に書き込む際のエラー。
0	1	8432_H	ファイル DB から読み出す際のエラー。
0	1	8433_H	ファイル DB にアクセスする際のエラー。
0	1	8434_H	操作が中止されました。
0	1	8435_H	チャンネルがリセットされます。
0	1	8436_H	予期せぬサーバー応答
0	1	8437_H	証明書を検証できませんでした。
0	1	8438_H	不明なエラーが発生しました
0	1	8439_H	FTP コマンドによりエラーが発生しました。 FTP サーバーで原因を探す必要があります(REST コマンド)。
0	1	8440_H	FTP サーバーが、要求された SSL プロトコルをサポートしていません。
0	1	8446_H	FTP パスワードが FTP サーバーに送信された後、FTP サーバーから予期しないコードが返されました。
0	1	8451_H	伝送モードをバイナリから ASCII に変更しようとした際にエラーが発生しました。
0	1	8455_H	メモリ要求が CM/CP で失敗しました。
0	1	8460_H	SSL/TLS の処理で問題が発生しました。
0	1	8469_H	インターフェースエラー 指定した出力インターフェースが使用できませんでした。 対策: 発信接続に使用するインターフェースを設定します。
0	1	8475_H	SSL 証明書または SSH md5 指紋が信頼できると判断できませんでした。
0	1	8476_H	FTP サーバーからの受信なし。 現在のステータスで、正しくない応答が仮定されたはずです。
0	1	8477_H	指定した「Crypto エンジン」(暗号モジュール)が見つかりませんでした。
0	1	8478_H	選択済み SSL「Crypto エンジン」を既定で設定しようとして失敗しました。
0	1	8480_H	FTP クライアントの証明書で問題が発生しました。
0	1	8481_H	指定された番号が使用できませんでした。
0	1	8482_H	FTP サーバーがサポートされないコーディングを使用しています。
0	1	8484_H	最大ファイルサイズを超過しました。
0	1	8485_H	ファイル DB が送信処理中に変更されたか、ファイル DB の構造が間違っています。
0	1	8489_H	データは送信できませんでした。 FTP サーバーでの操作に十分なメモリが使用できません。

0	1	8492_H	ファイルが既に存在します。 ファイルは上書されません。
0	1	8496_H	SSL CA 証明書の読み取りで問題が発生しました。
0	1	8497_H	SSH セッションで予期しないエラーが発生しました。
0	1	8498_H	SSL 接続を終了できません。
0	1	8499_H	ソケットの送信/受信の準備ができていません。 準備完了まで待ってから、もう一度試してください。
0	1	8501_H	FTP サーバーによる SSL 証明書チェックが失敗しました。
0	1	8507_H	FTP サーバーを待機中、動作中の FTP セッションの間に接続の確立でタイムアウトが発生しました。
0	1	$8F55_H$	ヘッダーステータスビット: ロック
0	1	$8F56_H$	ファイル DB ヘッダー内の新しいビットがリセットされませんでした
0	1	$8F6B_H$	考えられる原因: • CMD パラメータの不正な値 値 0～15 が許可されています。 • FB40 コマンドはサポートされていません。 考えられる原因: CP 上の誤ったファームウェア 修復方法: ファームウェアの更新(古い CP では、FB 40 ではなくファンクション FC 40～FC 44 を使用)
0	1	$8F7F_H$	内部エラー(例: 無効な任意の参照)。

ファイル DB の構造と使用 - FTP クライアントモード

この章には下記に関する情報が記載されています：

- FTP サービス用データブロック(ファイル DB)の構造 - FTP クライアントモード
- テンプレートとしての FILE_DB_HEADER データブロック - FTP クライアントモード

FTP サービス用データブロック(ファイル DB)の構造 - FTP クライアントモード

動作原理:

FTP によってデータを転送するには、S7 ステーションの CPU 上にデータブロック(ファイル DB)を作成します。 これらのデータブロックは、FTP サービスによって転送可能なファイルとして処理できるように特定の構造を持っている必要があります。 データ構造体は以下のセクションから成ります。

- セクション 1: ファイル DB ヘッダー(20 バイトの固定長を持つ)
- セクション 2: ユーザーデータ(可変長および構造体を持つ)

CPU コンフィグレーションの要件

FTP アクセスを許可するには、以下の設定を使用します。

- [プロパティ|全般|保護]の CPU の設定データ内で、 [PUT/GET 通信の無効化]オプションを無効にします。
- ファイル DB として使用されているすべてのデータブロックに対して、「最適化したブロックアクセス」属性を無効にします。

FTP クライアントモードのファイル DB ヘッダー

注記: ここで説明するファイル DB ヘッダーは、サーバーモードのファイル DB ヘッダーと大部分は同じです。 違いは以下のパラメータに関するものです。

- WRITE_ACCESS
- FTP_REPLY_CODE

パラメータ	タイプ	値/意味	電源
EXIST	BOOL	EXIST ビットは、ユーザーデータ領域に有効なデータが含まれているかどうかを示します。 取得 FTP コマンドは EXIST=1 の場合のみジョブを実行します。 • 0: ファイル DB に有効なユーザーデータが含まれていません(ファイルが存在しません)。 • 1: ファイル DB に有効なユーザーデータが含まれています(ファイルが存在します)。	DELETE FTP コマンドは EXIST=0 を設定します。 STORE FTP コマンドは EXIST=1 を設定します。
LOCKED	BOOL	LOCKED ビットはファイル DB へのアクセスを制限するために使用します。 • 0: ファイル DB にアクセス可能です。	「STORE」および「RETRIEVE」の FTP コマンドは、実行されると LOCKED=1 を設定します。 S7 CPU 上のユーザープログラムは、データの整合性を達成するた

		• 1: ファイル DB がロックされています。	めに書き込みアクセス中に LOCKED をセットまたはリセットできます。 ユーザープログラムでの推奨シーケンス: 1. LOCKED ビットが 0 かどうかチェックします。 2. WRITEACCESS ビット = 0 にセットします。 3. LOCKED ビットが 0 かどうかチェックします。 4. LOCKED ビット = 1 にセットします。 5. データを書き込みます。 6. LOCKED ビット = 0 にセットします。
NEW	BOOL	NEW ビットは、データが最後の読み取りアクセスから変更されたかどうかを示します。 • 0: ファイル DB の内容は最後の書き込みアクセスから変更されていません。 S7 CPU のユーザープログラムは最終変更を登録しました。 • 1: S7 CPU のユーザープログラムはまだ最後の書き込みアクセスを登録していません。	実行後、stor FTP コマンドは NEW=1 をセットします。 データの読み出し後、S7-CPU 内のユーザープログラムは新しい「RETRIEVE」コマンドを可能にするために NEW=0 に設定する必要があります。
WRITE_ACCESS	BOOL	• 0: ユーザープログラム (FTP クライアントブロック)は、S7 CPU 上のファイル DB に書き込みアクセス権を持っています。 • 1: ユーザープログラム (FTP クライアントブロック)は、S7 CPU 上のファイル DB に書き込みアクセス権を持っていません。	DB の構成中、ビットは初期値にセットされます。 推奨事項: 可能な限り、このビットは変更なしのままにしてください。 特殊な状況で、操作中の調整が可能です。
ACT_LENGTH	DINT	ユーザーデータ領域の現在の長さ。 このフィールドの内容は EXIST = 1 の場合にのみ有効です。	現在の長さは書き込みアクセス後に更新されます。
MAX_LENGTH	DINT	ユーザーデータ領域の最大長(DB 全体の長さマイナス 20 バイトのヘッダー分)。	最大長は DB の構成中に指定する必要があります。 この値も操作中にユーザープログラムによって変更することができます。

FTP_RE-PLY_CODE	INT	FTP からの最後の応答コードをバイナリ値として含む符号なし整数(16 ビット)。このフィールドの内容は EXIST = 1 の場合にのみ有効です。	これは FTP コマンドが実行されると FTP クライアントによって更新されます。
DATE_TIME	DATE_AND_TIME	ファイルへの最終変更の日付と時刻。 このフィールドの内容は EXIST = 1 の場合にのみ有効です。	現在の日付は書き込みアクセス後に更新されます。 時刻転送のためのファンクションを使用する場合、エントリは転送された時刻に対応します。 時刻転送のためのファンクションを使用しない場合、相対的な時刻が入力されます。 参照は、以下の初期値のスタートアップ時刻です。 01.01.1994 00:00h).

テンプレートとしての FILE_DB_HEADER データブロック - FTP クライアントモード

意味

データタイプ FILE_DB_HEADER はファイル DB ヘッダーの作成のため、あらかじめ定義済みです。

動作原理

FTP によってデータを転送するには、S7 ステーションの CPU 上にデータブロック(ファイル DB)を作成します。 これらのデータブロックは、FTP サービスによって転送可能なファイルとして処理できるように特定の構造を持っている必要があります。 データ構造体は以下のセクションから成ります。

- セクション 1: ファイル DB ヘッダー(20 バイトの固定長を持つ)
- セクション 2: ユーザーデータ(可変長および構造体を持つ)

下記で説明されている手順に従ってください。

1. FTP 命令を使用してユーザープログラムを作成する CPU に、タイプ「グローバル DB」のデータブロックを作成します。
2. ファイル DB の開始行として使用する行を選択します。
3. [データタイプ]列で、最下部のドロップダウンリストからタイプ「FILE_DB_HEADER」の構造体要素を選択します。

   結果: 作成するファイル DB にヘッダー構造体付きのデータ構造体が必要です。
4. 新規作成されたデータブロックのプロパティ(ショートカットメニュー)を選択し、「最適化したブロックアクセス」属性を無効にします。

注記

[(新規ブロックの追加)]ファンクション - タイプ選択

新しいデータブロックを作成するとき、ドロップダウンリストの[タイプ]エントリで[FILE_DB_HEADER]ブロックタイプも使用可能です。 このオプションは使用しないでください。この方法で作成されるデータブロックにはヘッダー構造体しか含まれず、ユーザーデータを保存するために必要な領域によって拡張できません。

FILE_DB_HEADER データブロック -ファイル DB ヘッダーの例およびテンプレート

宣言表示で、以下の構造体が表示されます。

アドレス	名前	タイプ	初期値	コメント
0.0		STRUCT		
+0.0	bit08	BOOL	FALSE	予約済み
+0.1	bit09	BOOL	FALSE	予約済み
+0.2	bit10	BOOL	FALSE	予約済み
+0.3	bit11	BOOL	FALSE	予約済み
+0.4	bit12	BOOL	FALSE	予約済み
+0.5	bit13	BOOL	FALSE	予約済み

+0.6	bit14	BOOL	FALSE	予約済み
+0.7	bit15	BOOL	FALSE	予約済み
+1.0	EXIST	BOOL	FALSE	TRUE の場合: ファイル DB に有効なデータがありません。
+1.1	LOCKED	BOOL	FALSE	TRUE の場合: 内容が変更されたため、ファイル DB がブロックされています。
+1.2	NEW	BOOL	FALSE	TRUE の場合: ファイル DB の内容が新しくなっており、上書きすることはできません。
+1.3	WRITE_ACCESS	BOOL	FALSE	TRUE の場合: FTP サーバーに書き込みアクセスがあります。
+1.4	bit04	BOOL	FALSE	予約済み
+1.5	bit05	BOOL	FALSE	予約済み
+1.6	bit06	BOOL	FALSE	予約済み
+1.7	bit07	BOOL	FALSE	予約済み
+2.0	ACT_LENGTH	DINT	L#0	内容の現在の長さ(単位バイト、20 バイトのヘッダー分は除く)
+6.0	MAX_LENGTH	DINT	L#0	内容の最大長(単位バイト、20 バイトのヘッダー分は除く)
+10.0	FTP_REPLY_CODE	INT	0	リモート FTP サーバーからの最後の応答情報。
+12.0	DATE_TIME	DATE_AND_TIME	DT#00-1-1-0:0:0.000	ファイル DB の内容に対する最終変更の日付と時刻。
=20.0		END_STRUCT		

モードによる違い

FTP クライアントモードのファイル DB ヘッダー

ここで説明するファイル DB ヘッダーは、FTP クライアントモードおよび FTP サーバーモードのファイル DB ヘッダーと大部分は同じです。 違いは以下のパラメータに関するものです。

- WRITE_ACCESS
- FTP_REPLY_CODE

S7-1200 CM/CP

この章には下記に関する情報が記載されています：

- 遠隔制御

遠隔制御

この章には下記に関する情報が記載されています：

- **遠隔制御命令**

遠隔制御命令

この章には下記に関する情報が記載されています：

TC_CON: GSM ネットワーク経由の接続の確立

意味

「TC_CON」命令を使用すると、S7-1200 が CP 1242-7 によって以下のタイプの接続を確立できます。

- ISO-ON-TCP

  接続パートナーは CP 1242-7 です。

  ISO-on-TCP 接続は、「GPRS 直接」モードでのみ使用します。
- UDP

  任意の接続パートナーが可能です。
- SMS

  接続パートナーは SMS クライアントです。
- 遠隔制御接続

  接続パートナーは、遠隔制御サーバーを経由して到達できる遠隔制御サーバーまたは他のステーションのいずれかです。

TC_CON は 1 つの接続しか確立しません。 CP 1242-7 のモードおよび使用しているプロトコルに応じて、1 つの CP につき、1 つの ID (下記参照)で最大 3～5 の同時接続がサポートされています。同時接続の最大数は、CP のパフォーマンスデータに記載されています。

CONNECT パラメータは接続記述子としてシステムデータタイプ(SDT)の構造体を持つデータブロック(DB)を使用します。

接続固有の SDT「TCON_...」を使用して、必要な接続タイプが定義されます(下記参照)。 上述した接続タイプごとに、次の SDT のいずれかを割り当てる必要があります。

- ISO-on-TCP 接続用の TCON_IP_RFC
- UDP 接続用の TCON_IP_V4
- SMS 接続用の TCON_PHONE
- 遠隔制御接続用の TCON_WDC

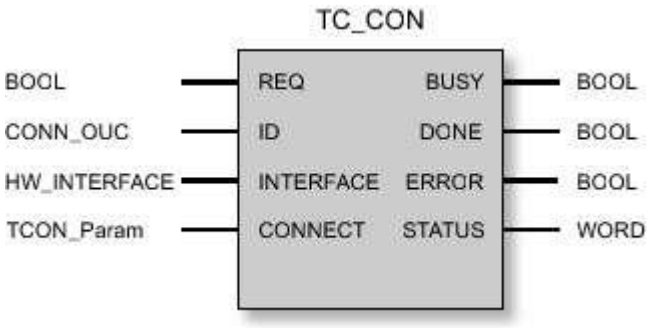
これらの SDT の「ActiveEstablished」パラメータは、接続の確立がアクティブまたはパッシブであるかどうかも指定します。

これらの SDT に対するパラメータ設定については、TCON_...: 遠隔制御接続の確立のための SDT を参照してください。

ID パラメータは、GPRS 接続を参照します。 割り当てられる ID は、CPU 内で一意となる必要があります。

INTERFACE パラメータは、要求されたローカル CP の GPRS インターフェースを参照します。これは STEP 7 から取得する必要があります。

FBD 表記での呼び出しインターフェース



仮パラメータの説明

次の表は、TC_CON 命令の仮パラメータすべてを説明しています。

パラメータ	宣言	データタイプ	値の範囲	説明
REQ	INPUT	BOOL	0, 1	命令が開始され、立ち上がりエッジでステータスコードが初期化されました。 立ち上がりエッジがある場合の DONE、ERROR および STATUS ステータスコードの更新。
ID	INPUT	CONN_OUC	1...07FF$_h$	関連する接続への参照。 ID が割り当てられています。 ID の値も、CONNECT パラメータのシステムデータタイプ(SDT)によって要求されます。
INTER-FACE	INPUT	HW_INTER-FACE		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
CON-NECT	INOUT	TCON_Par-am	「TCON_...: 遠隔制御確立用の SDT」も参照してください。	接続確立用のデータブロックへの参照。 タイプ TCON_IP_RFC、TCON_IP_V4、TCON_PHONE、または TCON_WDC は、関連する接続に最適なデータブロックの構造体を指定します。 SDT では、パラメータ「ActiveEs-tablished」(アクティブ/パッシブ接続の確立)にご注意ください。
ENO	OUTPUT	BOOL	0: エラー 1: エラーなし	イネーブル出力 命令でランタイムエラーが発生する場合、ENO は「0」にセットされます。
BUSY	OUTPUT	BOOL	0: 命令の実行は開始、完了または中止していません。	命令の処理ステータスの表示

			1: 命令は実行中です。	
DONE	OUTPUT	BOOL	0: - 1: 命令は正常に実行されました。	このパラメータは、ジョブがエラーなく完了したかどうかを示します。 パラメータ ERROR と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
ERROR	OUTPUT	BOOL	0: - 1: エラー	エラーコード パラメータ DONE と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
STATUS	OUTPUT	WORD		ステータスコード パラメータ ERROR と ERROR を組み合わせた場合の意味については、命令のコードを参照してください。

コード BUSY、DONE および ERROR

BUSY = 0 の場合のみ DONE と ERROR のコードが関連します。

BUSY	DONE	ERROR	意味
0	0	0	ジョブは実行されません。

次の表に、DONE と ERROR のその他のすべてのコード組み合わせを示します。

呼び出し時、命令は数秒間にわたり BUSY = 1 状態のままになります。 以下の状況では、BUSY 状態「1」がより長い間続くことがあります。

- アクティブ ISO-on-TCP 接続で、パートナーに到達できない場合。
- パッシブ接続で、受信するフレームがない場合。

コード DONE、ERROR および STATUS

次の表に、ユーザープログラムで評価する必要のある DONE、ERROR および STATUS パラメータによって形成される条件コードを示します。

DONE	ERROR	STATUS	意味
1	0	0000_H	ジョブはエラーなく実行されました
0	0	7000_H	ジョブ処理は動作中ではありません(最初の命令呼び出し)
0	0	7001_H	ジョブ処理が開始しました(最初の命令呼び出し)
0	0	7002_H	ジョブ処理がすでに動作中です(BUSY = 1 の場合の命令呼び出しの更新)
0	1	8086_H	ID の無効な値

0	1	8087_H	接続の最大数に達しました。もう接続できません。
0	1	$80E3_H$	ID が既に別の接続によって使用されています。 これは、TC_SEND、TC_RECV、または TC_DISCON に対して、現在 BUSY が TRUE であることを意味します。 EN_R of TC_RECV が常に TRUE である場合、このステータスコードが出力されます。 これにより、通常 TC_RECV が呼び出されます。 この状況への対策: TC_CON または TC_DISCON が呼び出される前に、EN_R をオフにします。 TC_CON がエラーなく実行された場合にのみ、EN_R を再度オンにすることが可能です。
0	1	$80E6_H$	進行中の照会はありません(命令の呼び出しが開始していません)
0	1	$80E8_H$	リモートパートナーに到達できません。 接続パラメータを確認してください。 「GPRS 直接」モードでは、パートナーには到達できるがパートナーが接続要求を許可しない場合にメッセージが出力されます。
0	1	$80EB_H$	要求が一時的に拒否されました(TC_CON はすでに同じ宛先アドレスで呼び出されています)。
0	1	$80EC_H$	Listener Port を開けませんでした: 接続パラメータを確認してください。
0	1	$80F2_H$	CP のモードが不正です: • 遠隔制御接続は、「遠隔制御」モードでのみ許可されています。 • ISO-on-TCP 接続は、「GPRS 直接」モードでのみ許可されています
0	1	$80F3_H$	データ送信の接続エンドポイントが空いていません: • 使用する接続を減らす、または • 使用するパッシブ接続を減らす、または • NTP を終了してください。 CP 1242-7 の同時接続の最大数にご注意ください。
0	1	$80F4_H$	接続エンドポイントを生成できません。 呼び出しを繰り返します。 必要に応じて、接続パラメータを確認してください。
0	1	$80F5_H$	無効な接続エンドポイント:TC_CON による接続の確立に失敗しました。 ブロック呼び出しを繰り返します。
0	1	$80F6_H$	呼び出されたデータブロック内にパラメータのフォーマットエラーが存在します(不正な長さ、不正なフォーマット、無効な値、または 20 文字を超える TCON_Phone の電話番号)。 「TC_CON...」 SDT の設定をチェックしてください。

TC_DISCON: GSM ネットワーク経由の接続の終了

意味

CP 1242-7 を有する S7-1200 上で TC_DISCON 命令を実行すると、TC_CON 命令によって確立された ISO-on-TCP、UDP、SMS または遠隔制御接続が終了します。

詳細情報については、「TC_CON」命令の説明に記載されている接続タイプを参照してください。

TC_DISCON は、遠隔制御サーバーへの接続を論理的にのみ終了します。 TCP/IP レベルでは、接続は保持されます。

遠隔制御サーバーへの接続を物理的に終了する場合は、STEP 7 の「遠隔制御サーバー」パラメータグループ内で、接続を「一時接続」として設定します。一時ステーションはデータ送信後、自動的に接続を終了します。

注記

TC_DISCON による以降のプログラムブロック実行の停止

TC_DISCON を呼び出すことで、同一の接続 ID(パラメータ「ID」)およびインターフェース (パラメータ「INTERFACE」) で呼び出された TC_CON、TC_SEND、および TC_RECV ブロックの実行が終了します。 これにより、これらのブロックを ERROR のシグナルを発信します。

TC_CON が「Error = 1」を示す場合は、TC_DISCON を呼び出してはなりません。

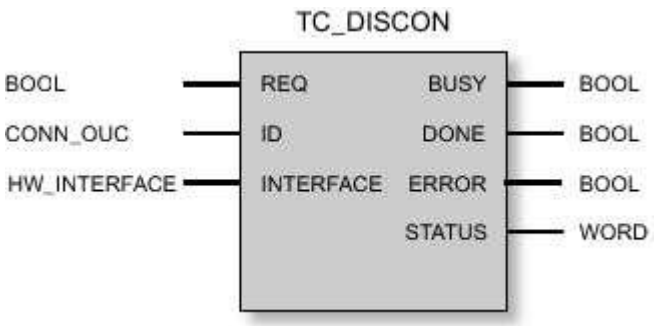
TC_CON が「ERROR」を示す場合は、接続が確立されません。 この場合、TC_DISCON は呼び出してはなりません。

この状況で TC_DISCON が呼び出されると、接続 ID (「ID」)が短期間予約され、直後に呼び出された TC_CON が ERROR および STATUS 80E3 を示します。

ID パラメータは、GPRS 接続を参照します。 ID は CPU 内で一意であり、かつ TC_CON で使用される ID と同じである必要があります。

INTERFACE パラメータは、要求されたローカル CP の GPRS インターフェースを参照します。この値は INTERFACE として TC_CON に使用される値と同じです。

FBD 表記での呼び出しインターフェース



仮パラメータの説明

次の表に、TC_DISCON 命令の仮パラメータすべてを示します。

パラメータ	宣言	データタイプ	値の範囲	説明
REQ	INPUT	BOOL	0, 1	命令が開始され、立ち上がりエッジでステータスコードが初期化されました。 立ち上がりエッジがある場合の DONE、ERROR および STATUS ステータスコードの更新。
ID	INPUT	CONN_OUC	1...07FF$_h$	関連する接続への参照
INTER-FACE	INPUT	HW_INTER-FACE		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
ENO	OUTPUT	BOOL	0: エラー 1: エラーなし	イネーブル出力 命令でランタイムエラーが発生する場合、ENO は「0」にセットされます。
BUSY	OUTPUT	BOOL	0: 命令の実行は開始、完了または中止していません。 1: 命令は実行中です。	命令の処理ステータスの表示
DONE	OUTPUT	BOOL	0: - 1: 命令は正常に実行されました。	このパラメータは、ジョブがエラーなく完了したかどうかを示します。 パラメータ ERROR と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
ERROR	OUTPUT	BOOL	0: - 1: エラー	エラーコード パラメータ DONE と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
STATUS	OUTPUT	WORD		ステータスコード パラメータ ERROR と ERROR を組み合わせた場合の意味については、命令のコードを参照してください。

コード BUSY、DONE および ERROR

BUSY = 0 の場合のみ DONE と ERROR のコードが関連します。

BUSY	DONE	ERROR	意味
0	0	0	命令はまだ呼び出されていません。

次の表に、DONE と ERROR のその他のすべてのコード組み合わせを示します。

注記
呼び出し時、命令は数秒間にわたり BUSY = 1 状態のままになります。

コード DONE、ERROR および STATUS

次の表に、ユーザープログラムで評価する必要のある DONE、ERROR および STATUS パラメータによって形成される条件コードを示します。

DONE	ERROR	STATUS	意味
1	0	0000_H	ジョブはエラーなく実行されました
0	0	7000_H	ジョブ処理は動作中ではありません(最初の命令呼び出し)
0	0	7001_H	ジョブ処理が開始しました(最初の命令呼び出し)
0	0	7002_H	ジョブ処理がすでに動作中です(BUSY = 1 の場合の命令呼び出しの更新)
0	1	8086_H	ID の無効な値
0	1	$80E4_H$	不明な ID: この ID の接続が TC_CON によって確立されていません。
0	1	$80E6_H$	進行中の照会はありません(命令の呼び出しが開始していません)
0	1	$80F5_H$	無効な接続エンドポイント: • TC_CON による接続の確立に失敗しました。または、 • リモートパートナーによって接続が終了されました。
0	1	$80F6_H$	呼び出されたデータブロック内にパラメータのフォーマットエラーが存在します(不正な長さ、不正なフォーマット、または無効な値) 「TC_CON...」 SDT の設定をチェックしてください。

TC_SEND: GSM ネットワーク経由のデータ送信

意味

TC_SEND 命令を使用すると、次のタイプのプログラム済み接続を経由してデータを送信できます。

- ISO-ON-TCP 接続
- UDP 接続
- SMS 接続

  SMS メッセージの送信は、CP の STEP 7 コンフィグレーションで設定されている場合のみサポートされます。
- 遠隔制御接続

> **注記**
>
> **複数の宛先への SMS メッセージの送信**
>
> 複数の宛先に同一の SMS メッセージを送信する場合、それぞれの宛先への接続を確立する必要があります。

詳細情報については、「TC_CON」命令の説明に記載されている接続タイプを参照してください。

ID パラメータは、GPRS 接続を参照します。 ID の値は、TC_CON によって ID に使用される値に対応していなければなりません。

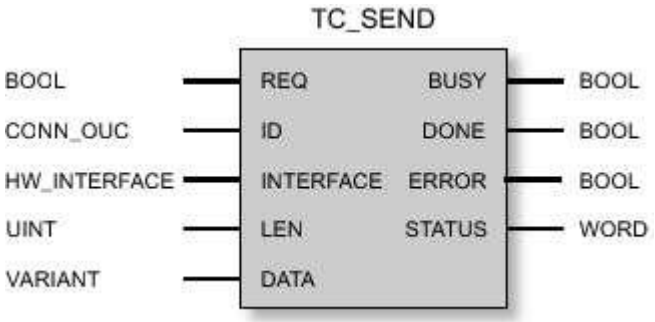
INTERFACE パラメータは、要求されたローカル CP の GPRS インターフェースを参照します。この値は INTERFACE として TC_CON に使用される値と同じです。

送信されるデータ量は LEN パラメータを使用して指定されます。

DATA に指定されるデータ領域のサイズは、少なくとも LEN に設定されるバイト数と同じでなければなりません。 DATA に指定されるデータ領域の許容データタイプは、BOOL 以外のすべてと BOOL の ARRAY です。

送信されるデータのコピー先アドレス(接続パートナー)は TC_CON 命令に設定されます。

FBD 表記での呼び出しインターフェース



仮パラメータの説明

次の表に、TC_SEND 命令の仮パラメータすべてを説明します。

パラメータ	宣言	データタイプ	値の範囲	説明
REQ	INPUT	BOOL	0, 1	命令が開始され、立ち上がりエッジでステータスコードが初期化されました。 立ち上がりエッジがある場合の DONE、ERROR および STATUS ステータスコードの更新。
ID	INPUT	CONN_OUC	1...07FF$_h$	関連する接続への参照
INTER-FACE	INPUT	HW_INTER-FACE		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
LEN	INPUT	UINT	1...2048	送信されるデータのバイト数。 この値は 1 以上、2048 以下である必要があります。 この値は、DATA の範囲のサイズと一致する必要があります。
DATA	INOUT	VARIANT		CPU のデータ送信領域へのアドレス参照 *
ENO	OUTPUT	BOOL	0: エラー 1: エラーなし	イネーブル出力 命令でランタイムエラーが発生する場合、ENO は「0」にセットされます。
BUSY	OUTPUT	BOOL	0: 命令の実行は開始、完了または中止していません。 1: 命令は実行中です。	命令の処理ステータスの表示
DONE	OUTPUT	BOOL	0: - 1: 命令は正常に実行されました。	このパラメータは、ジョブがエラーなく完了したかどうかを示します。 ** パラメータ ERROR と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
ERROR	OUTPUT	BOOL	0: - 1: エラー	エラーコード パラメータ DONE と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
STATUS	OUTPUT	WORD		ステータスコード パラメータ ERROR と ERROR を組み合わせた場合の意味については、命令のコードを参照してください。
* SMS テキスト用の DATA パラメータの特殊機能については、次のセクションを参照してください。				

** フレームの送信後、TC_SEND は DONE に「1」をセットします。以下の応答に注意してください:
ISO-on-TCP 接続が切断された場合、送信者は 1～2 分経過しないとそれが分かりません。 送信者側で TC_SEND が DONE に「1」をセットしても、転送されたデータが欠落している可能性があります。
フレームを受信してから TC_RECV の開始までに ISO-on-TCP 接続が中断されると、送信者側で TC_SEND が DONE に「1」をセットしたとしても、転送されたデータが欠落している可能性があります。

DATA パラメータによる SMS テキストの設定

この命令は、DATA パラメータのタイプ VARIANT のポインタが参照するデータを SMS テキストとして送信します。

DATA が SMS テキストを取得するためにデータタイプ STRING のオペランドを参照する場合、最初の 2 バイトが文字列の長さの情報とともに転送されます。

SMS メッセージのテキストが正しく表示されるように送信する方法の 1 つは、変換ファンクション Strg_TO_Chars を使用してテキスト文字列を Array of BYTE または Array of CHAR に変換することです。EN パラメータの Strg_TO_Chars は、TC_SEND によって出力パラメータ ENO にリンクされています。

CP は SMS テキストについて、ウムラウト(ü、ä など)といったすべての特殊文字をサポートしていません。 GSM 03.38 の仕様が適用されます。 GSM ネットワークプロバイダによってその他の制限が課される場合もあります。

コード BUSY、DONE および ERROR

BUSY = 0 の場合のみ DONE と ERROR のコードが関連します。

BUSY	DONE	ERROR	意味
0	0	0	ジョブは実行されません。

次の表に、DONE と ERROR のその他のすべてのコード組み合わせを示します。

コード DONE、ERROR および STATUS

次の表に、ユーザープログラムで評価する必要のある DONE、ERROR および STATUS パラメータによって形成される条件コードを示します。

DONE	ERROR	STATUS *	意味
1	0	0000_H	ジョブはエラーなく実行されました
0	0	7000_H	ジョブ処理は動作中ではありません(最初の命令呼び出し)
0	0	7001_H	ジョブ処理が開始しました(最初の命令呼び出し)
0	0	7002_H	ジョブ処理がすでに動作中です(BUSY = 1 の場合の命令呼び出しの更新)
0	1	8086_H	ID の無効な値
0	1	$80E0_H$	内部エラー

			フレームを遠隔制御サーバーに直接送信する場合(モード「Telecontrol」)、送信サイクルタイムが 1 秒以上となっていることを確認してください。
0	1	$80E1_H$	タイムアウト: • CP 1242-7 の設定で[接続モニタ時間]の値を増加させるか、 • 接続パートナーを確認してください。
0	1	$80E4_H$	不明な ID: TC_CON の最初の呼び出し。
0	1	$80E6_H$	進行中の照会はありません(命令の呼び出しが開始していません)
0	1	$80E7_H$	送信対象のデータは完全に転送されませんでした。 ジョブを繰り返します。
0	1	$80E8_H$	リモートパートナーに到達できません。 接続パラメータを確認してください。 「GPRS 直接」モードでは、パートナーには到達できるがパートナーが接続要求を許可しない場合にメッセージが出力されます。
0	1	$80E9_H$	リモートパートナーによって接続が確立しました。 接続パートナーを確認してください。 必要に応じて、TC_DISCON によって接続を終了し、TC_CON によって接続を再度確立します。
0	1	$80EA_H$	リモートパートナーからのエラーメッセージ。 • 接続パートナーを確認してください。 通信パートナーで、「TC_RECV」命令を有効にします。 • 必要に応じて、TC_DISCON によって接続を終了し、TC_CON によって接続を再度確立します。
0	1	$80EF_H$	SMS は送信できませんでした。 • 宛先アドレス(宛先サブスクライバの電話番号)が存在しているかチェックしてください。 • 挿入されている SIM カードが SMS の送信を許可しているかチェックしてください。 • 送信された SMS テキストの長さをチェックしてください。 160 文字を超える SMS テキストは送信されません。 • データブロック TCON_PHONE の作成時、ブロックアクセスに対して[標準]オプションが選択されていることを確認してください。
0	1	$80F1_H$	SMS メッセージの送信は、CP の STEP 7 コンフィグレーションでは有効になりません。 CP のコンフィグレーションで[SMS を許可]オプションを有効にしてください。
0	1	$80F4_H$	接続エンドポイントを生成できません。 接続パートナーを確認してください。
0	1	$80F5_H$	無効な接続エンドポイント: • TC_CON による接続の確立に失敗しました。 または • リモートパートナーによって接続が終了されました。 TC_DISCON の最初の呼び出し。

0	1	$80F6_H$	呼び出されたデータブロック内にパラメータのフォーマットエラーが存在します(不正な長さ、不正なフォーマット、または無効な値)。 「TC_CON...」 SDT の設定をチェックしてください。
* ここに記載されていないその他のステータスは、「RDREC」または「WRREC」命令の中ほどの 2 つのステータスバイト(STATUS[2]、STATUS[3])のステータス表示にあります。			

TC_RECV: GSM ネットワーク経由のデータ受信

意味

「TC_RECV」命令を使用すると、次のタイプのプログラム済み接続を経由してデータを送信できます。

- ISO-ON-TCP 接続
- SMS 接続

  SMS メッセージを受信するには、送信元の電話番号を受信 CP の STEP 7 コンフィグレーション内で設定する必要があります(許可されている電話番号)。 送信元は、CLIP ファンクションでサポートされている必要があります。

  接続パートナーの電話番号は、「TCON_PHONE」SDT 内に入力する必要があります。

  ウェイクアップ SMS メッセージは除去されます。
- 遠隔制御接続

注記

異なる送信元からの SMS メッセージの受信

異なる送信元から SMS メッセージを受信する場合、2 つの方法があります。

- 複数の接続を設定します(TC_CON, TC_RECV, TC_DISCON)。

  または
- 要求されたデータブロック「TCON_PHONE」で設定された接続が 1 つしかない場合は、「PhoneNumber」パラメータには電話番号を入力することはできません。 メッセージの受信時には、このパラメータはすべての許可された接続パートナー用のプレースホルダとして解釈されます。

詳細情報については、「TC_CON」命令の説明に記載されている接続タイプを参照してください。

ID パラメータは、GPRS 接続を参照します。 ID の値は、TC_CON によって ID に使用される値に対応していなければなりません。

INTERFACE パラメータは、要求されたローカル CP の GPRS インターフェースを参照します。この値は INTERFACE として TC_CON に使用される値と同じです。

受信対象の最大データ量は LEN パラメータを使用して指定されます。

DATA に指定されるデータ領域のサイズは、少なくとも LEN に設定されるバイト数と同じでなければなりません。 DATA に指定されるデータ領域の許容データタイプは、BOOL 以外のすべてと BOOL の ARRAY です。 受信したデータは、送信側パラメータが同じデータタイプを使用したかのように変換されます。

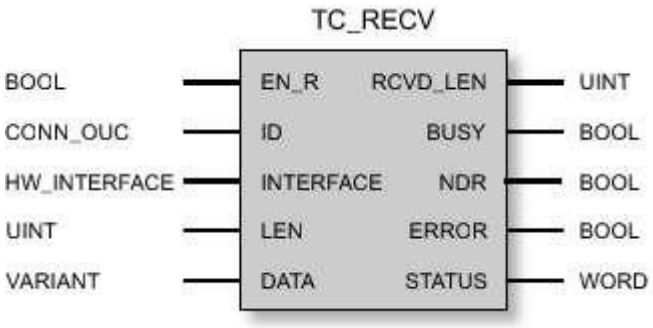
TC_RECV の接続記述子として使用される DB (システムデータタイプ)は、TC_SEND に使用される DB とは別のものである必要があります。

SMS メッセージの保存

受信した SMS メッセージは、 CP 1242-7 (25 のストレージ空間)、および SIM カード(ストレージ空間数はさまざま)に保持的に保存されます。

- SMS メッセージは TC_RECV によって読み取られると、ストレージ空間から削除されます。
- ストレージ空間がすべて割り当てられた状態で新しい SMS メッセージを受信すると、最も古い SMS メッセージが削除されます。

FBD 表記での呼び出しインターフェース



仮パラメータの説明

次の表に、TC_RECV 命令の仮パラメータすべてを示します。

パラメータ	宣言	データタイプ	値の範囲	説明
EN_R	INPUT	BOOL	0: データ受信はロックされました 1: データ受信は有効になりました	データの受信を有効化/ロックします。 • ブロックバージョン 1.1: 1 から 0 に設定すると、ブロックは無効になります。 • ブロックバージョン 1.0: 1 から 0 に設定すると、プログラムブロックは再度データを受信します(DONE = 0 かつ ERROR = 0 となるまで)。 TC_CON のステータスコード 80E3 の情報に注意してください。
ID	INPUT	CONN_OUC	1...07FF$_h$	関連する接続への参照
INTERFACE	INPUT	HW_INTERFACE		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
LEN	INPUT	UINT	1...2048	受信対象のデータの(最小)バイト数、最大 2048
DATA	INOUT	VARIANT		CPU のデータ受信領域へのアドレス参照 *
ENO	OUTPUT	BOOL	0: エラー 1: エラーなし	イネーブル出力 命令でランタイムエラーが発生する場合、ENO は「0」にセットされます。
RCVD_LEN	OUTPUT	UINT		受信したデータのバイト数

BUSY	OUTPUT	BOOL	0: 命令の実行は開始、完了または中止していません。 1: 命令は実行中です。	命令の処理ステータスの表示
DONE	OUTPUT	BOOL	0: - 1: 命令は正常に実行されました。	このパラメータは、ジョブがエラーなく完了したかどうかを示します。 パラメータ ERROR と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
ERROR	OUTPUT	BOOL	0: - 1: エラー	エラーコード パラメータ DONE と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
STATUS	OUTPUT	WORD		ステータスコード パラメータ ERROR と ERROR を組み合わせた場合の意味については、命令のコードを参照してください。
* SMS テキスト用の DATA パラメータの特殊機能については、次のセクションを参照してください。				

DATA パラメータによる SMS テキストの設定

この命令は、CPU のデータ領域に対する、DATA パラメータのタイプ VARIANT のポインタを使用して、受信した SMS テキストを参照します。

DATA が SMS テキストを取得するためにデータタイプ STRING のオペランドを参照する場合、その SMS テキストの最初の 2 バイトは SMS テキストではなく、データタイプ STRING の長さの情報として解釈されます。

SMS メッセージのテキストが正しく表示されるように受信する方法の 1 つは、変換ファンクション Chars_TO_Strg を使用して Array of BYTE または Array of CHAR をテキスト文字列に変換することです。EN パラメータの Chars_TO_Strg は、TC_RECV の出力パラメータ ENO にリンクされています。

CP は SMS テキストについて、ウムラウト(ü、ä など)といったすべての特殊文字をサポートしていません。 GSM 03.38 の仕様が適用されます。 GSM ネットワークプロバイダによってその他の制限が課される場合もあります。

コード BUSY、DONE および ERROR

BUSY = 0 の場合のみ DONE と ERROR のコードが関連します。

BUSY	DONE	ERROR	意味
0	0	0	ジョブは実行されません。

次の表に、DONE と ERROR のその他のすべてのコード組み合わせを示します。

コード DONE、ERROR および STATUS

次の表に、ユーザープログラムで評価する必要のある DONE、ERROR および STATUS パラメータによって形成される条件コードを示します。

DONE	ERROR	STATUS *	意味
1	0	0000_H	ジョブはエラーなく実行されました
0	0	7000_H	ジョブ処理は動作中ではありません(最初の命令呼び出し)
0	0	7001_H	ジョブ処理が開始しました(最初の命令呼び出し)
0	0	7002_H	ジョブ処理がすでに動作中です(BUSY = 1 の場合の命令呼び出しの更新)
0	1	$80A3_H$	• 既存の接続を再確立しようとしています。 • 存在していない接続を終了しようとしています。
0	1	$80E0_H$	内部エラー
0	1	8086_H	ID の無効な値
0	1	$80E4_H$	不明な ID: TC_CON の最初の呼び出し。
0	1	$80E6_H$	進行中の照会はありません(命令の呼び出しが開始していません)
0	1	$80F5_H$	無効な接続エンドポイント: • TC_CON による接続の確立に失敗しました。 または • リモートパートナーによって接続が終了されました。 TC_DISCON の最初の呼び出し。
0	1	$80F6_H$	呼び出されたデータブロック内にパラメータのフォーマットエラーが存在します(不正な長さ、不正なフォーマット、または無効な値) 「TC_CON...」 SDT の設定をチェックしてください。
* ここに記載されていないその他のステータスは、「RDREC」または「WRREC」命令の中ほどの 2 つのステータスバイト(STATUS[2]、STATUS[3])のステータス表示にあります。			

TC_CONFIG: 設定データの CP への転送

意味

TC_CONFIG 命令を使用すると、STEP 7 で設定された CP 1242-7 のパラメータを変更できます。 設定した値は、決して上書きされません。 上書きされた値は、TC_CONFIG が再度呼び出されるまで、またはステーションが再起動(電源サイクル後のコールドリスタート)するまで有効です。

CP の STEP 7 設定データを恒久的に変更する必要がある場合、ステーションが再起動(コールドリスタート)するたびに命令を再度呼び出すか、変更されたプロジェクトをステーションにダウンロードする必要があります。

CONFIG パラメータは、構成データのあるメモリ領域をポイントします。 設定データはデータブロック(DB)内に格納されます。 DB の構造は、システムデータタイプ(SDT) IF_CONF で指定します。

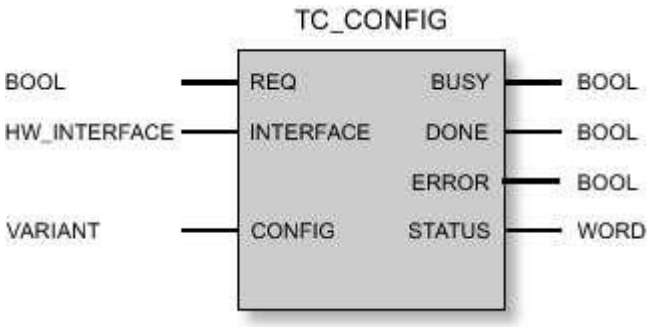
CP 上で変更する設定データは、必要に応じてそれぞれのパラメータについて IF_CONF「IF_CONF_...」のブロック内にまとめます。

命令の結果として変更しないパラメータは、IF_CONF には含めません。 これらのパラメータは、STEP 7 で設定された値を維持します。

IF_CONF に値を割り当てる際の詳細情報については、セクション「IF_CONF: 遠隔制御構成データの SDT」を参照してください。

INTERFACE パラメータは、要求されたローカル CP の GPRS インターフェースを参照します。

FBD 表記での呼び出しインターフェース



仮パラメータの説明

次の表に、TC_CONFIG 命令の仮パラメータすべてを示します。

パラメータ	宣言	データタイプ	値の範囲	説明
REQ	INPUT	BOOL	0, 1	命令が開始され、立ち上がりエッジでステータスコードが初期化されました。 立ち上がりエッジがある場合の DONE、ERROR および STATUS ステータスコードの更新。

INTER-FACE	INPUT	HW_INTER-FACE (WORD)		ローカル CP 1242-7 のインターフェースへの参照
CONFIG	INOUT	VARIANT	「IF_CONF: 遠隔制御構成データの SDT」も参照してください。	収集された変更対象の設定データのあるメモリ領域への参照
ENO	OUTPUT	BOOL	0: エラー 1: エラーなし	イネーブル出力 命令でランタイムエラーが発生する場合、ENO は「0」にセットされます。
BUSY	OUTPUT	BOOL	0: 命令の実行は開始、完了または中止していません。 1: 命令は実行中です。	命令の処理ステータスの表示
DONE	OUTPUT	BOOL	0: - 1: 命令は正常に実行されました。	このパラメータは、ジョブがエラーなく完了したかどうかを示します。 パラメータ ERROR と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
ERROR	OUTPUT	BOOL	0: - 1: エラー	エラーコード パラメータ DONE と STATUS を組み合わせた場合の意味については、命令のコードを参照してください。
STATUS	OUTPUT	WORD		ステータスコード パラメータ ERROR と ERROR を組み合わせた場合の意味については、命令のコードを参照してください。

コード BUSY、DONE および ERROR

BUSY = 0 の場合のみ DONE と ERROR のコードが関連します。

BUSY	DONE	ERROR	意味
0	0	0	ジョブは実行されません。

次の表に、DONE と ERROR のその他のすべてのコード組み合わせを示します。

コード DONE、ERROR および STATUS

次の表に、ユーザープログラムで評価する必要のある DONE、ERROR および STATUS パラメータによって形成される条件コードを示します。

DONE	ERROR	STATUS	意味
1	0	0000_H	ジョブはエラーなく実行されました
0	0	7000_H	ジョブ処理は動作中ではありません(最初の命令呼び出し)
0	0	7001_H	ジョブ処理が開始しました(最初の命令呼び出し)
0	0	7002_H	ジョブ処理がすでに動作中です(BUSY = 1 の場合の命令呼び出しの更新)
0	1	$80E6_H$	進行中の照会はありません(命令の呼び出しが開始していません)
0	1	$80EB_H$	照会が一時的に拒否されました(CP は現在 STEP 7 によって設定中です)。
0	1	$80F6_H$	呼び出されたデータブロック内にパラメータのフォーマットエラーが存在します(不正な長さ、不正なフォーマット、または無効な値) 「IF_CONF」SDT を確認します。
0	1	$80F7_H$	設定データのパラメータフィールド内に不正な ID が存在します。 「IF_CONF」SDT を確認します。

TCON_...: 遠隔制御接続の確立のための SDT

TC_CON 命令で使用するシステムデータタイプ TCON_...

TC_CON 命令を使用して遠隔制御接続を設定するには、接続記述子に命令の CONNECT パラメータが使用されます。

接続記述子はシステムデータタイプ(SDT)の構造体によって指定されます。 関連 SDT の構造体には、リモート通信パートナーとの接続を確立するのに必要なパラメータが含まれます。

リモート通信パートナーによって異なる接続タイプの場合、次の SDT が使われます。

- CP 1242-7 を搭載した IPv4 ステーションへの ISO-on-TCP 接続用の TCON_IP_RFC
- IPv4 ステーションへの UDP 接続用の TCON_IP_V4 (送信のみ)
- SMS クライアントへの接続の TCON_PHONE
- 遠隔制御サーバーを経由して到達できる遠隔制御サーバーまたはステーションへの接続の TCON_WDC

接続記述子のパラメータは、SDT と同じタイプのデータブロックで割り当てられます。

タイプ TCON_...の DB の作成

関連する DB のデータタイプは、キーボードを使用して入力する必要があります。 これらは選択リストには表示されません。 データタイプは大文字と小文字を区別しません。

TCON_... DB を作成するには、以下に記載された手順に従います。

1. ブロックアクセスが「標準」のタイプ「グローバル DB」のデータブロックを作成します。
2. データタイプのセル内で名前を割り当て、必要なタイプを入力して、DB のパラメータ設定のテーブル内に SDT を作成します(たとえば、"TCON_IP_RFC")。

   SDT とそのパラメータが作成されます(下記参照)。
3. 各 SDT タイプについて、以下に記載されているパラメータを設定します。

予約済みビットは表示されません。

IPv4 ステーションへの接続のシステムデータタイプ TCON_IP_RFC

この接続タイプは、固定 IP アドレスを持つ通信パートナーへの ISO-on-TCP 接続でのみサポートされています。 「GPRS 直接」モードに CP を設定する必要があります。

TCON_IP_RFC のパラメータ

バイト	パラメータ	データタイプ	初期値	説明
0 ... 1	InterfaceID	HW_ANY		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
2 ... 3	ID	CONN_OUC	1...07FF$_h$	GPRS 接続への参照。 割り当てられる ID は、CPU 内で一意となる必要があります。

				ここで、TC_CON 命令の ID パラメータと同じ値を使用する必要があります。
4	ConnectionType	BYTE	W#16#0C	プロトコル変数 12 (C_h): ISO-on-TCP 接続
5	ActiveEstablished	BOOL		接続確立のタイプの識別子: • 0: パッシブ接続の確立 • 1: アクティブ接続の確立
6 ... 7	-	-	-	- 予約済み-
8 ... 11	RemoteAddress	IP_V4		接続パートナーの IP アドレス
	ADDR	Array [1...4] of Byte		関連する接続パートナーの IP アドレス
12 ... 13	RemoteTSelector	TSelector		リモート T セレクタ
	TSelLen	UINT		リモート T セレクタ「RemoteTSelector」の長さ
14 ... 45	TSel	Array [1...32] of Byte	any	接続のリモート転送セレクタ •「ActiveEstablished」 = 1 の場合: アクティブ接続の確立では、ローカルパートナーの T セレクタが接続パートナーの T セレクタと同一である必要があります(リモートパートナーでのパッシブ接続の確立)。 • それぞれ「ActiveEstablished」 = 0 の場合 (ローカルでのパッシブ接続の確立、リモートでのアクティブ接続の確立)
46 ... 47	LocalTSelector	TSelector		ローカル T セレクタ
	TSelLen	UINT		ローカル T セレクタ「LOCAL_TSel」の長さ
48 ... 79	TSel	Array [1...32] of Byte	any	接続のローカル転送セレクタ •「ActiveEstablished」 = 1 の場合: アクティブ接続の確立では、ローカルパートナーの T セレクタが接続パートナーの T セレクタと同一である必要があります(リモートパートナーでのパッシブ接続の確立)。 • それぞれ「ActiveEstablished」 = 0 の場合 (ローカルでのパッシブ接続の確立、リモートでのアクティブ接続の確立)

IPv4 ステーションへの接続のシステムデータタイプ TCON_IP_V4

この接続タイプは、固定 IP アドレスを持つ通信パートナーへの UDP 接続での送信のみでサポートされています。

受信するには、ActiveEstablished = 0 がセットされている必要があります。

TCON_IP_V4 のパラメータ

バイト	パラメータ	データタイプ	初期値	説明
0 ... 1	InterfaceID	HW_ANY		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
2 ... 3	ID	CONN_OUC	1...07FF$_h$	GPRS 接続への参照。 割り当てられる ID は、CPU 内で一意となる必要があります。 ここで、TC_CON 命令の ID パラメータと同じ値を使用する必要があります。
4	ConnectionType	BYTE	W#16#0B	プロトコル変数 11 (B$_h$): UDP 接続
5	ActiveEstablished	BOOL		接続確立のタイプの識別子: • 0: パッシブ接続の確立 データの送受信のための設定 • 1: アクティブ接続の確立 データ送信のみの設定。
6 ... 9	RemoteAddress	IP_V4		接続パートナーの IP アドレス
	ADDR	Array [1...4] of Byte		4 バイト(ADDR[1] ... ADDR[4])で IP アドレスの 4 ブロックを指定します。
10 ... 11	RemotePort	UINT	1...65535	接続パートナーの IP ポート ActiveEstablished = 0.の場合は関連なし
12 ... 13	LocalPort	UINT	1...65535	ローカル IP ポート(「0」は許可されていません) ActiveEstablished = 1.の場合は関連なし

SMS 接続用のシステムデータタイプ TCON_PHONE

注記

許可されている電話番号

送信側の通信パートナーが電話番号に基づいて許可されている場合のみ、CP は SMS を承認します。 これらの番号は、STEP 7 の CP に対して[許可されている電話番号]リストに設定されます。

SMS テキスト

- 送信する SMS メッセージ用にプログラムされた SMS テキストは、「TC_SEND」命令の DATA パラメータを使用してアクセスされます。
- 受信した SMS メッセージのテキストは、「TC_RECV」命令の DATA パラメータによって CPU のアドレス領域に割り当てられます。

TCON_PHONE のパラメータ

バイト	パラメータ	データタイプ	初期値	説明
0 ... 1	InterfaceID	HW_ANY		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)

2 ... 3	ID	CONN_OUC	1...07FF$_h$	GPRS 接続への参照。 割り当てられる ID は、CPU 内で一意となる必要があります。 ここで、TC_CON 命令の ID パラメータと同じ値を使用する必要があります。
4	ConnectionType	BYTE	W#16#0E	プロトコル変数 14 (E$_h$): SMS 接続
5	ActiveEstablished	BOOL		接続確立タイプの識別子(CP 1242-7 に関連なし): • 0: パッシブ接続の確立(ここでは関連なし) • 1: アクティブ接続の確立
6...7	-	-	-	- reserviert -
8 ... 31	PhoneNumber	STRING[22]		接続パートナーの呼び出し番号 許可されている値: プラスの文字(+)と番号 ネットワークプロバイダによって割り当てられた関連する電話番号の国際ダイヤルコードの正確な表記にご注意ください(「+」記号またはゼロ)。 PhoneNumber パラメータに入力がない場合、接続パラメータが指定されておらず、SMS メッセージはすべての許可された接続パートナーに受信される可能性があります。 スタートアップ中は次のことにご注意ください: 入力がない場合、TC_RECV は最初に一番先に受信した SMS メッセージを配送します。

遠隔制御サーバーまたはリモートステーションへの接続のシステムデータタイプ TCON_WDC

S7-1200 に割り当てられた遠隔制御サーバーまたは TCON_WDC を使用して遠隔制御サーバーを経由して到達できるリモートステーションへの接続を設定できます。 CP に割り当てられた遠隔制御サーバーのアドレスデータは、STEP 7 では CP の[遠隔制御インターフェース|モード]タブにあります。遠隔制御サーバーまたはリモートステーションはホスト名または IP アドレスを使用してアドレス指定されます。

TCON_WDC の「RemoteWdcAddress」パラメータは、接続パートナーのアクセス ID を指定します。

TCON_WDC のパラメータ

バイト	パラメータ	データタイプ	初期値	説明
0 ... 1	InterfaceID	HW_ANY		ローカル CP 1242-7 のインターフェースへの参照([STEP 7|CP コンフィグレーション|遠隔制御インターフェース|ハードウェア識別子]を参照)
2 ... 3	ID	CONN_OUC	1...07FF$_h$	GPRS 接続への参照。 割り当てられる ID は、CPU 内で一意となる必要があります。 ここで、TC_CON 命令の ID パラメータと同じ値を使用する必要があります。

4	ConnectionType	BYTE	W#16#0F	プロトコル変数 15 (F_h): IP アドレスを使用した遠隔制御接続
5	ActiveEstablished	BOOL		接続確立のタイプの識別子: • 0: パッシブ接続の確立 • 1: アクティブ接続の確立
6 ... 7	-	-	-	- 予約済み -
8 ... 11	RemoteWdcAd-dress	DWORD		アクセス ID を指定します(16 進数)。 アクセス ID は、接続パートナーによって異なります。 • リモート CP への接続: アクセス ID は以下の要素で構成されます。 ○ STEP 7 プロジェクト番号 ○ ステーション番号 ○ スロット リモートステーションに 2 つ以上の GPRS-CP が存在し、かつパスを指定しない場合、スロットの最後のバイトを 0 に設定する必要があります。 STEP 7 プロジェクトでのアクセス ID は、「CP の CPU 認証」パラメータグループ内にあります。 • 遠隔制御サーバーへの接続: アクセス ID = 0 • CP のプロセスイメージへの書き込みのみ場合: アクセス ID = DW#16#FEEDDADA

IF_CONF: 遠隔制御構成データの SDT

TC_CONFIG 命令のシステムデータタイプ IF_CONF の構造体

TC_CONFIG 命令の CONFIG パラメータは、変更される CP 1242-7 の設定データが格納されているメモリ領域を参照します。 データブロックに格納されている設定データは、システムデータタイプ (SDT) IF_CONF の構造体として記述されます。

IF_CONF はヘッダーと、それに続く STEP 7 プロジェクトのデバイスプロパティ内の CP のパラメータまたはパラメータ領域に対応したフィールドで構成されます。

変更される CP 設定データは、IF_CONF フィールドとしてまとめられます。 変更されないパラメータは、IF_CONF 構造体内では無視され、STEP 7 プロジェクトで設定されたままの状態で維持されます。

DB および IF_CONF 構造体の作成

CP のパラメータは、それぞれ 1 つ以上のフィールドがある 1 つ以上の構造体の IF_CONF DB 内に作成することができます。

フィールドのデータタイプは、キーボードを使用して入力する必要があります。 これらは選択リストには表示されません。 データタイプは大文字と小文字を区別しません。

以下の手順に従って、IF_CONF を作成します。

1. ブロックアクセスが「標準」のタイプ「グローバル DB」のデータブロックを作成します。
2. DB のパラメータ設定のテーブル内で、構造体(データタイプ"Struct")を作成します。

   任意の名前を指定できます。
3. この構造体で、ヘッダー名を割り当て、データタイプ"IF_CONF_Header"のセル内にそのヘッダー名を入力してヘッダーを追加します。

   構造体のヘッダー、および 3 つのパラメータ(下記参照)が作成されます。
4. 必要なデータタイプ(たとえば"IF_CONF_APN")をデータタイプのセルに入力して、変更する最初のパラメータのフィールドを作成します。
5. TC_CONFIG 命令を使用して CP 上で変更するすべてのパラメータについて、最後の手順を繰り返します。
6. 最後に、「subfieldCnt」パラメータのヘッダー内のフィールド数を更新します。

IF_CONF のヘッダー

IF_CONF_Header

バイト	パラメータ	データタイプ	初期値	説明
0 … 1	fieldType	UINT		フィールドタイプ: 常に 0 にする必要があります。
2 … 3	fieldId	UINT		Field ID: 常に 0 にする必要があります。
4 … 5	subfieldCnt	UINT		構造体に含まれたフィールドの合計数

パラメータフィールドの一般パラメータ

各フィールドには、以下の一般パラメータが含まれています。

- Id

  このパラメータはフィールドを識別しますが、変更してはいけません。

- Length

  このパラメータは、フィールドの長さを示します。 値は情報として使用します。

  文字列または配列、またはその両方を持つフィールドの長さは可変です。 非表示のバイトにより、フィールドの実際の長さは表示されるパラメータの合計よりも大きくなる場合があります。

- Mode

  これらのパラメータに許可されている値は次の通りです。

「モード」の値

値	意味
1	構成データが固定的に有効 CP 1242-7 には関連しません
2	既存の固定構成データの削除を含めて、構成データが一時的に有効 固定の設定データは、IF_CONF のパラメータフィールドによって置換されます。

パラメータ領域「GPRS アクセス」のフィールド

IF_CONF_APN

パラメータ	データタイプ	初期値	説明
Id	UINT	4	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 174
Mode	UINT		有効性(1: 固定的、2: 一時的)
AccesspointGPRS	STRING [98]		APN: GSM ネットワークプロバイダのインターネットへのアクセスポイント名
AccesspointUser	STRING [42]		APN ユーザー名
AccesspointPass-word	STRING [22]		APN パスワード

パラメータ領域「CP ID」のフィールド

IF_CONF_Login

パラメータ	データタイプ	初期値	説明
Id	UINT	5	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 54

Mode	UINT		有効性(1:固定的、2: 一時的)
ModemName	STRING [22]		アクセス ID 値を設定することはできません。
ModemPassword	STRING [22]		遠隔制御パスワード (最大 20 文字)

パラメータ領域「遠隔制御サーバーアクセスのフィールド

DNS によって解決可能な名前をによって遠隔制御サーバーをアドレス指定する時にのみ、このフィールドが使用されます。 遠隔制御サーバーが IP アドレスでアドレス指定される場合は、「IF_CONF_TCS_IP_V4」フィールドを使用します。

STEP 7 では、対応するデータは「モード」パラメータ領域に置かれます。

複数の遠隔制御サーバーがある場合、サーバーあたり 1 回フィールドを使用します。

IF_CONF_TCS_Name

パラメータ	データタイプ	初期値	説明
Id	UINT	6	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 266
Mode	UINT		有効性(1:固定的、2: 一時的)
TcsName	-	-	- 予約済み-
	STRING [254]		DNS によって解決可能な遠隔制御サーバーの名前
RemotePort	UINT		遠隔制御サーバーのポート
Rank	UINT		サーバーの優先度[1、2] 1 = 1 番目の遠隔制御サーバー 2 = 2 番目の遠隔制御サーバー

パラメータ領域「遠隔制御サーバーアクセスのフィールド

IP アドレスによって遠隔制御サーバーをアドレス指定する時にのみ、このフィールドを使用します。遠隔制御サーバーが DNS 名によってアドレス指定される場合は、「IF_CONF_TCS_Name」フィールドを使用します。

STEP 7 では、対応するデータは「モード」パラメータ領域に置かれます。

複数の遠隔制御サーバーがある場合、サーバーあたり 1 回フィールドを使用します。

IF_CONF_TCS_IP_v4

パラメータ	データタイプ	初期値	説明
Id	UINT	7	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 14
Mode	UINT		有効性(1:固定的、2: 一時的)
RemoteAddress	IP_V4		遠隔制御サーバーの IP アドレス

RemotePort	UINT		遠隔制御サーバーのポート
Rank	UINT		サーバーの優先度[1、2] 1 = 1 番目の遠隔制御サーバー 2 = 2 番目の遠隔制御サーバー

「モード」パラメータ領域のフィールド

STEP 7 では、対応するデータがパラメータ領域「モード」および「モデム設定」に置かれます。

IF_CONF_GPRS_Mode

パラメータ	データタイプ	初期値	説明
Id	UINT	8	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 10
Mode	UINT		有効性(1:固定的、2: 一時的)
GPRSmode	UINT		CP のモード: • 0 = Telecontrol • 1 = GPRS 直接
TemporaryStation	BOOL		ビット 0: 一時接続 このオプションが選択されている場合、CP はデータ送信のための一時接続のみを確立します。 フレームが転送されると、CP は再度接続を終了します。 • 1: 有効(一時接続) • 0: 無効(常時接続)
SMS_Enabled	BOOL		ビット 1: SMS が使用可能 このオプションを選択すると、S7 ステーションが SMS メッセージを送信できます。 • 1: 有効(SMS が使用可能) • 0: 無効(SMS なし)

「SMSC」パラメータのフィールド

STEP 7 では、対応するデータがパラメータ領域「モデム設定」に置かれます。

IF_CONF_SMS_Provider

パラメータ	データタイプ	初期値	説明
Id	UINT	10	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 28
Mode	UINT		有効性(1:固定的、2: 一時的)
SMSProvider	STRING [20]		GSM ネットワークプロバイダの SMS センター(SMSC)のノード番号であり、この番号によ

			ってこのステーションの接点が信号を送信しました。

「PIN」パラメータのフィールド

STEP 7 では、対応するデータがパラメータ領域「モデム設定」に置かれます。

IF_CONF_PIN

パラメータ	データタイプ	初期値	説明
Id	UINT	11	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 16
Mode	UINT		有効性(1:固定的、2: 一時的)
Pin	STRING [8]		PIN of the SIM カードの PIN が SIM カードに挿入されました。 PIN が正しく設定された場合、パラメータは関係ありません。 PIN が間違って設定された場合、正しい PIN を入力できます。

モニタ時間のフィールド

STEP 7 では、対応するデータがパラメータ領域「キープアライブタイムアウト」および「動作モード」に置かれます。

IF_CONF_TC_Timeouts

パラメータ	データタイプ	初期値	説明
Id	UINT	12	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 12
Mode	UINT		有効性(1:固定的、2: 一時的)
KeepAliveTimeout	-	-	- 予約済み - (設定不可)
SendTimeout	UINT		接続モニタ時間: 通信パートナーへの接続のモニタ時間(秒) モード「Telecontrol」および「GPRS 直接」に関連します
RedialTimeout	UINT		再接続遅延: 接続確立が不成功に終わった後、次の接続確立を試みるまでの待機時間の基本値。 3 回試みた後、基本値は最大 900 秒まで倍増。値の範囲: 10～600 秒です。 例: 基本値 20 を指定すると、次のダイヤル間隔になります。 3 回 20 秒、3 回 40 秒、 3 回 80 秒など、最大 900 秒まで。 2 番目の遠隔制御サーバー/ルータが設定済みの場合、CP は 4 回目の試行で 2 番目のパートナーへの接続を試行します。 2 番目のパートナーにも到達できない場合、CP は 7 回目に再度 1 番目のパートナーへの接続を試行します。

			SMS 接続には関連しません。

「ウェイクアップ権限」パラメータ領域のフィールド

IF_CONF_WakeupList

パラメータ	データタイプ	初期値	説明
Id	UINT	13	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 246
Mode	UINT		有効性(1:固定的、2: 一時的)
WakeupPhone [1...10]	ARRAY [1...10] of STRING [22]		ウェークアップが許可されている電話番号サブスクライバ 呼び出し番号の最後のアスタリスク(*)は、直接ダイヤル番号のプレースホルダに使用されます。

「推奨の GSM ネットワーク」パラメータ領域のフィールド

IF_CONF_PrefProvider

パラメータ	データタイプ	初期値	説明
Id	UINT	14	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 46
Mode	UINT		有効性(1:固定的、2: 一時的)
Provider [1...5]	ARRAY [1...5] of STRING [6]		CP がダイヤルする優先度 1～5 の選択可能な GSM ネットワーク。 最大 5 つのネットワークを設定できます。No. 1 が最高位優先度、No.5 が最下位優先度。 ネットワークプロバイダの Public Land Mobile Network (PLMN)のエントリは Mobile Country Code (MCC) and Mobile Network Code (MNC) で構成される。 例(Siemens AG のテストネットワーク): 26276

「DNS 構成」パラメータ領域のフィールド

IF_CONF_DNS

パラメータ	データタイプ	初期値	説明
Id	UINT	16	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 14
Mode	UINT		有効性(1:固定的、2: 一時的)

DNS_IP [1]	IP_V4		最初のドメイン名システムサーバーの IP アドレス
DNS_IP [2]	IP_V4		2 番目のドメイン名システムサーバーの IP アドレス

「時刻同期」パラメータ領域のフィールド

IF_CONF_NTP

パラメータ	データタイプ	初期値	説明
Id	UINT	17	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 24
Mode	UINT		有効性(1:固定的、2: 一時的)
NTP_IP [1]	ARRAY [1...4] of IP_V4		NTP サーバー 1 の IP アドレス
...	...		(NTP サーバー 2...3 の IP アドレス)
NTP_IP [4]	ARRAY [1...4] of IP_V4		NTP サーバー 4 の IP アドレス

TeleService ユーザーを有効/無効にするためのブロック

CP の STEP 7 プロジェクト内で既に設定されている TeleService ユーザーを有効または無効にするための SDT。STEP 7 内では、対応するデータはパラメータ領域[TeleService 設定|TeleService ユーザー管理]にあります。

IF_CONF_GPRS_UserList

パラメータ	データタイプ	初期値	説明
Id	UINT	19	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 506
Mode	UINT		有効性(1:固定的、2: 一時的)
GPRS_User [1...10]	ARRAY [1...10] of GPRS_User		TeleService ユーザー No.1 ~ 最大 No.10

この配列は、TeleService ユーザー用のパラメータレコードから生成されます("GPRS_User" [1...n])。

GPRS_User [n] (TeleService ユーザーのパラメータ)

パラメータ	データタイプ	初期値	説明
UserName [n]	STRING [22]		TeleService ユーザー名

Password [n]	STRING [22]		- 文字列は空に違いありません! -
Diag_Allowed [n]	BOOL		- 予約済み - (設定不可)
Teleserv_Allowed [n]	BOOL		TeleService ユーザーの有効化 • 0 = ユーザーは無効 • 1 = ユーザーは有効
FW_Load_Allowed [n]	BOOL		- 予約済み - (設定不可)

TeleService アクセス用のパラメータ設定のためのフィールド(サーバーの DNS 名)

TeleService サーバーのアクセスデータ(ステーションの切り替え)

STEP 7 では、対応するデータがパラメータ領域「TeleService 設定」に置かれます。

複数の TeleService サーバーがある場合、サーバーあたり 1 回フィールドを使用します。

IF_CONF_TS_Name

パラメータ	データタイプ	初期値	説明
Id	UINT	20	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 266
Mode	UINT		有効性(1:固定的、2: 一時的)
ts_name	String [254]		DNS によって解決可能な TeleService サーバーの名前
RemotePort	UINT		エンジニアリングステーションのポート
Rank	UINT		サーバーの優先度[1]または[2] 1 = サーバー 1、2 = サーバー 2

TeleService アクセス用のパラメータ設定のためのフィールド(サーバーの IP アドレス)

TeleService サーバーのアクセスデータ(ステーションの切り替え)

STEP 7 では、対応するデータがパラメータ領域「TeleService 設定」に置かれます。

複数の TeleService サーバーがある場合、サーバーあたり 1 回フィールドを使用します。

IF_CONF_TS_IF_V4

パラメータ	データタイプ	初期値	説明
Id	UINT	21	パラメータフィールドの ID
Length	UINT		バイト単位のパラメータフィールドの長さ: 14
Mode	UINT		有効性(1:固定的、2: 一時的)
RemoteAddress	IP_V4		TeleService サーバーの IP アドレス
RemotePort	UINT		TeleService サーバーのポート
Rank	UINT		サーバーの優先度[1]または[2]

			1 = サーバー 1、2 = サーバー 2

ポイントツーポイント

この章には下記に関する情報が記載されています：

フリーポート通信の概要

STEP 7 では、ユーザープログラムで指定されたプロトコルによるフリーポート通信で使用できる拡張命令が提供されます。 この命令は 2 つのカテゴリに分類できます。

- 設定命令
- 通信命令

注記

CPU の容量

フリーポート命令は、データレコードの読み出し/書き込みによって通信モジュールとの通信を行います。
そのため、命令を使用するときに、データレコードの読み出しと書き込みのための CPU の容量をチェックする必要があります。
複数の命令が CPU 上で同時にデータレコードの読み出し/書き込みを行う必要がある場合、ユーザープログラムで各命令の呼び出しの間に時間オフセットを作成する必要が生じる場合があります。

設定命令

ユーザープログラムがフリーポート通信を開始するには、その前に、通信インターフェースとデータの送受信用のパラメータを設定する必要があります。

インターフェースの設定とデータの設定は、デバイス構成で CM ごとに設定するか、ユーザープログラムの以下の命令で設定することができます。

- Port_Config
- Send_Config
- Receive_Config
- P3964_Config

通知

デバイス構成 <-> 設定命令

デバイス構成パラメータは、CPU の Power On (電圧の復旧)ごとに、CM に転送されます。

設定命令のパラメータは、ユーザープログラムでの定義に従って CM に転送されます。

デバイス構成のパラメータは、設定命令のパラメータと同期していません。すなわち、設定命令のパラメータは CPU デバイス構成には適用されません。
ユーザープログラムで、CM で適用するパラメータと、パラメータを適用する時点を決定します。

通信命令

ユーザープログラムは、フリーポート通信用の命令を使用して通信インターフェースとのデータの送受信を行います。 CM は、通信ステーションとのデータの送受信を行います。

- Send_P2P
- Receive_P2P

注記

データの整合性

- 送信するデータが整合性を保持して伝送される場合は、DONE が Send_P2P 命令によって設定されるまでは、REQ パラメータでの立ち上がり エッジ後にデータを変更することはできません。
- 受信データを整合性を保持して読み出す場合、受信データは、NDR = TRUE のサイクルでのみ評価できます。

受信バッファは追加命令でリセットすることができ、特殊な RS232 信号を照会して設定することが可能です。

- Receive_Reset
- Signal_Get
- Signal_Set

以下の命令がモジュールでサポートされている場合は、以下の命令を使用して拡張ファンクションを読み出しまたは書き込むことができます。

- Get_Features
- Set_Features

すべてのフリーポート命令は非同期で動作します。 このため、フリーポート命令は、DONE 出力パラメータによって実行の完了が示されるまで、呼び出す必要があります。

ユーザープログラムは、照会アーキテクチャを使用して送信および受信ステータスを判定できます。 Send_P2P および Receive_P2P は同時に実行可能です。 通信モジュールは、モジュール固有の最大バッファサイズに達するまで、要求に従って送信および受信データをバッファリングします。

注記

ビット時間単位の分解能

各パラメータに対して設定されたデータ伝送率で、ビット時間数を指定します。 パラメータをビット時間で指定すると、パラメータがデータ伝送率と無関係になります。 ビット時間単位のすべてのパラメータでは、最大 65535 までの数を指定できます。

命令の使用

フリーポート命令は、受信データ、または、送信プロセスの伝送の終了を照会するために、周期的に呼び出す必要があります。

データ量に応じて、データ伝送は複数の呼び出し(プログラムサイクル)にわたって行われる場合があります。
コマンドが DONE = TRUE または NDR = TRUE で完了する場合、コマンドはエラーなしで実行されたことになります。

注記

STATUS のバックアップ

DONE、NDR、ERROR、および STATUS パラメータは、1 つのブロックサイクルでのみ使用できます。 このため、STATUS を表示するには、それを空きのデータ領域にコピーする必要があります。

マスタ

マスタの代表的なシーケンス

1. Send_P2P 命令は、CM への伝送をトリガします。
   データ伝送は、REQ 入力での立ち上がりエッジで開始されます。
2. その後のサイクルで Send_P2P 命令が実行されて、伝送プロセスのステータスが照会されます。
3. Send_P2P 命令が、DONE 出力で、伝送が完了したことを通知すると、ユーザーコードは応答の受信準備を行うことができます。
4. Receive_P2P 命令が応答を照会するために繰り返し実行されます。 CM が応答データを取得すると、Receive_P2P 命令はその応答を CPU にコピーし、NDR 出力で新規データが受信されたことを通知します。
5. ユーザープログラムは応答を処理することができます。
6. ステップ 1 に戻って、シーケンスを繰り返します。

スレーブ

スレーブの代表的なシーケンス

1. ユーザープログラムが、サイクルごとに Receive_P2P 命令を実行します。
2. CM が要求を受信すると、Receive_P2P は、NDR 出力で新規データが使用できることを通知し、要求は CPU にコピーされます。
3. ユーザープログラムが要求を処理し、応答を作成します。
4. 応答が Send_P2P 命令でマスタに返されます。
5. 送信プロセスが実際に行われるようにするために、Send_P2P 命令を繰り返し実行する必要があります。
6. ステップ 1 に戻って、シーケンスを繰り返します。

スレーブは、応答待機中のタイムアウトのためにプロセスをキャンセルする前に、マスタが伝送を受信するのに十分な回数の Receive_P2P の呼び出しを行う必要があります。 そのため、タイムアウト設定が経過する前にマスタが伝送を受信できるような短いサイクルタイムのサイクル OB 内から、ユーザープログラム Receive_P2P を呼び出すことができます。 OB サイクルタイムが、マスタのタイムアウト設定内で 2 回の実行が行われるように設定されている場合、ユーザープログラムは一切の損失なしですべての伝送を受信することができます。

フリーポート操作の一般パラメータ

Freeport 命令の一般的な入力パラメータ

パラメータ	説明
REQ	データ伝送は、REQ 入力の立ち上がりエッジで開始されます。 コマンドが終了した後(DONE または ERROR)でないと、REQ の次のエッジは作成できません。 データ伝送は、データの量によって数回の呼び出し(プログラムサイクル)がかかることがあります。 STEP 7 では、ユーザープログラムに Freeport 命令を追加すると、インスタンス DB の指定(STEP 7 による対応インスタンス DB の作成)を促すプロンプトが表示されます。 PtP 命令の呼び出しごとに、一意の DB を使用します。
PORT	ポートアドレスは、通信モジュールの構成時に割り当てられます。 PORT パラメータは、特定の通信モジュールへの割り当てを命令に通知します。 構成後に、標準ポートのシンボル名を選択できます。 割り当てられた CM ポート値は、S7-1200/1500 の CM ポート値はデバイス構成の[ハードウェア ID]プロパティ、または S7-300/400 の[入力アドレス]です。ポートのシンボル名は、シンボルテーブルで割り当てられます。

Freeport 命令の出力パラメータ DONE、NDR、ERROR、および STATUS は、Freeport ファンクションの実行ステータスを示します。

出力パラメータ DONE、NDR、ERROR、および STATUS

パラメータ	データタイプ	標準	説明
DONE	Bool	FALSE	最後の要求がエラーで終了したことを示すために、1 サイクルの間 TRUE に設定されます。それ以外の場合は、FALSE に設定されます。
NDR	Bool	FALSE	新しいデータが受信されたことを示すために、1 サイクルの間 TRUE に設定されます。それ以外の場合は、FALSE に設定されます。
ERROR	Bool	FALSE	最後の要求がエラーで終了したことを示すために、1 サイクルの間 TRUE に設定されます。対応するエラーコードは、STATUS にあります。それ以外の場合は、FALSE に設定されます。
STATUS	Word	16#0000 または または 16#7000	結果ステータス • DONE または NDR ビットが設定されている場合、STATUS が 0/16#7000 または特定のエラーコードに設定されます。 • ERROR ビットが設定されている場合、STATUS にエラーコードが表示されます。 • 上記のビットがどれも設定されていない場合、命令はファンクションの現在のステータスを説明するステータス結果を返します。 STATUS の値は、この命令を(同一のポートアドレスを使用して)再度呼び出すまで有効です。

COM_RST 入力/出力パラメータ

パラメータ	データタイプ	標準	説明
COM_RST	Bool	FALSE	命令の初期化 命令は TRUE に初期化されます。 次に、COM_RST の設定は FALSE に戻ります 注記: スタートアップ中に COM_RST を TRUE に設定する必要があり、その後にパラメータを変更することはできません(つまり、命令を呼び出す時に値を割り当てません)。 COM_RST は、インスタンス DB の初期化後の命令によってリセットされます。

注記
パラメータ DONE、NDR、ERROR、および STATUS は 1 サイクルに対してのみ設定されます。

共有エラーコード

エラーコード	説明
16#0000	エラーはありません。
16#7000	ファンクションがアクティブではありません
16#7001	要求後に初期呼び出しが開始されました。
16#7002	要求後に後続呼び出しが開始されました。
16#8x3A	パラメータ x のポインタが無効

STATUS パラメータの共有エラークラス

クラスの説明	エラークラス	説明
ポートの構成	16#81Ax	インターフェースの構成で頻繁に発生するエラーの説明
送信構成	16#81Bx	送信構成のエラーの説明
受信構成	16#81Cx	受信構成のエラーの説明
送信	16#81Dx	送信中のランタイムエラーの説明
受信	16#81Ex	受信中のランタイムエラーの説明
RS232 の付随信号	16#81Fx	信号処理との接続のエラーの説明

Port_Config: PtP 通信ポートを設定

説明

Port_Config 命令(ポートの構成)使用して、データ伝送率などのパラメータをユーザープログラムでランタイム時に変更することができます。 CM 内のデータ保留は Port_Config の実行で削除されます。

Port_Config の構成変更は CM に保存され、CPU には保存されません。 電圧が復旧すると、CM はデバイス構成に保存されたデータで構成定義されます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	この入力での立ち上がりエッジ時に CM へのデータ伝送が開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは､HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
PROTO-COL	IN	UInt	Word	0	プロトコル • 0 = Freeport プロトコル • 1 = プロトコル 3964(R)
BAUD	IN	UInt	Word	6	ポートのデータ伝送率 • 1 = 300 Bit/s • 2 = 600 Bit/s • 3 = 1200 Bit/s • 4 = 2400 Bit/s • 5 = 4800 Bit/s • 6 = 9600 Bit/s • 7 = 19200 Bit/s • 8 = 38400 Bit/s • 9 = 57600 Bit/s • 10 = 76800 Bit/s • 11 = 115200 Bit/s
PARITY	IN	UInt	Word	1	ポートのパリティ: • 1 = パリティなし

					• 2 = 偶数パリティ • 3 = 奇数パリティ • 4 = マークパリティ • 5 = スペースパリティ • 6 = 任意
DATA-BITS	IN	UInt	Word	1	文字当たりのビット数: • 1 = 8 データビット • 2 = 7 データビット
STOP-BITS	IN	UInt	Word	1	ストップビット: • 1 = 1 ストップビット • 2 = 2 ストップビット
FLOWCTRL	IN	UInt	Word	1	フロー制御 • 1 = フロー制御なし • 2 = XON/XOFF • 3 = ハードウェア RTS は常に ON • 4 = ハードウェア RTS 切り替え済み • 5 = ハードウェア RTS は常に ON、DTR/DSR を無視
XON-CHAR	IN	Char		16#0011	XON 文字として機能する文字が指定されます。 これは、通常、DC1 文字です(11H)。 このパラメータはソフトウェアフロー制御が有効な場合のみ評価されます。
XOFF-CHAR	IN	Char		16#0013	XOFF 文字として機能する文字が指定されます。 これは、通常、DC3 文字です(13H)。 このパラメータはソフトウェアフロー制御が有効な場合のみ評価されます。
WAI-TIME	IN	UInt	Word	2000	XON 文字の受信後の XOFF 文字の待機時間、または CTS = OFF 後の CTS = ON 信号の待機時間が指定されます(0～65535 ms)。 このパラメータはフロー制御が有効の場合にのみ評価されます。
MODE	IN	USInt	Byte	0	動作モード 有効な動作モード以下のとおりです。 • 0 = 全二重(RS232) • 1 = 全二重(RS422) 4 線式モード(ポイントツーポイント) • 2 = 全二重(RS 422) 4 線式モード(マルチポイントマスタ; CM PtP (ET 200SP)) • 3 = 全二重(RS 422) 4 線式モード(マルチポイントスレーブ; CM PtP (ET 200SP)) • 4 = 半二重(RS485) 2 線式モード [1)]
LINE_PRE	IN	USInt	Byte	0	回線初期状態の受信 有効な初期状態は以下のとおりです。 • 0 = 初期状態「なし」 [1)]

					• 1 = シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) この初期状態では、ブレーク検出が可能です。 以下とともにのみ選択できます。「全二重(RS422) 4 線式モード(ポイントツーポイント接続)」および「全二重(RS422) 4 線式モード(マルチポイントスレーブ)」 • 2 = シグナル R(A) 0 V/シグナル R(B) 5 V この初期設定はアイドル状態に相当します(送信動作がアクティブでない)。 この初期状態では、ブレーク検出は不可能です。
CA-BLE_BRK	IN	USInt	Byte	0	ブレーク検出 以下の設定が有効です。 • 0 = ブレーク検出が無効 • 1 = ブレーク検出が有効
COM_RST	IN/OUT	---	Bool	FALSE	Port_Config 命令の初期化 命令は 1 で初期化されます。その後、命令は COM_RST を 0 にリセットします。
DONE	OUT	Bool		FALSE	最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE
ERROR	OUT	Bool		FALSE	最後の要求がエラーありで完了した後、1 サイクルの間 TRUE
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)
[1] RS485 の CM 1241 で PROFIBUS ケーブルを使用するために必要な設定。					

Send_Config PtP 送信元を設定

説明

Send_Config 命令(送信構成)を使用して、ユーザープログラムでランタイム中に送信パラメータ(送信するデータの開始と終了を識別する条件)を変更できます。 Send_Config を実行すると、CM 内のすべての保留中のデータが削除されます。

Send_Config の構成の変更は CM に保存され、CPU には保存されません。 デバイス構成に保存されたパラメータは、電圧が CPU または通信モジュールに復旧すると、復元されます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	この入力の立ち上がりエッジで構成の変更を有効にします。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
RTSONDLY	IN	UInt	Word	0	RTS の有効化から送信データの伝送の開始までの待機時間(ミリ秒)。 このパラメータはハードウェアフロー制御が有効の場合にのみ有効です。 有効な範囲は 0 ~ 65535 ms です。値 0 はファンクションを無効にします。
RTSOFFDLY	IN	UInt	Word	0	送信データの伝送から RTS の無効化までの待機時間(ミリ秒)。 このパラメータはハードウェアフロー制御が有効の場合にのみ有効です。 有効な範囲は 0～65535 ms です。値 0 はファンクションを無効にします。
BREAK	IN	UInt	Word	0	このパラメータは、BREAK が各フレームの開始時に、指定されたビット時間数の間、送信されることを指定します。 最大値は 65535 ビット時間です。値 0 はファンクションを無効にします。
IDLE-LINE	IN	UInt	Word	0	このパラメータは、ラインが各フレームの開始前に、指定されたビット時間数の間、アイドルになっていることを指定します。 最大値は 65535 ビット時間です。 値 0 はファンクションを無効にします。
USR_END	IN	STRING[2]		0	エンドデリミタの入力 2 つ以上のエンドデリミタを構成定義することはできません。

				エンドデリミタを含むすべてのデータが、構成定義されたフレーム長とは無関係に送信されます。
APP_END	IN	STRING[5]	0	追加する文字の入力。 最大 5 文字まで追加できます。
COM_RST	IN/OUT	--- Bool	FALSE	Send_Config 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DONE	OUT	Bool	FALSE	最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE
ERROR	OUT	Bool	FALSE	最後の要求がエラーありで完了した後、1 サイクルの間 TRUE
STATUS	OUT	Word	16#7000	エラーコード(エラーメッセージを参照)

Receive_Config: PtP 受信先を設定

説明

Receive_Config 命令(受信構成)を使用して、ユーザープログラムでランタイム中に受信パラメータを変更できます。 この命令は、受信データの開始と終了をマークする条件を構成定義します。 Receive_Config を実行すると、CM 内のすべての保留中のデータが削除されます。

Port_Config の構成の変更は CM に保存され、CPU には保存されません。 デバイス構成に保存されたパラメータは、電圧が CPU または通信モジュールに復旧すると、復元されます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400 / WinAC		
REQ	IN	Bool		FALSE	この入力の立ち上がりエッジで構成の変更を有効にします。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
Receive_Conditions	IN	Variant	Any	-	Receive_Conditions のデータ構造体は、フレームの開始と終了を識別するのに使用する開始および終了条件を指定します。
COM_RST	IN/OUT	---	Bool	FALSE	Receive_Config 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DONE	OUT	Bool		FALSE	最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE
ERROR	OUT	Bool		FALSE	最後の要求がエラーありで完了した後、1 サイクルの間 TRUE
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)

Receive_P2P 命令の開始条件

Receive_P2P 命令は、デバイス構成または Receive_Config 命令で指定された構成を使用して、Freeport 通信フレームの開始と終了を識別します。 フレームの開始は開始条件で定義されます。 フレームの開始は、1 つまたは複数の開始条件を使用して識別できます。

ブレークとアイドルラインが有効になると、最初にブレークを満たし、その後にアイドルラインも満たす必要があります。 その後、他の条件の 1 つ(開始文字または開始シーケンス)を満たすと、データ伝送が開始されます。
開始条件「任意の文字」は、他の開始条件と組み合わせることはできません。

Receive_Conditions パラメータのデータタイプ構造、パート 1 (開始条件)

Receive_Conditions の開始条件の構造

パラメータ	宣言	データタイプ	既定	説明
START.STARTCOND	IN	Word	16#0002	開始条件の指定 • 01H - 開始文字の検出 • 02H - 任意の文字 • 04H - 改行の検出 • 08H - アイドルラインの検出 • 10H - 開始シーケンス 1 の検出 • 20H - 開始シーケンス 2 の検出 • 40H - 開始シーケンス 3 の検出 • 80H - 開始シーケンス 4 の検出 開始条件は、値の追加によって組み合わせることができます。
START.IDLETIME	IN	Word	16#0028	アイドル状態で新しいフレームの開始が検出されるのに必要なビット時間数(既定値: W#16#28)。条件「アイドルラインの検出」での接続時のみ。 0 ~ FFFF
START.STARTCHAR	IN	Byte	16#0002	条件「開始文字」の開始文字。 (既定: B#16#2)
START.SEQ[1].CTL	IN	Byte	0	開始シーケンス 1、各文字ごとの比較を無効化/有効化: (既定: B#16#0) 以下は、開始文字列の文字ごとの有効化ビットです。 • 01H - 文字 1 • 02H - 文字 2 • 04H - 文字 3 • 08H - 文字 4 • 10H - 文字 5 ビットが特定の文字に対して無効になっている場合、文字列内の当該位置の各文字は有効な開始文字列を表しています(たとえば、1FH = 解釈済みのすべての 5 文字)。
START.SEQ[1].STR[1] .. START.SEQ[1].STR.[5]	IN	Char[5]	0	開始シーケンス 1、開始文字(5 文字)

START.SEQ[2].CTL	IN	Byte	0	開始シーケンス 2、各文字ごとの比較を無効化/有効化。 既定値: B#16#0)
START.SEQ[2].STR[1] .. START.SEQ[2].STR.[5]	IN	Char[5]	0	開始シーケンス 2、開始文字(5 文字)
START.SEQ[3].CTL	IN	Byte	0	開始シーケンス 3、各文字ごとの比較を無効化/有効化。既定値: B#16#0
START.SEQ[3].STR[1] .. START.SEQ[3].STR.[5]	IN	Char[5]	0	開始シーケンス 3、開始文字(5 文字)
START.SEQ[4].CTL	IN	Byte	0	開始シーケンス 4、各文字ごとの比較を無効化/有効化。既定値: B#16#0
START.SEQ[4].STR[1] .. START.SEQ[4].STR.[5]	IN	Char[5]	0	開始シーケンス 4、開始文字(5 文字)

例

次のような 16 進コードの受信データを見てみましょう: "「68 10 aa 68 bb 10 aa 16」。 構成定義済みの開始文字列を以下の表に示します。 最初の文字 68H が正常に受信されると、開始文字列が評価されます。 4 番目の文字が正常に受信されると(2 つ目の 68H)、開始条件 1 が満たされます。 開始条件が満たされると、終了条件の評価が開始されます。

開始文字列の処理は、パリティ、フレーミング、文字間の時間間隔のさまざまな工ラーによってキャンセルされることがあります。 これらの工ラーにより、開始条件が満たされないため、データの受信が阻止されます(工ラーメッセージが出力されます)。

開始条件

開始条件	最初の文字	最初の文字+1	最初の文字+2	最初の文字+3	最初の文字+4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

Receive_P2P 命令の終了条件

フレームの終了は、構成定義された 1 つ以上の終了条件の最初の発生によって定義されます。

終了条件は、デバイス構成の通信インターフェースのプロパティまたは Receive_Config 命令のいずれかで構成定義できます。 受信パラメータ(開始および終了条件)は、電圧が CPU または通信モジュールに復旧するたびにデバイス構成の設定にリセットされます。 STEP 7 ユーザープログラムが Receive_Config を実行すると、設定は Receive_Config のパラメータにセットされます。

Receive_Conditions パラメータのデータタイプ構造、パート 2 (終了条件)

Receive_Conditions の終了条件の構造

パラメータ	宣言	データタイプ	既定	説明
END.END-COND	IN	Word	0	このパラメータはフレーム終了の条件を指定します。 • 01H - 応答タイムアウト • 02H - メッセージタイムアウト • 04H - キャラクタ遅延時間 • 08H - 最大フレーム長 • 10H - メッセージからのメッセージ長の読み出し(N+LEN+M) • 20H - 終了シーケンス • 40H - 固定フレーム長
END.FIXLEN	IN	Word	1	固定フレーム長: 終了条件[固定フレーム長]が選択されている場合にのみ使用されます。 1～4000 バイト(モジュールに応じて最大 4 KB)
END.MAX-LEN	IN	Word	1	最大フレーム長: 終了条件[最大フレーム長]が選択されている場合にのみ使用されます。 1～4000 バイト(モジュールに応じて最大 4 KB)
END.N	IN	Word	0	フレームの長さフィールドのバイト位置。 終了条件 N+LEN+M でのみ使用。 1～4000 バイト(モジュールに応じて最大 4 KB)
END.LENGTHSIZE	IN	Word	0	長さフィールドのサイズ(1、2、または 4 バイト)。終了条件 N+LEN+M でのみ使用。
END.LENGTHM	IN	Word	0	長さフィールドの値に含まれていない、長さフィールドの後の文字数 このエントリは、終了条件 N+LEN+M でのみ使用されます。0～255 バイト
END.RCVTIME	IN	Word	200	フレームが送信された後の最初の受信文字の待機時間を指定します。 文字が指定した時間内に受信されない場合、この受信命令はエラーメッセージ付きで終了します。 この情報は条件「応答タイムアウト」でのみ使用されます。 (0～65535 ms)。 注記: このパラメータは単独の終了条件として使用することはできず、最低でも 1 つの他の終了条件と組み合わせてのみ使用可能です。
END.MSGTIME	IN	Word	200	最初の文字を受信してからフレーム全体を受信するまでの待機時間を指定します。 このパラメータは、条件「メッセージタイムアウト」を選択した場合のみ使用されます。 (0～65535 ms)
END.CHAR-GAP	IN	Word	12	文字間の最大ビット時間数を入力します。 文字間のビット時間数が指定された値を超えている場合、終了条件は満たされていません。 この情報は条件「キャラクタ間遅延時間」でのみ使用されます。 (0～65535 ビット時間)
END.SEQ.CTL	IN	Byte	0	文字シーケンス 1、各文字ごとの比較を無効化/有効化: 以下は、開始文字列の文字ごとの有効化ビットです。 文字 1 はビット 0、文字 2 はビット 1、...、文字 5 はビット 4 です。ビットが特定の文字に対し

				て無効になっている場合、各文字は文字列の当該位置が適合していることを表しています。
END.SEQ.STR[1] .. END.SEQ.STR[5]	IN	Char[5]	0	文字列 1、開始文字(5 文字)

Receive_P2P 命令の一般パラメータ

パラメータ	宣言	データタイプ	既定	説明
GENERAL.MBUF_SIZE	IN	Byte	255	CM の受信バッファにバッファリングされる入力フレーム数。 受信バッファの応答に影響する他の条件(タイムアウトの阻止、データフロー制御)が有効でない場合は、この限界値に達すると、追加のフレームは破棄されます。 (1～255 フレーム)
GENERAL.OW_PROT	IN	Byte	0	CM が新しいフレームを受信し、CM の受信バッファがまだ読み出されていない場合に、バッファリングされたフレームの上書き防止ファンクションをアクティブにします。 このステップによって、バッファ済みの受信フレームが失われるのを防ぎます。 • 0 - 有効になっていません • 1 - 有効になっています
GENERAL.CLR_MBUF	IN	Byte	0	CPU スタートアップ中の受信バッファの削除を有効にします。 受信バッファは、CPU が STOP から RUN に切り替わると自動的に削除されます。 受信バッファには、CPU のスタートアップ後に受信されたフレームのみが格納されています。 • 0 - 有効になっていません • 1 - 有効になっています

P3964_Config: 3964(R)プロトコルの設定

説明

P3964_Config 命令(プロトコルの構成)を使用して、キャラクタ間遅延時間、優先度、およびブロックチェックなどの 3964(R)用プロトコルパラメータをユーザープログラムでランタイム中に変更することができます。

P3964_Config の構成の変更は CM に保存され、CPU には保存されません。 デバイス構成に保存されたパラメータは、電圧が CPU または通信モジュールに復旧すると、復元されます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	この入力での立ち上がりエッジ時に命令が開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
BCC	IN	SInt	Byte	1	ブロックチェックの使用の有効化/無効化 • 0 = ブロックチェックなし • 1 = ブロックチェックあり
Priority	IN	SInt	Byte	1	優先度の選択 • 0 = 低い優先度 • 1 = 高い優先度
Character-DelayTime	IN	UInt	Word	16#00DC	キャラクタ間遅延時間の設定(設定されたデータ伝送速度に応じて異なります) (既定値: 220 ミリ秒) 1 ~ 65535 ms
AcknDelay-Time	IN	UInt	Word	16#07D0	確認遅延時間の設定(設定されたデータ伝送速度に応じて異なります) (既定値: 2000 ミリ秒) 1 ~ 65535 ms
BuildupAt-tempts	IN	SInt	Byte	16#0006	接続試行回数の設定(既定値: 6 回の接続試行回数) 1 ~ 255

RepetitionAt-tempts	IN	SInt	Byte	16#0006	伝送試行回数の設定(既定値: 6 回の接続試行回数) 1 ~ 255
COM_RST	IN/OUT	---	Bool	FALSE	P3964_Config 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DONE	OUT	Bool		FALSE	最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE
ERROR	OUT	Bool		FALSE	最後の要求がエラーありで完了した後、1 サイクルの間 TRUE
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)

Send_P2P:データ送信

説明

Send_P2P 命令(ポイントツーポイントデータの送信)はデータの伝送を開始し、割り当てられたバッファの内容を通信モジュールに伝送します。 CM が指定されたデータ伝送速度でデータを送信している間、CPU プログラムがまだ実行中です。 1 回に保留できる送信命令は通信モジュール当たり 1 つのみです。 CM が既にフレームを送信中に 2 番目の Send_P2P 命令が実行されると、CM はエラーを通知します。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	この入力での立ち上がり工ッジ時に CM へのデータ伝送が開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
BUF-FER	IN	Variant	Any	0	このパラメータは、送信バッファのメモリ領域を指します。 **注記:** • Boolean データと Boolean フィールドはサポートされていません。 • 送信バッファが最適化されたメモリ領域内に存在する場合、送信データの最大許容長は 1024 バイトです。 例外: Arrays of Byte、Word、または DWord の場合、最大 4096 バイトの長さまでサポートされます。 • 送信バッファが String または WString の場合、文字列の内容が現在の長さおよび最大長のフィールドなしで転送されます。
LENGTH	IN	UInt	Word	0	転送されるデータの長さ(バイト単位) BUFFER パラメータでアドレス指定されたメモリ領域は、LENGTH = 0 で完全に伝送されます。
COM_RST	IN/OUT	---	Bool	FALSE	Send_P2P 命令の初期化 命令は 1 で初期化されます。その後、命令は COM_RST を 0 にリセットします。

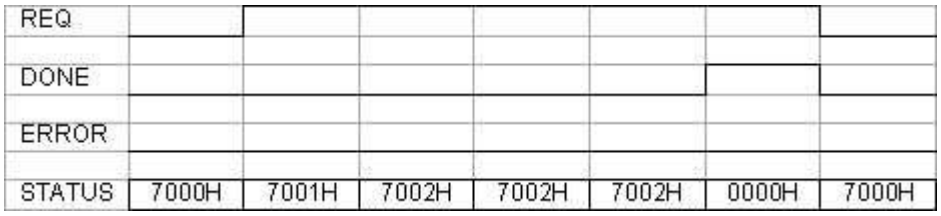
DONE	OUT	Bool	FALSE	最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE
ERROR	OUT	Bool	FALSE	最後の要求がエラーありで完了した後、1 サイクルの間 TRUE
STATUS	OUT	Word	16#7000	エラーコード(エラーメッセージを参照)

パラメータ

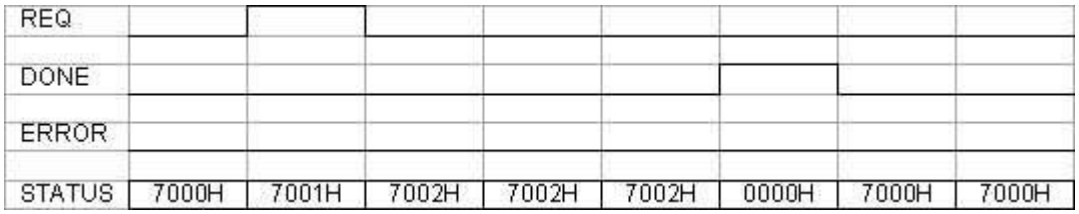
送信命令が処理中の場合、DONE および ERROR 出力は FALSE ステータスです。 送信命令の終了時に、DONE または ERROR 出力の 1 つが、送信命令のステータスを通知するために、1 サイクルの間 TRUE に設定されます。 STATUS 出力のエラーコードは、ERROR のステータスが TRUE の場合に評価することができます。

この命令は、通信インターフェースが送信データを承認すると、ステータス 16#7001 を出力します。Send_P2P のそれ以後の実行では、CM がまだ送信中の場合、値 16#7002 が出力されます。 送信命令の終了時に、CM は送信命令のステータス 16#0000 を出力します(エラーが発生しなかった場合)。 それ以後の REQ = 0 の Send_P2P の実行では、ステータス 16#7000 (空き)が出力されます。

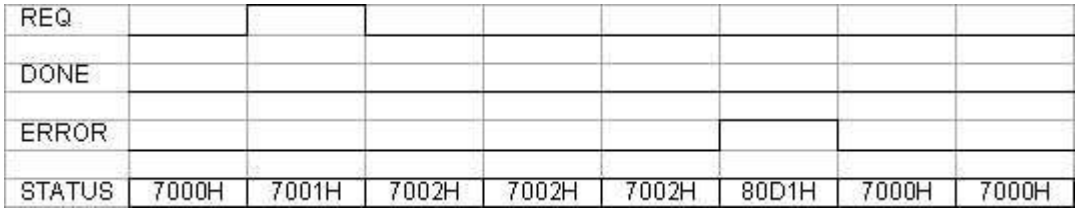
下の図に、出力値と REQ との間の関係を示します。 これは、送信プロセスのステータス(STATUS 値で示される)をチェックするために命令が周期的に呼び出されることを前提にしています。



下の図に、送信命令をトリガするパルスが REQ ラインで保留中になっている場合に(1 サイクルの間)、DONE および STATUS パラメータが 1 サイクルの間のみ有効になる様子を示します。



下の図に、エラー発生時の、DONE、ERROR、および STATUS パラメータの関係を示します。



DONE、ERROR、および STATUS の値は、Send_P2P が同一インスタンス DB で再び実行されると無効になります。

通信操作での LENGTH および BUFFER パラメータの使用

Send_P2P の LENGTH パラメータと BUFFER パラメータの相互関係

「Send_P2P」命令によって送信できる最小データサイズは、1 バイトです。 呼び出し中に LENGTH パラメータで「0」が渡される場合、BUFFER パラメータに送信するデータのサイズを指定します。 データタイプ Bool または Bool タイプの配列を BUFFER パラメータで使用することはできません。

LENGTH パラメータ

LENGTH	説明
> 0	設定されたバイト数を転送します。
= 0	BUFFER でアドレス指定された送信バッファの内容全体を転送します。 BUFFER が文字列を指す場合、最大長および現在の長さのバイトを除く文字列の内容全体が転送されます。

BUFFER パラメータ

BUFFER	説明
基本データタイプ	送信時: LENGTH 値には、このデータタイプのバイトサイズを含める必要があります。 例: Word 値の場合、LENGTH は 2 です。 DWord 値または Real 値の場合、LENGTH は 4 です。
構造体	最適化されたメモリの場合: BUFFER の最大許容長は 1024 Byte です; その他の場合、モジュールに応じて 4 KB が許可されます。 送信の場合: LENGTH 値には、構造体の全体のバイト長よりも小さいバイトサイズを格納できます。この場合、BUFFER の構造体の最初の LENGTH バイトのみが送信されます。
Array	最適化されたメモリの場合: 配列のデータタイプが Byte、Word、または DWord と等しくない場合、最大許容バッファ長は 1024Byte. バイトです。モジュールに応じて異なりますが、メモリが最適化されていない場合は、データ構造体には関わりなく、最大 4 KB を送信できます。 送信の場合: LENGTH 値には、配列の全体のバイト長よりも小さいバイトサイズを格納できます。このため、このバイトサイズはデータエレメントのバイトサイズの倍数になります。例: Word タイプの配列の LENGTH パラメータは 2 の倍数、Real タイプの配列の場合は 4 の倍数とする必要があります。 LENGTH を指定すると、そのバイト数に対応する数の配列エレメントが送信されます。 たとえば、BUFFER に 15 個の DWord エレメント(合計で 60 Bytes)の配列を格納し、LENGTH = 20 を指定すると、配列の最初の 5 つの DWord エレメントが送信されます。
String	LENGTH パラメータには、送信する文字数を格納します。 String の文字のみが転送されます。 String の現在の最大長のバイトは送信されません。

Receive_P2P:データ受信

説明

Receive_P2P 命令(ポイントツーポイント通信によるデータ受信)は、CM に受信されたフレームをチェックします。フレームが使用可能な場合、それを CM から CPU に送信します。 受信エラーは STATUS パラメータで示されます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400 / WinAC		
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
BUF-FER	IN	Variant	Any	0	このパラメータは、受信バッファの開始アドレスをポイントします。 このバッファは、最大フレーム長を受信するのに十分な大きさである必要があります。 **注記:** • Boolean データまたは Boolean フィールドはサポートされていません。 • 受信バッファが最適化されたメモリ領域内に存在する場合、受信データの最大許容長は 1024 バイトです。 例外: Arrays of Byte、Word、または DWord の場合、最大 4096 バイトの長さまでサポートされます。 • 受信バッファが String または WString の場合は、受信データが文字列の内容に書き込まれ、文字列の現在の長さがそれに応じて設定されます。
COM_RST	IN/OUT	---	Bool	FALSE	Receive_P2P 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
LENGTH	OUT	UInt	Word	0	受信されたフレームの長さ(バイト単位)
NDR	OUT	Bool		FALSE	新しいデータが使用可能で、命令がエラーなしで完了した場合、1 サイクルの間 TRUE。

ERROR	OUT	Bool	FALSE	命令がエラーで完了した場合、1 サイクルの間 TRUE。
STATUS	OUT	Word	16#7000	エラーコード(エラーメッセージを参照)

STATUS 出力のエラーコードは、ERROR のステータスが TRUE の場合に評価することができます。
STATUS 値は、CM で受信動作を終了する理由を提供します。
この値は、通常、受信動作が成功したことと、検出されたフレーム条件を示す正の値です。
STATUS 値が負の場合(16 進数値の最上位ビットが設定されている)、受信動作はパリティエラー、フレームエラー、またはオーバーフローエラーなどのエラー条件によって終了しました。

各通信モジュールは、モジュール固有の数のフレームをバッファリングすることができます。 CM で複数のフレームが使用可能な場合、Receive_P2P 命令は最も古い使用可能なフレームを出力します (FIFO)。

Receive_Reset: 受信バッファのクリア

説明

Receive_Reset 命令(受信元のリセット)は CM の受信バッファをクリアします。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400 / WinAC		
REQ	IN	Bool		FALSE	この入力での立ち上がりエッジ時に CM へのデータ伝送が開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
COM_RST	IN/OUT	---	Bool	FALSE	Receive_Reset 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DONE	OUT	Bool		FALSE	1 サイクルの間の TRUE は、最後の要求がエラーなしで完了したことを示します。
ERROR	OUT	Bool		FALSE	TRUE は、最後の要求がエラーで完了したことを示します。 この出力が TRUE の場合、STATUS 出力には対応するエラーコードが含まれています。
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)

Signal_Get: ステータスの読み出し

説明

Signal_Get 命令(RS232 信号の取得)は、RS232 に付随する信号の現在の状態を読み出し、対応する命令出力にそれを表示します。

注記

制約

このファンクションは、CMs RS232 BA および RS232 HF の場合のみ使用できます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400 / WinAC		
REQ	IN	Bool		FALSE	この入力での立ち上がりエッジ時に CM へのデータ伝送が開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
COM_RST	IN/OUT	---	Bool	FALSE	Signal_Get 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
NDR	OUT	Bool		FALSE	RS232 付随信号が読み出され、命令がエラーなしで完了した場合、1 サイクルの間 TRUE。
ERROR	OUT	Bool		FALSE	命令がエラーで完了した場合、1 サイクルの間 TRUE
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)
DTR	OUT	Bool		FALSE	データデバイス準備完了、モジュール準備完了(出力)
DSR	OUT	Bool		FALSE	データデバイス準備完了、通信ステーション準備完了(入力)
RTS	OUT	Bool		FALSE	送信要求、モジュールの送信準備完了(出力)
CTS	OUT	Bool		FALSE	送信準備完了、通信ステーションがデータを受信できます(入力)
DCD	OUT	Bool		FALSE	データキャリア信号検出済み、信号レベル受信済み

RING	OUT	Bool	FALSE	呼び出し表示、着信呼び出しの通知

Signal_Set: 付属信号の設定

説明

Signal_Set 命令(RS232 信号の設定)を使用して、RS232 通信信号を設定することができます。

注記

制約事項

このファンクションは、CMs RS232 BA および RS232 HF の場合のみ使用できます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/ WinAC		
REQ	IN	Bool		FALSE	この入力の立ち上がりエッジ時に命令が開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
SIGNAL	IN	Byte		0	設定する信号の選択(複数の選択が可能) • 01H = RTS • 02H = DTR • 04H = DSR (インターフェースタイプ DCE の場合のみ)
RTS	IN	Bool		FALSE	送信要求、モジュールの送信準備完了 出力にこの値を設定します(TRUE または FALSE)、既定値: FALSE
DTR	IN	Bool		FALSE	データ端末準備完了、モジュール準備完了 出力にこの値を設定します(TRUE または FALSE)、既定値: FALSE
DSR	IN	Bool		FALSE	データ端子準備完了(DCE インターフェースタイプの場合のみ)、未使用。
COM_RST	IN/OUT	---	Bool	FALSE	Signal_Set 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。

DONE	OUT	Bool	FALSE	最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE
ERROR	OUT	Bool	FALSE	最後の要求がエラーありで完了した後、1 サイクルの間 TRUE
STATUS	OUT	Word	16#7000	エラーコード(エラーメッセージを参照)

Get_Features: 拡張ファンクションを取得

説明

モジュールでサポートされている場合は、Get_Features 命令(拡張ファンクションを取得)を使用して、CRC をサポートし診断メッセージを生成するためのモジュールの機能についての情報を取得できます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	この入力の立ち上がりエッジ時に命令が開始されます。
PORT	IN	PORT	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
COM_RST	IN/OUT	---	Bool	FALSE	Get_Features 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
NDR	OUT	Bool		FALSE	新しいデータが使用可能で、命令がエラーなしで完了した場合、1 サイクルの間 TRUE
MOD-BUS_CRC	OUT	Bool		FALSE	Modbus CRC のサポート
DI-AG_ALARM	OUT	Bool		FALSE	診断メッセージの生成
SUP-PLY_VOLT	OUT	Bool		FALSE	電源電圧 L+が不明の診断が使用できます
ERROR	OUT	Bool		FALSE	命令がエラーで完了した場合、1 サイクルの間 TRUE
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)

Set_Features: 拡張ファンクションの設定

説明

モジュールでサポートされている場合、Set_Features 命令(拡張ファンクションの選択)を使用して、CRC サポートと診断メッセージの生成を有効にすることができます。

パラメータ

パラメータ	宣言	データタイプ		既定	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		FALSE	拡張ファンクションを設定するための命令は、当該入力の立ち上がりエッジで開始されます。
PORT	IN	PORT (UInt)	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
EN_MOD-BUS_CRC	IN	Bool		FALSE	Modbus CRC のサポートの有効化
EN_DI-AG_ALARM	IN	Bool		FALSE	診断メッセージの生成の有効化
EN_SUP-PLY_VOLT	IN	Bool		FALSE	電源電圧 L+が不明の診断の有効化
COM_RST	IN/OUT	---	Bool	FALSE	Set_Features 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DONE	OUT	Bool		FALSE	最後の要求がエラーなしで完了した後、1 回の実行の間 TRUE
ERROR	OUT	Bool		FALSE	命令がエラーで完了した場合、1 サイクルの間 TRUE
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)

エラーメッセージ

エラーメッセージの概要 - PtP

エラーメッセージは命令の STATUS 出力で提供され、そこで評価したりユーザープログラムで処理したりすることができます。

エラーコード	説明	対策
16#0000	エラーはありません。	-
受信ステータスとエラーコード		
16#0094	「固定/最大フレーム長の受信」に基づいて識別されたフレームの終了	-
16#0095	「メッセージタイムアウト」に基づいて識別されたフレームの終了	-
16#0096	「キャラクタ遅延時間」の経過に基づいて識別されたフレームの終了	-
16#0097	最大応答時間に達したためにフレームが中止されました。	-
16#0098	「メッセージからのメッセージ長の読み出し」条件の適合に基づいて識別されたフレームの終了	-
16#0099	「終了シーケンス」の受信に基づいて識別されたフレームの終了	-
送信ステータスとエラーコード		
16#7000	ブロックアイドル	-
16#7001	新しいフレームの最初の呼び出し データ伝送が開始されました	-
16#7002	中間呼び出し: データ伝送が実行中	-
16#8085	長さが無効	適切なフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8088	指定された長さが受信バッファの設定された範囲を超えています。 注記: データタイプ STRING が BUFFER パラメータで指定されている場合、現在の文字列が LENGTH パラメータで指定された長さよりも短いと、このエラーコードも表示されます。	受信バッファの範囲を変更するか、または受信バッファの設定された範囲に対応するフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8090	構成エラー: WString のバイト数が奇数です	偶数のバイト数を選択します。
受信ステータスとエラーコード		
16#8088	受信した文字数が BUFFER パラメータで指定された文字数を超えています。	適切なフレーム長を選択します。

		以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8090	構成エラー: WString のバイト数が奇数です	偶数のバイト数を選択します。
特殊ファンクションのエラーメッセージコード		
16#818F	不正なパラメータ番号の設定(USS のみ)	適切なパラメータ番号(PARAM)を選択します。 次の番号が有効です。 0-2047
16#8190	CRC 計算の不正な設定	CRC 計算の適切な値を選択します。 以下が有効です。 無効または有効。 アドレス指定されたモジュールが CRC 計算をサポートしているかどうかチェックします。
16#8191	診断エラー割り込みの不正な設定	[診断エラー割り込み]の適切な値を選択します。 以下が有効です。 診断エラー割り込みが無効、または診断エラー割り込みが有効。 アドレス指定されたモジュールが診断割り込みの生成をサポートしているかどうかをチェックします。
「ポートの構成」のエラーメッセージコード		
16#81A0	モジュールがこのプロトコルをサポートしていません。	モジュールの有効なプロトコル(PROTOCOL)を選択します。
16#81A1	モジュールがこのデータ伝送率をサポートしていません。	モジュールの有効なデータ伝送速度(BAUD)を選択します。
16#81A2	モジュールがこのパリティ設定をサポートしていません。	「パリティ」(PARITY)の適切な値を選択します。 以下が有効です。 • なし(1) • 偶数(2) • 奇数(3) • マーク(4) • スペース(5) • 任意(6)
16#81A3	モジュールがこのデータビット数をサポートしていません。	「データビット数」(DATABITS)の適切な値を選択します。 以下が有効です。 • 7 (2) • 8 (1)
16#81A4	モジュールがこのストップビット数をサポートしていません。	「ストップビット数」(STOPBITS)の適切な値を選択します。 以下が有効です。 • 1 (1) • 2 (2)

16#81A5	モジュールがこのタイプのデータフロー制御をサポートしていません。	モジュールの有効なデータフロー制御(FLOWCTRL)を選択します。
16#81A7	XON または XOFF の無効な値	XON (XONCHAR)および XOFF (XOFF-CHAR)の適切な値を選択します。 値の有効な範囲: 0...255
16#81AA	動作モードが無効	有効な動作モード以下のとおりです。 • 全二重(RS232) (0) • 全二重(RS422) 4 線式モード(ポイントツーポイント) (1)/ (CM PtP (ET 200SP)) • 全二重(RS422) 4 線式モード(マルチポイントマスタ) (2)/ (CM PtP (ET 200SP)) • 全二重(RS 422) 4 線式モード(マルチポイントスレーブ) (3) • 半二重(RS485) 2 線式モード (4)
16#81AB	回線初期状態が無効	有効な初期状態は以下のとおりです。 • 初期設定[なし] (0) • シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) (1) 以下とともにのみ選択できます。「全二重(RS422) 4 線式モード(ポイントツーポイント接続)」および「全二重(RS422) 4 線式モード(マルチポイントスレーブ)」 • シグナル R(A) 0 V/シグナル R(B) 5 V (2): この初期設定はアイドル状態に相当します(送信動作がアクティブでない)。
16#81AC	[ブレーク検出]の値が無効	[ブレーク検出]の適切な値を選択します。以下が有効です。 • ブレーク検出が無効(0) • ブレーク検出が有効(1)
16#81AF	モジュールがこのプロトコルをサポートしていません。	モジュールの有効なプロトコルを選択します。
「送信構成」のエラーコード		
16#81B5	エンドデリミタが 2 つ以上、または終了シーケンスが 5 文字以上	[エンドデリミタ]と[終了シーケンス]に適切な値を選択します。 以下が有効です。 • 無効(0)、 • 1 つ (1)または 2 つ (2)のエンドデリミタ または • 無効(0)、 • 終了シーケンスに 1 文字 (1)から 5 文字 (5)
16#81B6	3964(R)プロトコルが選択されていたため、送信構成が拒否されました。	3964(R)プロトコルが設定されている場合は、送信構成が送信されないようにします。

「受信構成」のエラーコード		
16#81C0	開始条件が無効	適切な開始条件を選択します。 以下が有効です。 • フレーム開始前の送信中断 • アイドルラインの送信
16#81C1	終了条件が無効、または終了条件が選択されていません	適切な条件を選択します(Freeport によるデータ送信を参照)。
16#81C3	[最大メッセージ長]の値が無効	「最大メッセージ長」(MAXLEN)の適切な値を選択します。 値の有効な範囲(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#81C4	[メッセージ内の長さ指定のオフセット]の値が無効	[メッセージ内の長さ指定のオフセット]に有効な値を選択します。 値の有効な範囲(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#81C5	[長さフィールドのサイズ]の値が無効	「長さフィールドのサイズ」(LENGTHSIZE)の適切な値を選択します。 値の有効な範囲(バイト単位): • 1 (1) • 2 (2) • 4 (4)
16#81C6	[長さ指定でカウントされていない文字数]の値が無効	「長さ指定でカウントされていない文字数」(LENGTHM)の適切な値を選択します。 値の有効な範囲: 0～255 (バイト)
16#81C7	「メッセージ内のオフセット + 長さフィールドのサイズ + カウントされていない文字数」の合計が最大フレーム長を超えています	[メッセージ内のオフセット]、[長さフィールドのサイズ]、および[カウントされていない文字数]に適切な値を選択します。 値の有効な範囲: • メッセージ内のオフセット(モジュールによって異なります): 0-1024/2048/4096 (Byte) • 長さフィールドのサイズ: 1、2、または 4 (バイト) • カウントされていない文字数: 0～255 (バイト)
16#81C8	[応答タイムアウト]の値が無効	[応答タイムアウト]に適切な値を選択します。 値の有効な範囲: 1-65535 (ms)
16#81C9	[キャラクタ遅延時間]の値が無効	[キャラクタ遅延時間]に適切な値を選択します。 値の有効な範囲: 1～65535 (ビット時間)
16#81CB	フレーム終了シーケンスが有効になっていますが、チェック用の文字が有効になっていません	チェック用に 1 つ以上の文字を有効にします。

16#81CC	フレーム開始シーケンスが有効になっていますが、チェック用の文字が有効になっていません	チェック用に 1 つ以上の文字を有効にします。
16#81CD	[上書き禁止]の値が無効	[上書き禁止]に適切な値を選択します。 以下が有効です。 • 上書き禁止が無効(0)または • 上書き禁止が有効(1)
16#81CE	[スタートアップ時に受信バッファをクリアする]の値が無効	[スタートアップ時に受信バッファをクリアする]に適切な値を選択します。 以下が有効です。 • スタートアップ時に受信バッファをクリアが無効(0) • スタートアップ時に受信バッファをクリアが無効(1)
送信ステータスとエラーコード		
16#81D0	送信コマンドのランタイム中の送信要求の受信	送信コマンドのランタイム中に追加の送信要求を受信するのを防止します。
16#81D1	XON または CTS = ON の待機時間が経過しました。	通信パートナーで障害が発生したか、速度が遅すぎるか、またはオフラインになっています。 通信パートナーをチェックするか、または必要に応じてパラメータを変更します。
16#81D2	「ハードウェア RTS は常に ON」: DSR = ON から OFF への切り替えのために送信ジョブがキャンセルされました	通信パートナーをチェックします。 伝送時間全体にわたって、DSR が確実に ON になるようにします。
16#81D3	送信バッファオーバーフロー/送信フレームが長すぎます	短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1～1024/2048/4096 (バイト)
16#81D5	パラメータ変更、断線の検出、または CPU の STOP によって、伝送がキャンセルされました	パラメータ割り当て、断線、および CPU ステータスをチェックします。
16#81D6	終了識別子が受信されなかったため、伝送がキャンセルされました。	終了文字のパラメータ割り当てと通信パートナーのフレームをチェックします。
16#81D7	ユーザープログラムとモジュール間の通信エラー	通信をチェックします(たとえば、シーケンス番号の一致など)。
16#81D8	モジュールが構成定義されていないため、伝送の試行が拒否されました。	モジュールを構成定義します。
受信構成のエラーコード		
16#81E0	フレームが中断されました: 送信バッファオーバーフロー/送信フレームが長すぎます	ユーザープログラムで受信ファンクションの呼び出し率を上げるか、またはデータフロー制御による通信を構成定義する必要があります。
16#81E1	フレームが中断されました: パリティエラー	通信パートナーの接続線をチェックするか、または両方のデバイスが同じデータ伝送率、パリティ、ストップビット数に構成定義されているかどうかを検証します。

16#81E2	フレームが中断されました: キャラクタフレームエラー	スタートビット、データビット、パリティビット、データ伝送率、ストップビットの設定をチェックします。
16#81E3	フレームが中断されました: キャラクタオーバーフロエラー	ファームウェアエラー: カスタマーサポートに問い合わせてください。
16#81E4	フレームが中断されました: 「メッセージ内のオフセット + 長さフィールドのサイズ + カウントされていない文字数」の合計長が受信バッファを超えています	メッセージ内のオフセット、長さフィールドのサイズ、およびカウントされていない文字数に適切な値を選択します。
16#81E5	フレームが中断されました: break	パートナーへの受信ラインが中断しています。 再接続するか、またはパートナーのスイッチをオンにします。
16#81E6	[バッファリングされる受信フレーム]の最大数を超過しました	ユーザープログラムで命令を呼び出す回数を増やすか、データフロー制御による通信を構成定義するか、またはバッファリングされるフレームの数を増やします。
16#81E7	同期エラーモジュールと Receive_P2P	Receive_P2P のさまざまな異なるインスタンスが確実に同一モジュールにアクセスしないようにします。
16#81E8	フレームが中断されました: メッセージまたは条件が検出される前に、キャラクタ遅延時間が経過しました。	パートナーデバイスに障害が発生しているか、またはパートナーデバイスが遅すぎます。 必要に応じてインターフェーステスターを使用し、パートナーデバイスが伝送線で相互接続されていることをチェックします。
16#81E9	Modbus CRC エラー(Modbus をサポートしている通信モジュールのみ)	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#81EA	Modbus フレームが短すぎます(Modbus をサポートしている通信モジュールのみ)	Modbus フレームの最小長さが満たされていません 通信パートナーをチェックします。
16#81EB	フレームが中断されました: 最大フレーム長に達しました	通信パートナー側で短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte) フレーム終了検出のパラメータをチェックします。
エラーコード V24 に付随する信号		
16#81F0	モジュールが V24 に付随する信号をサポートしていません。	V24 に付随する信号をサポートしていないモジュールに対して付随する信号の設定を試みました。 必ず RS232 モジュールにするか、または RS232 モード(ET 200SP)を設定します。
16#81F1	V24 に付随する信号を操作できません	ハードウェアデータフロー制御が有効の場合、V24 に付随する信号を手動で操作することはできません。
受信構成のエラーコード		
16#8201 1)	Receive_Conditions は無効なデータタイプへのポインタです。	データタイプ

		DB、BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、DATE、TIME_OF_DAY、TIME、S5TIME、DATE_AND_TIME、STRING の 1 つを指すポインタを入力します。
16#8225	Receive_Conditions が、1 KB 以上の最適化されたメモリ領域をポイントしています。 または Receive_Conditions が最適化されたメモリ領域をポイントしており、受信長が Receive_Conditions でアドレス指定された領域を越えています。	以下の最大長を持つ領域へのポインタを入力します。 • 最適化されたメモリ領域: 1 kByte • 最適化されていない領域: 4 kByte 注記: ポインタが最適化されたメモリ領域を指す場合、1 kByte を超えるバイト数を送信してはいけません。
16#8229 1)	Receive_Conditions が、ビット数が n * 8 と等しくない BOOL へのポインタです。	BOOL へのポインタを使用している場合、ビット数は 8 の倍数とする必要があります。
エラーコード、一般		
16#8280	モジュールの読み出し時の否定確認	エラーの原因の詳細については、RDREC.STATUS 静的パラメータと、SFB RDREC の説明を参照してください。 • PORT パラメータの入力をチェックします。 • 初期呼び出しの前に、COM_RST パラメータを設定します。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、WRREC.STATUS 静的パラメータと、SFB WRREC の説明を参照してください。
16#8282	モジュールが使用不可	PORT パラメータの入力をチェックし、モジュールに確実にアクセスできるようにします。
受信構成のエラーコード		
16#82C1	[バッファリングされた受信フレーム]の値が無効	[バッファリングされた受信フレーム]に適切な値を選択します。 値の有効な範囲: 1-255
16#82C2	3964(R)プロトコルが選択されていたため、受信構成が拒否されました。	3964(R)プロトコルが設定されている場合は、受信構成が送信されないようにします。
16#8301 1)	Receive_Conditions は無効なデータタイプへのポインタです。	有効なデータタイプを選択します。 以下が有効です。 DB、BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、DATE、TIME_OF_DAY、TIME、S5TIME、DATE_AND_TIME、STRING
16#8322	パラメータ読み出し時の範囲長さエラー	Receive_Conditions パラメータの入力をチェックします。
16#8324	パラメータ読み出し時の範囲エラー	Receive_Conditions パラメータの入力をチェックします。
16#8328	パラメータ読み出し時の設定エラー	Receive_Conditions パラメータの入力をチェックします。

送信ステータスとエラーコード		
16#8328 1)	BUFFER が、ビット数が n * 8 と等しくない BOOL へのポインタです	BOOL へのポインタを使用している場合、ビット数は 8 の倍数とする必要があります。
受信構成のエラーコード		
16#8332	Receive_Conditions パラメータの無効なデータブロック	Receive_Conditions パラメータの入力をチェックします。
16#833A	Receive_Conditions パラメータのデータブロックの名称が、ロードされていないデータブロックを示しています。	Receive_Conditions パラメータの入力をチェックします。
16#8351	無効なデータタイプ	Receive_Conditions パラメータの入力をチェックします。
16#8352 1)	Receive_Conditions がデータブロックをポイントしていません。	Receive_Conditions へのポインタをチェックします。
16#8353 1)	Receive_Conditions タイプの構造をポイントしていません。 Receive_Conditions	Receive_Conditions へのポインタをチェックします。
エラーコード 3964(R)プロトコル		
16#8380	パラメータ割り当てエラー: [キャラクタ遅延時間]の値が無効	[キャラクタ遅延時間](CharacterDelay-Time)に適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8381	パラメータ割り当てエラー: [応答タイムアウト]の値が無効	[応答タイムアウト](AcknDelayTime)に適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8382	パラメータ割り当てエラー: [優先度]の値が無効	[優先度](Priority)に適切な値を選択します。 以下が有効です。 • 高い(1) • 低い(0)
16#8383	パラメータ割り当てエラー: [ブロックチェック]の値が無効	[ブロックチェック] (BCC)の適切な値を選択します。 以下が有効です。 • ブロックチェックあり(1) • ブロックチェックなし(0)
16#8384	パラメータ割り当てエラー: [接続試行]の値が無効.	[接続試行](BuildupAttempts)に適切な値を選択します。 値の有効な範囲: 1-255
16#8385	パラメータ割り当てエラー: [送信試行]の値が無効.	[送信試行](RepetitionAttempts)に適切な値を選択します。 値の有効な範囲: 1-255
16#8386	ランタイムエラー: 接続試行回数を超過しました	インターフェースケーブルおよび伝送パラメータをチェックします。 さらに、パートナーデバイス側で受信ファンクションが正しく構成定義されているかどうかもチェックします。

16#8387	ランタイムエラー: 送信試行回数を超過しました	インターフェースケーブル、伝送パラメータ、および通信パートナーの構成をチェックします。
16#8388	ランタイムエラー:「ブロックチェックキャラクタ」のエラー 内部的に計算したブロックチェックキャラクタの値が、接続の終了時にパートナーが受信したブロックチェックキャラクタに対応していません。	接続が致命的に中断されていないかどうかチェックします。その場合は、こまめにエラーコードをチェックすることをお勧めします。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#8389	ランタイムエラー: 空き受信バッファの待機中に無効な文字が受信されました	通信パートナーの送信要求(STX, 02H)は、受信バッファが空いている場合のみ、DLEで応答します。 この前に、追加文字を受信することはできません(再び、STX を除いて)。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838A	ランタイムエラー: 受信時の論理エラー DLE 受信後、追加のランダム文字(DLE または ETX を除く)を受信しました。	フレームヘッダー内およびデータ文字列内のパートナー DLE が常に重複しているか、または接続が DLE ETX でリリースされているかをチェックします。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838B	ランタイムエラー: キャラクタ遅延時間を超過しました	パートナーデバイスが遅すぎるか、またはパートナーデバイスに障害が発生しています。 必要に応じて、伝送線内に接続されたインターフェーステストデバイスを使用して検証します。
16#838C	ランタイムエラー: 空き受信バッファの待機時間が開始されました	ユーザープログラムで命令を呼び出す回数を増やすか、またはデータフロー制御による通信を構成定義します。
16#838D	ランタイムエラー: フレームの繰り返しが NAK 後、4 秒以内に開始されません。	通信パートナーをチェックします。 破壊されている可能性のある受信フレームをパートナーが 4 秒以内に繰り返す必要があります。
16#838E	ランタイムエラー: アイドルモードで、1 つ以上の文字(NAK、STX 以外)を受信しました。	パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838F	ランタイムエラー: 初期化の競合 - 両方のパートナーが高い優先度を設定しています。	パートナーのどちらかで、「低い」優先度を設定します。
16#8391	パラメータ割り当てエラー: Freeport が設定されているため、3964 構成データが拒否されました	Freeport プロトコルが設定されている場合は、3964 パラメータ割り当てデータが送信されないようにします。
[1] S7-300/400 CPU 用の命令の場合のみ		

エラーメッセージの概要 - Modbus

エラーコード	説明	対策
16#0000	エラーはありません。	-
インターフェースの構成エラー - Modbus_Comm_Load		
16#8181	モジュールがこのデータ伝送率をサポートしていません。	BAUD パラメータで、モジュールの有効なデータ伝送速度を選択します。
16#8182	モジュールがこのパリティ設定をサポートしていません。	PARITY パラメータの「パリティ」の適切な値を選択します。 以下が有効です。 • なし(1) • 偶数(2) • 奇数(3) • マーク(4) • スペース(5) • 任意(6)
16#8183	モジュールがこのタイプのデータフロー制御をサポートしていません。	FLOW_CTRL パラメータで、モジュールの有効なデータフロー制御を選択します。
16#8184	[応答タイムアウト]の値が無効	RESP_TO パラメータで、[応答タイムアウト]の適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、Send_Config.RDREC.STATUS/ Receive_Config.RDREC.STATUS 静的パラメータまたは RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、Send_Config.WRREC.STATUS/ Receive_Config.WRREC.STATUS 静的パラメータまたは WRREC.STATUS と、SFB WRREC の説明を参照してください。
16#8282	モジュールが使用不可	PORT パラメータの入力をチェックし、モジュールに確実にアクセスできるようにします。
構成エラー - Modbus_Slave		
16#8186	スレーブアドレスが無効	MB_ADDR パラメータで、適切なスレーブアドレスを選択します。 以下が有効です。 標準アドレス領域では 1～247; 拡張アドレス領域では 1～65535

		(0 はブロードキャスト用に予約済みです)
16#8187	MB_HOLD_REG パラメータの無効な値	MB_HOLD_REG パラメータで、保持レジスタの適切な値を選択します。
16#8188	動作モードまたはブロードキャストが無効(MB_ADDR = 0)および MODE パラメータ ≠ 1	ブロードキャストモードで MODE の値 1 を選択するか、または 異なる動作モードを選択します。
16#818C	MB_HOLD_REG 領域へのポインタは、データブロックまたはビットメモリアドレス領域であることが必要です。	MB_HOLD_REG 領域へのポインタの適切な値を選択します。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.RDREC.STATUS または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.WRREC.STATUS または Receive_P2P.WRREC.STATUS と、SFB WRREC の説明を参照してください。
16#8452 1)	MB_HOLD_REG が、DB またはビットメモリ領域へのポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8453 1)	MB_HOLD_REG が、タイプ BOOL または WORD のポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8454 1)	MB_HOLD_REG でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が読み出しまたは書き込みされるデータバイトの数に対して小さすぎます。	MB_HOLD_REG ポインタをチェックします
16#8455 1)	MB_HOLD_REG が書き込み禁止 DB を指しています	MB_HOLD_REG ポインタをチェックします
16#8456 1)	命令実行中のエラー エラーの原因は、STATUS パラメータに示されます。	SFCSTATUS パラメータの値を識別します。 SFC51、STATUS パラメータの説明でこの値が意味することをチェックします。
構成エラー - Modbus_Master		
16#8180	MB_DB パラメータの値が無効	Modbus_Comm_Load 命令で MB_DB(インスタンスデータ DB)に構成定義された値が有効ではありません。 Modbus_Comm_Load 命令とそのエラーメッセージの相互接続をチェックします。
16#8186	ステーションアドレスが無効	MB_ADDR パラメータで、適切なステーションアドレスを選択します。 以下が有効です。 標準アドレス領域では 1～247; 拡張アドレス領域では 1～65535 (0 はブロードキャスト用に予約済みです)

16#8188	動作モードまたはブロードキャストが無効(MB_ADDR = 0)および MODE パラメータ ≠ 1	ブロードキャストモードで MODE の値 1 を選択するか、または異なる動作モードを選択します。
16#8189	データアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。 情報システムの説明 Modbus_Master を選択します。
16#818A	長さが無効	DATA_LEN パラメータで、適切なデータ長を選択します。 情報システムの説明 Modbus_Master を選択します。
16#818B	DATA_PTR の値が無効	DATA_PTR パラメータで、データポインタの適切な値を選択します(M または DB アドレス)。 情報システムの説明 Modbus_Master を選択します。
16#818C	DATA_PTR パラメータの相互接続エラー	命令の相互接続をチェックしてください。
16#818D	DATA_PTR でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が 読み出しまたは書き込みされるデータバイトの数 に対して小さすぎます。	DATA_PTR ポインタをチェックします。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.RDREC.STATUS または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。. エラーの原因の詳細については、静的パラメータ Send_P2P.WRREC.STATUS、Receive_P2P.WRREC.STATUS 、または Receive_Reset と、SFB WRREC の説明を参照してください。
通信エラー - Modbus_Master および Modbus_Slave		
16#80D1	XON または CTS = ON の待機時間が経過しました。	通信パートナーで障害が発生したか、速度が遅すぎるか、またはオフラインになっています。 通信パートナーをチェックするか、または必要に応じてパラメータを変更します。
16#80D2	「ハードウェア RTS は常に ON」: DSR = ON から OFF への切り替えのために送信ジョブがキャンセルされました	通信パートナーをチェックします。 伝送時間全体にわたって、DSR が確実に ON になるようにします。
16#80E0	フレームが中断されました: 送信バッファオーバーフロー/送信フレームが長すぎます	ユーザープログラムで命令を呼び出す回数を増やすか、またはデータフロー制御による通信を構成定義します。

16#80E1	フレームが中断されました: パリティエラー	通信パートナーの接続線をチェックするか、または両方のデバイスが同じデータ伝送率、パリティ、ストップビット数に構成定義されているかどうかを検証します。
16#80E2	フレームが中断されました: キャラクタフレームエラー	スタートビット、データビット、パリティビット、データ伝送率、ストップビットの設定をチェックします。
16#80E3	フレームが中断されました: キャラクタオーバーフロエラー	通信パートナーのフレーム内のデータ数をチェックします。
16#80E4	フレームが中断されました: 最大フレーム長に達しました	通信パートナー側で短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1～1024/2048/4096 (バイト)
通信エラー - Modbus_Master		
16#80C8	スレーブが設定された時間内に応答しません。	スレーブのデータ伝送率、パリティ、および配線をチェックします。
16#80C9	スレーブが Blocked_Proc_Timeout によって設定された時間内に応答しません。	Blocked_Proc_Timeout の設定をチェックします。 モジュールが、Modbus_Comm_Load 命令を使用して構成定義されたかどうかをチェックします。 モジュールは、抜き差し後または電圧復旧後に Modbus_Comm_Load を使用して再構成しなければならない場合があります。
16#8200	インターフェースが出力要求でビジーになっています。	後からコマンドを繰り返します。 新しいコマンドを開始する前に、まだ実行中のコマンドが存在しないことを確認します。
プロトコルエラー - Modbus_Slave (Modbus をサポートしている通信モジュールの場合のみ)		
16#8380	CRC エラー	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#8381	ファンクションコードがサポートされていない、またはブロードキャストでサポートされていません。	通信パートナーをチェックして、有効なファンクションコードが送信されていることを確認します。
16#8382	要求フレーム内の情報の長さが無効	DATA_LEN パラメータで、適切なデータ長を選択します。
16#8383	要求フレーム内のデータアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。
16#8384	要求フレーム内のデータ値無効エラー	Modbus マスタの要求フレーム内のデータ値をチェックします。
16#8385	診断値が Modbus スレーブでサポートされません(ファンクションコード 08)	Modbus スレーブは、診断値 16#0000 および 16#000A のみをサポートします。
プロトコルエラー - Modbus_Master (Modbus をサポートしている通信モジュールの場合のみ)		
16#8380	CRC エラー	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#8381	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: ファンクションコードがサポートされていません。	通信パートナーをチェックして、有効なファンクションコードが送信されていることを確認します。

16#8382	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 長さが無効	適切なデータ長を選択します。
16#8383	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 要求フレーム内のデータアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。
16#8384	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: データ値エラー	Modbus スレーブへの要求フレームをチェックします。
16#8385	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 診断値が Modbus スレーブでサポートされていません。	Modbus スレーブは、診断値 16#0000 および 16#000A のみをサポートします。
16#8386	戻されたファンクションコードが要求されたファンクションコードと一致しません。	スレーブの応答フレームとアドレス指定をチェックします。
16#8387	応答を要求されなかったスレーブ	スレーブの応答フレームをチェックします。 スレーブのアドレス設定をチェックします。
16#8388	書き込み要求に対するスレーブの応答のエラー	スレーブの応答フレームをチェックします。
16#8828 [1)]	DATA_PTR が、n * 8 と等しくないビットアドレスを指しています	DATA_PTR ポインタをチェックします。
16#8852 [1)]	DATA_PTR が、DB またはビットメモリ領域へのポインタではありません	DATA_PTR ポインタをチェックします。
16#8853 [1)]	DATA_PTR が、タイプ BOOL または WORD のポインタではありません	DATA_PTR ポインタをチェックします。
16#8855 [1)]	DATA_PTR が書き込み禁止 DB を指しています	DATA_PTR ポインタをチェックします。
16#8856 [1)]	SFC51 呼び出し中のエラー	Modbus_Master 命令を再度呼び出してください。
エラー - Modbus_Slave (Modbus をサポートしている通信モジュールの場合のみ)		
16#8428 [1)]	MB_HOLD_REG が、n * 8 と等しくないビットアドレスを指しています	MB_HOLD_REG ポインタをチェックします
16#8452 [1)]	MB_HOLD_REG が、DB またはビットメモリ領域へのポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8453 [1)]	MB_HOLD_REG が、タイプ BOOL または WORD のポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8454 [1)]	MB_HOLD_REG でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が読み出しまたは書き込みされるデータバイトの数に対して小さすぎます。	MB_HOLD_REG ポインタをチェックします
16#8455 [1)]	MB_HOLD_REG が書き込み禁止 DB を指しています	MB_HOLD_REG ポインタをチェックします
16#8456 [1)]	SFC51 呼び出し中のエラー	Modbus_Slave 命令を再度呼び出してください。
[1)] S7-300/400 CPU 用の命令の場合のみ		

エラーメッセージの概要 - USS

エラーコード	説明	対策
16#0000	エラーはありません。	-
16#8180	ドライブの応答の長さエラー:	ドライブの応答フレームをチェックします。
16#8181	データタイプエラー	適切なデータタイプを選択します。 以下が有効です。 • REAL • ワード • ダブルワード
16#8182	データタイプエラー:「ダブルワード」または「実数」を「ワード」要求に対して返すことはできません。	ドライブの応答フレームをチェックします。
16#8183	データタイプエラー:「ワード」を「ダブルワード」または「実数」要求に対して返すことはできません。	ドライブの応答フレームをチェックします。
16#8184	ドライブの応答のチェックサムエラー:	ドライブと通信接続をチェックします。
16#8185	アドレス指定エラー	有効なドライブアドレス範囲: 1 ~ 16
16#8186	セットポイントエラー	有効なセットポイント範囲: -200 % ~ +200 %
16#8187	不正なドライブ番号が返されました	ドライブの応答フレームをチェックします。
16#8188	PZD 長さが無効	許可されている PZD 長さ: 2、4、6、8 ワード
16#8189	モジュールがこのデータ伝送率をサポートしていません。	モジュールの有効なデータ伝送率を選択します。
16#818A	このドライブに対する別の要求が現在有効です。	後から、パラメータ読み出しまたは書き込みコマンドを繰り返します。
16#818B	ドライブが応答しません。	ドライブをチェックします。
16#818C	ドライブがパラメータ要求に対してエラーメッセージで応答しました	ドライブの応答フレームをチェックします。 パラメータ要求をチェックします。 命令 USS_Read_Param、USS_Write_Param または USS_Port_Scan にエラーの報告があるかチェックします。 エラーの報告がある場合、USS_Drive_Control 命令の静的タグ USS_DB. w_USSExtendedError の値をチェックします。
16#818D	ドライブがパラメータ要求に対してアクセスエラーメッセージで応答しました	ドライブの応答フレームをチェックします。 パラメータ要求をチェックします。
16#818E	ドライブが初期化されていません。	ユーザープログラムをチェックして、USS_Drive_Control 命令がこのドライブに

		対して呼び出されていることを確認します。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Port_Config.RDREC.STATUS、Send_Config.RDREC.STATUS 、 Receive_Config.RDREC.STATUS 、Send_P2P.RDREC.STATUS 、または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。. エラーの原因の詳細については、静的パラメータ Port_Config.WRREC.STATUS、Send_Config.WRREC.STATUS 、 Receive_Config.WRREC.STATUS 、Send_P2P.RDREC.STATUS 、または Receive_P2P.RDREC.STATUS と、SFB WRREC の説明を参照してください。
[1] S7-300/400 CPU 用の命令の場合のみ		

USS

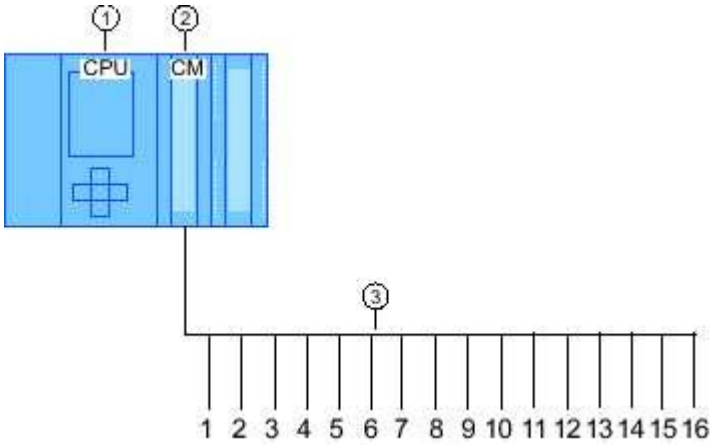
この章には下記に関する情報が記載されています：

USS 通信の概要

USS 通信

USS 命令は、ユニバーサルシリアルインターフェース(USS)のプロトコルをサポートするドライブ操作を制御します。 RS485 接続によって複数のドライブと、USS 命令によって PtP 通信モジュールとの通信を行うことができます。 RS485 の各ポートは最大 16 ドライブを操作できます。

USS プロトコルは、シリアルバスを経由して通信にマスタ/スレーブネットワークを使用します。マスタはアドレスパラメータを使用してデータを選択されたスレーブに送信します。 スレーブは、最初に送信要求を受信しないと送信できません。 個々のスレーブ間の通信は行えません。 USS 通信は半二重モードで動作します。 下の図に、16 ドライブによるアプリケーション例のネットワーク図を示します。



① CPU

② CM

③ USS ネットワーク内の USS ドライブ

S7-1500 通信モジュールでの配線例

注記

RS232 によるドライブとの通信

ドライブとの通信では、基本的に CM PtP RS232 BA および CM PtP RS232 HF も使用できます。ただし、RS232 ポートに接続できるドライブは **1 つ**のみです。

RS422 によるドライブとの通信

ドライブとの通信では、基本的に CM PtP RS422/485 BA および CM PtP RS422/485 HF の RS422 インターフェースも使用できます。 ただし、RS422 ポートに接続できるドライブは **1 つ**のみです。

プログラムの USS 命令

- USS_Port_Scan: 命令 USS_Port_Scan では、USS ネットワークを使用して最大 16 台のドライブと通信モジュール経由の通信を行うことができます(周期的に命令を呼び出す必要があります)。

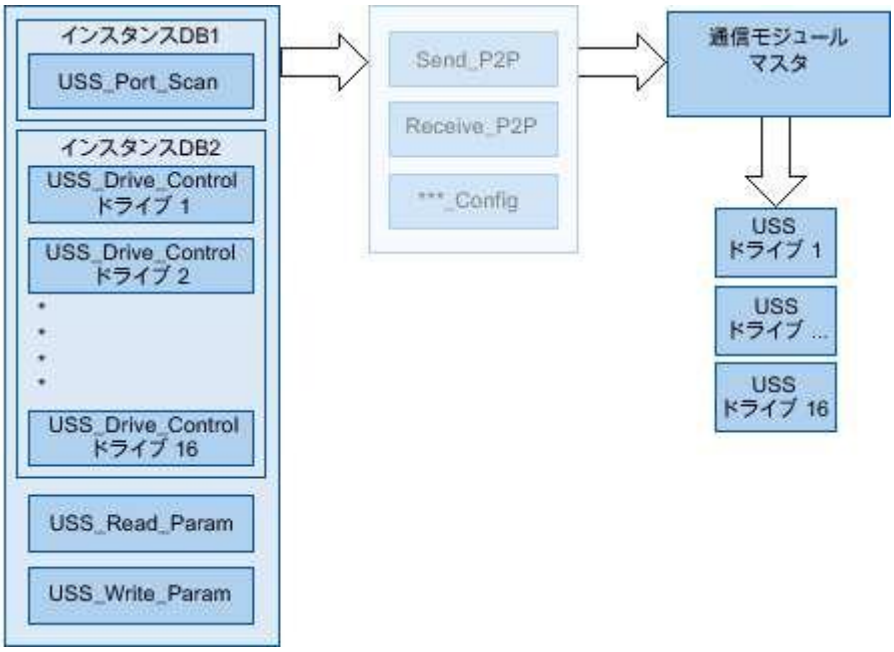
  プログラムには、PtP 通信ポートあたり 1 つずつ USS_Port_Scan 命令があり、この命令がすべてのドライブへの通信を制御します。

- USS_Drive_Control: 命令 USS_Drive_Control を使用して、USS_Port_Scan からのドライブへの送信データを準備し、その受信データを表示することができます。

  USS_Drive_Control は、送信するデータを設定し、前の要求で USS_Port_Scan から受信したデータを評価します。
- USS_Read_Param: 命令 USS_Read_Param を使用して、ドライブからパラメータを読み出すことができます。
- USS_Write_Param: 命令 USS_Write_Param を使用して、ドライブのパラメータを変更することができます。

USS プロトコル使用の必要条件

4 つの USS 命令は、2 つの FB と 2 つの FC を使用して USS プロトコルをサポートします。 各 USS ネットワークで、1 つのインスタンスデータブロック(DB)が USS_Port_Scan に使用され、1 つのインスタンスデータブロックが USS_Drive_Conrol のすべての呼び出しに使用されます。



USS プログラムシーケンス

1 つの RS485 ポートに接続されたすべてのドライブ(最大 16)は、同じ USS ネットワークの一部です。 別の RS485 ポートに接続されたすべてのドライブは、別の USS ネットワークの一部です。 各 USS ネットワークは、すべての USS_Drive_Control 命令に対する固有のインスタンスデータブロックと、USS_Port_Scan 命令に対するもう 1 つのインスタンスデータブロックで管理されます。 USS ネットワークの一部であるすべての命令は、USS_Drive_Conrol 用のこのインスタンスデータブロックを共有する必要があります。 USS_Port_Scan, USS_Read_Param および USS_Write_Param 命令には、このファンクションのための USS_DB パラメータがあります。 このパラメータは、USS_Drive_Control 命令のインスタンス DB の(静的)USS_DB パラメータと接続する必要があります。

- 命令 USS_Drive_Control および USS_Port_Scan はファンクションブロック(FB)です。 USS_Drive_Control または USS_Port_Scan 命令をプログラムエディタに追加する場合、[呼び出しオプション]ダイアログでこの FB 用の DB の割り当てを促すプロンプトが表示されます。 この命令が、この USS ネットワークのこのプログラム内の最初の USS_Drive_Control 命令であった場合、DB の標準割り当てを適用(または、必要に応じて名前を変更)することができ、新しい DB が作成されます。 ただし、この命令がこのドライブに対する最初の USS_Drive_Control 命令ではない場合、[呼び出しオプション]ダイアログのドロップダウンリストから、この USS ネットワークに既に割り当て済みの DB を選択する必要があります。
- USS_Port_Scan および USS_Read_Param 命令はファンクション(FC)です。 エディタでこれらの FC を追加する場合、DB は割り当てられません。 エディタでこれらの FC または USS_Port_Scan 命令を追加する場合は、対応する USS_Drive_Control インスタンス DB の USS_DB パラメータをこれらの命令の USS_DB 入力に割り当てる必要があります。 パラメータフィールドをダブルクリックし、シンボルをクリックして使用可能な DB を表示します。 ピリオド「.」を入力し、ドロップダウンリストから USS_DB パラメータを選択します。
- USS_Port_Scan ファンクションは、ポイントツーポイント RS485 通信ポート経由の CPU とドライブ間の通信を制御します。 ドライブとの通信は、このファンクションを呼び出すたびに処理されます。 ドライブがタイムアウトを通知しないように、このファンクションを素早く呼び出す必要が

あります。 フレーム通信に対して一定の時間応答を保証するには、この命令をサイクリック割り込み OB で呼び出す必要があります。

- USS_Drive_Control 命令を使用すると、プログラムが USS ネットワーク内の指定したドライブにアクセスできます。 入力と出力はドライブの状態およびオペレーティングファンクションに対応します。 ネットワーク内に 16 のドライブがある場合、プログラム内で USS_Drive_Control を少なくとも 16 回呼び出す必要があります。つまりドライブごとに 1 回です。

  USS_Drive_Control 命令は、サイクリック OB からのみ呼び出します。
- ドライブの動作パラメータは、USS_Read_Param および USS_Write_Param ファンクションで読み出しおよび書き込みします。 これらのパラメータは、ドライブの内部動作を制御します。 これらのパラメータの定義については、ドライブのマニュアルを参照してください。 プログラムにはこれらのファンクションの番号も含まれていることがありますが、ドライブに対しては 1 回の読み出し要求または 1 回の書き込み要求のみが有効です。 メインプログラムのサイクル OB から USS_Read_Param および USS_Write_Param ファンクションのみを呼び出すことができます。

通知

USS 命令の呼び出し

メインプログラムのサイクル OB からは、常に USS_Drive_Control, USS_Read_Param または USS_Write_Param のみを呼び出してください。 USS_Port_Scan 命令ファンクションはどの OB からも呼び出せますが、通常はサイクリック割り込み OB から呼び出されます。

USS_Drive_Control, USS_Read_Param または USS_Write_Param 命令を対応する USS_Port_Scan 命令よりも優先度の高い OB で使用しないでください。 たとえば、USS_Port_Scan をメインプログラムに追加したり、USS_Read_Param をサイクリック割り込み OB に追加したりしないでください。USS_Port_Scan の実行が別の命令で中断した場合には、予期しない エラーが発生することがあります。

ドライブとの通信に必要な時間の計算

ドライブとの通信は、CPU のサイクルとは非同期で行われます。 CPU は通常、ドライブとの通信が完了する前に複数のサイクルを実行します。

ドライブに設定されたウォッチドッグがトリガされないように、送信フレームをウォッチドッグタイム内にドライブに送信する必要があります。 通信エラーの発生時にトランザクションを完了するのに必要な試行回数を考慮してください。 既定では、USS プロトコルによるトランザクションごとに 2 回までの試行が行われます。
2 つの送信フレーム間の最大時間間隔は、以下のように計算されます。

N * (5 * サイクルタイム + フレームランタイム + 最大受信フレームのタイムアウト) * (送信試行回数)

N	このネットワーク内のドライブの数
係数 5	フレームの送受信には通常、5 サイクルが必要です。
サイクルタイム	USS_Port_Scan 命令が呼び出されるサイクリック割り込み OB の最大サイクルタイム
フレームのランタイム	フレームタイム = (フレーム当たりの文字数) * (文字当たり 11 Bit) / (Bit/s 単位のデータ伝送速度)
送信試行回数	試行回数+ 1
受信フレームのタイムアウト	RCVTIME(ドライブから応答を受信しない場合)
最大受信フレームのタイムアウト	RCVTIME + MSGTIME(RCVTIME の期限切れの少し前に不完全な応答を受信し、MSGTIME のモニタリングが期限切れの場合、**あるいは** RCVTIME

の期限切れの時に応答がまだ処理中の場合、タイムアウトは MSGTIME によって延長されます)

「フレーム受信タイムアウト」(ミリ秒)では、以下の時間が適用されます。

ビット/秒	115200	57600	38400	19200	9600	4800	2400	1200
Receive_Conditions.END.RCVTIME	25	29	33	56	72	100	100	100
Receive_Conditions.END.MSGTIME	25	29	33	56	72	124	240	460

最大受信フレームのタイムアウト = (Receive_Conditions.END.RCVTIME (0.072 s) + Receive_Conditions.END.MSGTIME (0.072 s))

例:

5 ライブ
データ伝送速度 = 9600 bps
28 文字/フレーム
サイクルタイム = 0.020 s
リトライ回数 = 2

時間間隔 = 5 * (5 * 0.02 s + 1 * 28 * 11 / 9600 s + 0.072 s + 0.072 s) * 3 = 3 s

この場合、ドライブの時間モニタリングは 3 秒に設定する必要があります。

USS_Port_Scan: USS ネットワークによる通信

説明

USS_Port_Scan 命令は、USS ネットワークを使用して、通信処理を行います。 この命令を追加すると、STEP 7 は自動的にインスタンス DB を作成します。

パラメータ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200 /1500	S7-300/ 400/ WinAC		
PORT	IN	Port	Word	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
BAUD	IN	DInt		9600	USS 通信のデータ伝送率 以下が有効です。 • 1200 bps • 2400 bps • 4800 bps • 9600 bps • 19200 bps • 38400 bps • 57600 bps • 115200 bps
USS_DB	INOUT	USS_BASE		–	この USS_DB パラメータは、USS_Drive_Control 命令をユーザープログラムに追加するときに生成され、初期化されるインスタンス DB の(静的)USS_DB パラメータに接続する必要があります。
COM_RST	INOUT	---	Bool	FALSE	USS_Port_Scan 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
ERROR	OUT	Bool		FALSE	TRUE の場合、この出力はエラーが発生したことと、STATUS 出力が有効であることを示します。 USS_Drive_Control 命令のインスタンス DB で静的タグ USS_DB. w_USSExtendedError の値

				をチェックしなければならない場合があります。
STA-TUS	OUT	Word	0	エラーコード(エラーメッセージを参照)。

プログラムには、PtP 通信ポートあたり 1 つずつ USS_Port_Scan 命令があり、この命令の各呼び出しがこのネットワーク内のすべてのドライブとの通信を制御します。 1 つの USS ネットワークと 1 つの PtP 通信ポートに割り当てられたすべての USS ファンクションが、同じインスタンス DB を使用する必要があります。

プログラムは、ドライブ内でのタイムアウトを防止するために十分な回数の USS_Port_Scan 命令を実行する必要があります(USS プロトコル使用の必要条件「ドライブとの通信に必要な時間を計算」を参照)。

通常、USS_Port_Scan 命令はドライブのタイムアウトを防止し、最後の USS データの更新を USS_Drive_Control の呼び出しに使用できるようにするために、サイクリック割り込み OB から呼び出されます。

USS_Port_Scan データブロックタグ

下の表に、プログラムで使用できる USS_Port_Scan のインスタンス DB 内のパブリックな静的タグを示します。

インスタンス DB の静的タグ

タグ	データタイプ	標準	説明
MODE	USInt	4	動作モード 有効な動作モード以下のとおりです。 • 0 = 全二重(RS232) • 1 = 全二重(RS422) 4 線式モード(ポイントツーポイント) • 2 = 全二重(RS422) 4 線式モード(マルチポイントマスタ; CM PtP (ET 200SP)) • 3 = 全二重(RS422) 4 線式モード(マルチポイントスレーブ; CM PtP (ET 200SP)) • 4 = 半二重(RS485) 2 線式モード [1)]
LINE_PRE	USInt	2	回線初期状態の受信 有効な初期状態は以下のとおりです。 • 0 = 初期状態「なし」 [1)] • 1 = シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) この初期状態では、ブレーク検出が可能です。 以下とともにのみ選択できます。 「全二重(RS422) 4 線式モード(ポイントツーポイント接続)」および「全二重(RS422) 4 線式モード(マルチポイントスレーブ)」 • 2 = シグナル R(A) 0 V/シグナル R(B) 5 V この初期設定はアイドル状態に相当します(送信動作がアクティブでない)。 この初期状態では、ブレーク検出は不可能です。
BRK_DET	USInt	0	診断割り込みを有効にします。 • 0 - 有効になっていません • 1 - 有効になっています

RE-TRIES_MAX	SInt/ Byte	2	通信エラーが発生した場合の試行回数。 このパラメータを使用して、応答フレームが設定時間内に受信されなかった場合の要求フレームの送信試行回数を設定できます。
EN_DI-AG_ALARM	Bool	0	診断割り込みを有効にします。 • 0 - 有効になっていません • 1 - 有効になっています
EN_SUP-PLY_VOLT	Bool	0	電源電圧 L+が不明の診断の有効化 • 0 - 有効になっていません • 1 - 有効になっています
[1] RS485 の CM 1241 で PROFIBUS ケーブルを使用するために必要な設定。			

USS_Drive_Control: ドライブ用のデータの準備と表示

説明

命令 USS_Drive_Control はドライブのデータを準備し、ドライブの応答データを評価します。 ドライブごとに個別の命令のインスタンスを使用する必要があります。また、1 つの USS ネットワークと 1 つの PtP 通信ポートに割り当てられたすべての USS ファンクションが、同じインスタンス DB を使用する必要があります。 最初の USS_Drive_Control 命令を追加する時に DB 名を入力する必要があります。 その後、最初の命令の追加時に作成されたこの DB を参照します。

この命令を追加すると、STEP 7 は自動的にこの DB を作成します。

パラメータ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200/1500	S7-300/400/WinAC		
RUN	IN	Bool		FALSE	ドライブの始動ビット: このパラメータが TRUE の場合、この入力を使用して事前に設定された速度でドライブを操作することができます。 ドライブの操作中に RUN が FALSE に切り替わった場合、モータは停止するまで惰性運転します。 この動作は、電源の切断(OFF2)およびモータのブレーキ(OFF3)の場合とは異なります。
OFF2	IN	Bool		FALSE	「停止するまで惰性運転」ビット: このパラメータが FALSE の場合、このビットによってドライブはブレーキをかけずに停止するまで惰性運転します。
OFF3	IN	Bool		FALSE	即時停止ビット: このパラメータが FALSE の場合、このビットによってドライブにブレーキがかけられ、即時停止します。
F_ACK	IN	Bool		FALSE	エラー確認ビット: このビットは、ドライブのエラービットをリセットします。 このビットはエラーのクリア後に設定され、ドライブはそれによって前のエラーを報告する必要がなくなったことを検出します。
DIR	IN	Bool		FALSE	ドライブの方向制御 ドライブを正転する場合 (SPEED_SP が正の場合、表「SPEED_SP パラメータと DIR パラメータの相互関係」を参照してください).はこのビットを設定します
DRIVE	IN	USInt	Byte	1	ドライブのアドレス: この入力は USS ドライブのアドレスです。 有効な範囲はドライブ 1 からドライブ 16 です。
PZD_LEN	IN	USInt	Byte	2	ワード長: これは PZD データワード数です。 有効な値は、2、4、6 または 8 ワードです。
SPEED_SP	IN	Real		0.0	速度セットポイント: これはドライブの速度で、構成定義した周波数のパーセンテージとして表

				されます。 正の値の場合、ドライブは正転します(DIR が真の場合)。 有効な範囲は 200.00 ～-200.00 です。
CTRL3	IN	Word	0	コントロールワード 3: ドライブのユーザー定義パラメータに書き込まれる値。 これは、ドライブで構成定義する必要があります(オプションパラメータ)。
CTRL4	IN	Word	0	コントロールワード 4: ドライブのユーザー定義パラメータに書き込まれる値。 これは、ドライブで構成定義する必要があります(オプションパラメータ)。
CTRL5	IN	Word	0	コントロールワード 5: ドライブのユーザー定義パラメータに書き込まれる値。 これは、ドライブで構成定義する必要があります(オプションパラメータ)。
CTRL6	IN	Word	0	コントロールワード 6: ドライブのユーザー定義パラメータに書き込まれる値。 これは、ドライブで構成定義する必要があります(オプションパラメータ)。
CTRL7	IN	Word	0	コントロールワード 7: ドライブのユーザー定義パラメータに書き込まれる値。 これは、ドライブで構成定義する必要があります(オプションパラメータ)。
CTRL8	IN	Word	0	コントロールワード 8: ドライブのユーザー定義パラメータに書き込まれる値。 これは、ドライブで構成定義する必要があります(オプションパラメータ)。
COM_RST	IN/OUT	--- Bool	FALSE	USS_Drive_Control 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
NDR	OUT	Bool	FALSE	新しいデータが使用可能: このパラメータが真の場合、ビットは新しい通信要求のデータが出力で使用可能であることを通知します。
ERROR	OUT	Bool	FALSE	エラー発生: TRUE の場合、この出力はエラーが発生したことと、STATUS 出力が有効であることを示します。 エラーの場合、他のすべての出力はゼロにセットされます。 通信エラーは、USS_Port_Scan 命令の ERROR および STATUS 出力でのみ通知されます。
STATUS	OUT	Word	0	エラーコード(エラーメッセージを参照)。
RUN_EN	OUT	Bool	FALSE	操作が有効: このビットは、ドライブが稼動中かどうかを通知します。
D_DIR	OUT	Bool	FALSE	ドライブの回転方向: このビットは、ドライブが正転するかどうかを通知します。 • FALSE – 正転 • TRUE – 逆転
INHIBIT	OUT	Bool	FALSE	ドライブブロック済み: このビットは、ドライブのブロックビットのステータスを通知します。 • FALSE – ブロックなし

				• TRUE – ブロック済み
FAULT	OUT	Bool	FALSE	ドライブエラー: このビットは、ドライブでエラーが発生したことを通知します。 異常を解消し、F_ACK ビットを設定してこのビットをクリアする必要があります。
速度	OUT	Real	0.0	ドライブ速度現在値(ドライブの STATUS 2 のスケール値): これはドライブの速度で、構成定義した速度のパーセンテージとして表されます。
STA-TUS1	OUT	Word	0	ドライブの STATUS 1 この値には、ドライブの固定ステータスビットが含まれています。
STA-TUS3	OUT	Word	0	ドライブの STATUS 3 この値には、ドライブのユーザー定義可能なステータスワードが含まれています。
STA-TUS4	OUT	Word	0	ドライブの STATUS 4 この値には、ドライブのユーザー定義可能なステータスワードが含まれています。
STA-TUS5	OUT	Word	0	ドライブの STATUS 5 この値には、ドライブのユーザー定義可能なステータスワードが含まれています。
STA-TUS6	OUT	Word	0	ドライブの STATUS 6 この値には、ドライブのユーザー定義可能なステータスワードが含まれています。
STA-TUS7	OUT	Word	0	ドライブの STATUS 7 この値には、ドライブのユーザー定義可能なステータスワードが含まれています。
STA-TUS8	OUT	Word	0	ドライブの STATUS 8 この値には、ドライブのユーザー定義可能なステータスワードが含まれています。

USS_Drive_Control が最初に実行されるときに、USS アドレス(DRIVE パラメータ)で指定されたドライブがインスタンス DB で初期化されます。 初期化後、後続の USS_Port_Scan 命令でこのドライブ番号からのドライブとの通信を開始することができます。

ドライブ番号を変更した場合、まず CPU を STOP に切り替えた後で RUN に戻し、インスタンス DB を初期化する必要があります。 入力パラメータは USS 送信バッファで構成定義され、すべての出力が「前の」有効な応答バッファから読み出されます。USS_Drive_Control は、送信するデータを構成定義し、前の要求で受信されたデータの評価のみを行います。

D_IR 入力(Bool)、または、SPEED_SP 入力(Real)の符号(正または負)を使用して、ドライブの回転方向を制御できます。 モータが正転する場合、これらの入力が共に作動してドライブの回転方向を決定する方法を下の表に示します。

SPEED_SP パラメータと DIR パラメータの相互関係

SPEED_SP	DIR	ドライブの回転方向
値 > 0	0	逆転

値 > 0	1	正転
値< 0	0	正転
値< 0	1	逆転

USS_Drive_Control データブロックタグ

下の表に、プログラムで使用できる USS_Drive_Control のインスタンス DB 内のパブリックな静的タグを示します。

インスタンス DB の静的タグ

タグ	データタイプ	標準	説明
USS_DB. w_USSExten- dedError	Word	16#0	USS ドライブ拡張エラーコード - ドライブ固有値 エラーメッセージの意味は、**最初に**エラーを報告した命令によって異なります(ERROR = TRUE)。 以下のケースが区別されます。 • USS_Write_Param: ドライブの説明にエラーコードの意味が記載されています。 • USS_Read_Param: ドライブの説明にエラーコードの意味が記載されています。 • USS_Port_Scan: エラーメッセージにより影響を受けるドライブ数。

USS_Read_Param: データをドライブから読み出す

説明

「USS_Read_Param」命令はドライブからパラメータを読み出します。 1 つの USS ネットワークと 1 つの PtP 通信ポートに割り当てられたすべての USS ファンクションは、USS_Drive_Control 命令のインスタンスデータブロックを使用する必要があります。USS_Read_Param は、メインプログラムのサイクル OB から呼び出す必要があります。

パラメータ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200/1500	S7-300/400 / WinAC		
REQ	IN	Bool		–	REQ の立ち上がりエッジで新しい読み出し要求が作成されます。
DRIVE	IN	USInt	Byte	–	ドライブのアドレス: DRIVE は USS ドライブのアドレスです。有効な範囲はドライブ 1 からドライブ 16 です。
PARAM	IN	UInt		–	パラメータ番号 PARAM では、書き込むドライブのパラメータが指定されます。このパラメータの範囲は 0～2047 です。ドライブの中には、INDEX パラメータの最上位バイトで 2047 より大きいパラメータ値にアクセスできるものがあります。拡張範囲へのアクセスに関する追加情報は、ユーザーのドライブマニュアルを参照してください。
INDEX	IN	UInt		–	パラメータインデックス:INDEX では、書き込むドライブパラメータインデックスが指定されます。これは 16 ビットの値で、最下位ビットは範囲(0～255)の実際のインデックス値です。ドライブは、ドライブ固有の最上位バイトを使用することもできます。 追加情報については、ドライブのマニュアルを参照してください。
USS_DB	INOUT	USS_BASE		–	この USS_DB パラメータは、USS_Drive_Control 命令をユーザープログラムに追加するときに生成され、初期化されるインスタンス DB の(静的)USS_DB パラメータに接続する必要があります。
DONE[1]	OUT	Bool		FALSE	このパラメータが TRUE の場合、直前に要求された読み出しパラメータの値は VALUE 出力で入手できます。 USS_Drive_Control 命令がドライブの読み出し応答を認識する時に、このビットがセットされます。 このビットは、USS_Read_Param が呼び出された時にリセットされます。

ERROR	OUT	Bool	FALSE	ERROR = TRUE: エラーが発生し、STATUS 出力が有効です。 エラーの場合、他のすべての出力はゼロにセットされます。 通信エラーは、USS_Port_Scan 命令の ERROR および STATUS 出力でのみ通知されます。 USS_Drive_Control 命令のインスタンス DB で静的タグ USS_DB. w_USSExtendedError の値をチェックしなければならない場合があります。
STATUS	OUT	Word	0	エラーコード(エラーメッセージを参照)。
VALUE	OUT	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	これは読み出されたパラメータの値で、DONE ビットの値が真の場合のみ有効です。
[1] DONE ビットは、有効なデータが参照されたモータドライブから読み出されて、CPU に伝送されたことを示します。 それは、この命令が追加のパラメータを直ちに読み出すことができることを示してはいません。 個々のドライブで使用するためにパラメータチャンネルを開放する前に、空の読み出し要求をモータドライブに送信し命令によって確認応答される必要があります。特定のモータドライブに対して USS_Read_Param または USS_Write_Param を即時に呼び出すと、エラー 16#818A が発生します。				

USS_Write_Param: ドライブのデータを変更

注記

EEPROM 書き込み命令の場合(USS ドライブ内の EEPROM):

EEPROM のサービス寿命を最大限にするために、EEPROM の書き込み操作回数を最小限に抑えてください。

説明

「USS_Write_Param」命令はドライブのパラメータを変更します。 1 つの USS ネットワークと 1 つの PtP 通信ポートに割り当てられたすべての USS ファンクションが、USS_Drive_Control のインスタンスデータブロックを使用する必要があります。

「USS_Write_Param」はメインプログラムのサイクル OB から呼び出す必要があります。

パラメータ

パラメータのデータタイプ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200/1500	S7-300/400/WinAC		
REQ	IN	Bool		–	REQ の立ち上がり エッジで新しい書き込み要求が作成されます。
DRIVE	IN	USInt	Byte	–	ドライブのアドレス: DRIVE は USS ドライブのアドレスです。有効な範囲はドライブ 1 からドライブ 16 です。
PARAM	IN	UInt		–	パラメータ番号 PARAM では、書き込むドライブのパラメータが指定されます。このパラメータの範囲は 0～2047 です。ドライブの中には、INDEX パラメータの最上位バイトで 2047 より大きいパラメータ値にアクセスできるものがあります。拡張範囲へのアクセスに関する追加情報は、ユーザーのドライブマニュアルを参照してください。
INDEX	IN	UInt		–	パラメータインデックス:INDEX では、書き込むドライブパラメータインデックスが指定されます。これは 16 ビットの値で、最下位ビットは範囲(0～255)の実際のインデックス値です。ドライブは、ドライブ固有の最上位バイトを使用することもできます。 追加情報については、ドライブのマニュアルを参照してください。
EE-PROM	IN	Bool		–	ドライブの EEPROM に保存: TRUE の場合、ドライブへの書き込み用パラメータのトランザクションがドライブの EEPROM に保存されます。 FALSE の場合、書き込まれた値が一時的にのみ保存され、次にドライブの電源を入れた時に失われます。

VALUE	IN	Variant (Word, Int, UInt, DWord, DInt, UDInt, Real)	Any (Word, Int, UInt, DWord, DInt, UDInt, Real)	–	書き込み先のパラメータの値。 REQ の立ち上がり エッジで有効になる必要があります。
USS_DB	INOUT	USS_BASE		–	この USS_DB パラメータは、USS_Drive_Control 命令をユーザープログラムに追加するときに生成され、初期化されるインスタンス DB の(静的)USS_DB パラメータに接続する必要があります。
DONE[1]	OUT	Bool		FALSE	TRUE の場合、VALUE 入力がドライブに書き込まれます。 USS_Drive_Control 命令がドライブの書き込み応答を認識する時に、このビットがセットされます。 このビットは、USS_Write_Param が呼び出された時にリセットされます。
ERROR	OUT	Bool		FALSE	TRUE の場合、エラーが発生し、STATUS 出力が有効です。 エラーの場合、他のすべての出力はゼロにセットされます。 通信エラーは、USS_Port_Scan 命令の ERROR および STATUS 出力でのみ通知されます。 USS_Drive_Control 命令のインスタンス DB で静的タグ USS_DB. w_USSExtendedError の値をチェックしなければならない場合があります。
STATUS	OUT	Word		0	エラーコード(エラーメッセージを参照)。

[1] DONE ビットは、有効なデータが参照されたモータドライブから読み出されて、CPU に伝送されたことを示します。 USS ライブラリが追加パラメータを即時読み出せることを示してはいません。個々のドライブで使用するためにパラメータチャンネルを開放する前に、空の書き込み要求をモータドライブに送信し命令によって確認応答される必要があります。 特定のモータドライブに対して USS_Read_Param または USS_Write_Param FC を即時に呼び出すと、エラー 0x818A が発生します。

ドライブのセットアップについての一般情報

ドライブセットアップの要件

- ドライブに対して 4 つの PIV ワード(ParameterIDValue)の使用を設定する必要があります。
- ドライブが 2、4、6、または 8 つの PZD ワード(process data area (プロセスデータ領域))に対して設定可能であること。
- ドライブの PZD ワード数が、ドライブの USS_Drive_Control 命令の PZD_LEN 入力に対応していること。
- すべてのドライブのデータ伝送速度が、USS_Port_Scan 命令の BAUD 入力に必ず対応していること。
- ドライブが、必ず USS 通信用にセットアップ済みであること。
- 周波数セットポイントが USS インターフェースによって提供されることが、ドライブで必ず指定されていること。
- ドライブアドレスが必ず指定されていること(領域: 1-16).
  このアドレスは、ドライブの USS_Drive_Control ブロックの DRIVE 入力に対応している必要があります。
- RS485 ネットワークが正しく終了すること。

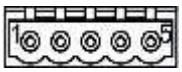
SINAMICS V20 ドライブの接続とセットアップ

S7-1200 での SINAMICS V20 の動作の応用例が、インターネットで入手できます。

SINAMICS V20 ドライブの接続

USS ネットワークへの SIEMENS G120(C)ドライブの接続例。 他のドライブの接続例は、それぞれのドライブのマニュアルを参照してください。

USS ネットワークへの SINAMICS G120(C)ドライブの接続は、差し込み式接続によって行われます。この接続は短絡耐性を持ち、絶縁されています。



1	0 V 基準電位
2	RS485N、受信および送信(-)
3	RS485N、受信および送信(+)
4	ケーブルシールド
5	未使用

USS 接続

通知

互いに異なる基準電圧

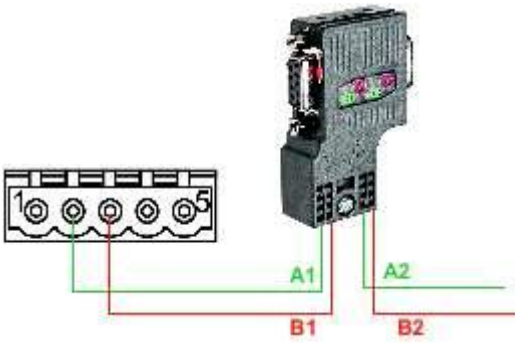
基準電圧が同じでないデバイスを接続すると、接続ケーブルに不要な電流が生じることがあります。 この不要な電流により通信エラーが発生したり、デバイスが損傷することがあります。

通信ケーブルで接続するすべてのデバイスが回路内に同じ基準コンダクタを持っているか、不要な電流の発生を防止するために電気的に切り離されていることを確認してください。

必ず、シールドが接地されるか、またはドライブのバスコネクタのピン 1 に接続されるようにしてください。

必ず、G120(C)の配線端子 2 (GND)が接地されるようにしてください。

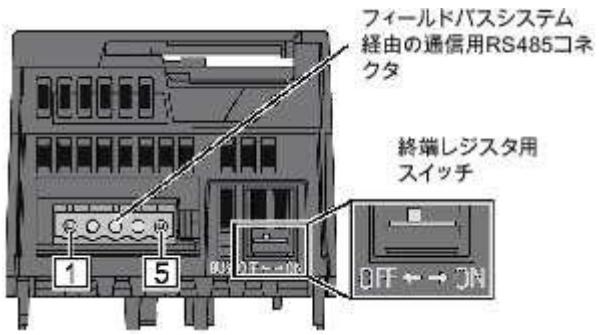
RS485 マスタ(たとえば、CM1241 通信モジュール付きの S7-1200 CPU)を PROFIBUS コネクタによって接続する場合は、以下に示すようにバスケーブルを配線してください。



通信モジュールの接続

RS485 マスタがネットワークまたはポイントツーポイント接続の終端ステーションの場合、端子 A1 および B1 は終端設定用であるため(たとえば、DP プラグコネクタ 6ES7972-0BB52-0XA0 を使用)、PROFIBUS コネクタの端子 A1 および B1 (A2 および B2 ではなく)を使用する必要があります。

G120(C)がネットワークの終端ステーションとして設定される場合は、バス終端レジスタ用のスイッチを「ON」に設定する必要があります。



終端ステーションの接続

G120(C)ドライブのセットアップ

ドライブを S7-1500 または ET 200SP に接続する前に、ドライブに以下のシステムパラメータがあることを確認します。

ステップ	命令	取扱説明書	
		G120 [1]	G120C [2]
1	オペレータパネル BOP-2 を使用して、ドライブの基本コミッショニングを実行します。	章 4.4.3	章 6.4.1

	このコンバータは、その入力/出力およびフィールドバスインターフェース用のさまざまな既定値(マクロ)を提供します。 基本コミッショニングの 9 番目のステップ(MAC PAR p15)で、USS 通信用のマクロ 21 を選択します。 これによって、以下のパラメータの既定値が決まります。 • データ伝送速度(p2020):38400 bps • PZD の数(p2022): 2 • PIV の数(p2023):variable 注記: STARTER コミッショニングソフトウェアまたは SINAMICS Startdrive を使用して、基本コミッショニングを行うこともできます。		
2	G120 の制御ユニットまたは G120(C)のアドレススイッチを使用して、コンバータの USS アドレスを指定します。 • 有効なアドレス範囲 1 ~ 30 注記: パラメータ p2021、または、STARTER/SINAMICS Startdrive を使用して、USS アドレスを設定することもできます。	章 6.2.2.1	章 8.4.2.1
	以下のステップでは、BOP-2 を使用してパラメータの数を入力したり、値を変更することによって、直接にパラメータにアクセスしています。	章 4.4.2	章 6.4.2
3	ユーザーアプリケーションの以下の通信関連コンバータパラメータを調整します。 • データ伝送速度(p2020)、38400 bps でない場合 (設定が、USS_Port_Scan 通信命令の BAUD パラメータと同一であることを確認します。) • PZD の数(p2022)、2 でない場合 (設定が、USS_Drive_Control 通信命令の PZD_LEN パラメータと同一であることを確認します。) • PIV の数(p2023) = 4 (既定でマクロ 21 によって「variable」(127)に設定された値を「4」に変更します(命令 USS_Read_Param および USS_Write_Param で必要)) • フィールドバス SS モニタ時間[ms] (p2040)	章 6.2.2.2	章 8.4.2.1
4	速度セットポイントのソースを指定します。 • n_set Eval (p1000[0]) = 6 (速度セットポイントは、USS バスによって提供されます。)		
5	速度および周波数の基準値を設定します。 • n_reference f_reference (p2000) = (6,00 分$^{-1}$ ~ 210000,00 分$^{-1}$) (すべての相対速度または周波数はこの基準値を参照します。) この基準値は、100%、$4000_{16\ 進}$(ワード)、または $4000\ 0000_{16\ 進}$(ダブルワード)に対応します。以下が当てはまります。 $f_{基準値}$ (Hz 単位) = $n_{基準値}$ (((分$^{-1}$) / 60) x ポールペアの数)単位)		
6	これらのパラメータは不揮発性メモリに転送します。 • Save par (p0971) = 1		

1) G120

2) G120(C)

エラーメッセージ

エラーメッセージの概要 - PtP

エラーメッセージは命令の STATUS 出力で提供され、そこで評価したりユーザープログラムで処理したりすることができます。

エラーコード	説明	対策
16#0000	エラーはありません。	-
受信ステータスとエラーコード		
16#0094	「固定/最大フレーム長の受信」に基づいて識別されたフレームの終了	-
16#0095	「メッセージタイムアウト」に基づいて識別されたフレームの終了	-
16#0096	「キャラクタ遅延時間」の経過に基づいて識別されたフレームの終了	-
16#0097	最大応答時間に達したためにフレームが中止されました。	-
16#0098	「メッセージからのメッセージ長の読み出し」条件の適合に基づいて識別されたフレームの終了	-
16#0099	「終了シーケンス」の受信に基づいて識別されたフレームの終了	-
送信ステータスとエラーコード		
16#7000	ブロックアイドル	-
16#7001	新しいフレームの最初の呼び出し データ伝送が開始されました	-
16#7002	中間呼び出し: データ伝送が実行中	-
16#8085	長さが無効	適切なフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8088	指定された長さが受信バッファの設定された範囲を超えています。 注記: データタイプ STRING が BUFFER パラメータで指定されている場合、現在の文字列が LENGTH パラメータで指定された長さよりも短いと、このエラーコードも表示されます。	受信バッファの範囲を変更するか、または受信バッファの設定された範囲に対応するフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8090	構成エラー: WString のバイト数が奇数です	偶数のバイト数を選択します。
受信ステータスとエラーコード		
16#8088	受信した文字数が BUFFER パラメータで指定された文字数を超えています。	適切なフレーム長を選択します。

		以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8090	構成エラー: WString のバイト数が奇数です	偶数のバイト数を選択します。
特殊ファンクションのエラーメッセージコード		
16#818F	不正なパラメータ番号の設定(USS のみ)	適切なパラメータ番号(PARAM)を選択します。 次の番号が有効です。 0-2047
16#8190	CRC 計算の不正な設定	CRC 計算の適切な値を選択します。 以下が有効です。 無効または有効。 アドレス指定されたモジュールが CRC 計算をサポートしているかどうかチェックします。
16#8191	診断エラー割り込みの不正な設定	[診断エラー割り込み]の適切な値を選択します。 以下が有効です。 診断エラー割り込みが無効、または診断エラー割り込みが有効。 アドレス指定されたモジュールが診断割り込みの生成をサポートしているかどうかをチェックします。
「ポートの構成」のエラーメッセージコード		
16#81A0	モジュールがこのプロトコルをサポートしていません。	モジュールの有効なプロトコル(PROTOCOL)を選択します。
16#81A1	モジュールがこのデータ伝送率をサポートしていません。	モジュールの有効なデータ伝送速度(BAUD)を選択します。
16#81A2	モジュールがこのパリティ設定をサポートしていません。	「パリティ」(PARITY)の適切な値を選択します。 以下が有効です。 • なし(1) • 偶数(2) • 奇数(3) • マーク(4) • スペース(5) • 任意(6)
16#81A3	モジュールがこのデータビット数をサポートしていません。	「データビット数」(DATABITS)の適切な値を選択します。 以下が有効です。 • 7 (2) • 8 (1)
16#81A4	モジュールがこのストップビット数をサポートしていません。	「ストップビット数」(STOPBITS)の適切な値を選択します。 以下が有効です。 • 1 (1) • 2 (2)

16#81A5	モジュールがこのタイプのデータフロー制御をサポートしていません。	モジュールの有効なデータフロー制御(FLOWCTRL)を選択します。
16#81A7	XON または XOFF の無効な値	XON (XONCHAR)および XOFF (XOFF-CHAR)の適切な値を選択します。 値の有効な範囲: 0...255
16#81AA	動作モードが無効	有効な動作モード以下のとおりです。 • 全二重(RS232) (0) • 全二重(RS422) 4 線式モード(ポイントツーポイント) (1)/ (CM PtP (ET 200SP)) • 全二重(RS422) 4 線式モード(マルチポイントマスタ) (2)/ (CM PtP (ET 200SP)) • 全二重(RS 422) 4 線式モード(マルチポイントスレーブ) (3) • 半二重(RS485) 2 線式モード (4)
16#81AB	回線初期状態が無効	有効な初期状態は以下のとおりです。 • 初期設定[なし] (0) • シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) (1) 以下とともにのみ選択できます。「全二重(RS422) 4 線式モード(ポイントツーポイント接続)」および「全二重(RS422) 4 線式モード(マルチポイントスレーブ)」 • シグナル R(A) 0 V/シグナル R(B) 5 V (2): この初期設定はアイドル状態に相当します(送信動作がアクティブでない)。
16#81AC	[ブレーク検出]の値が無効	[ブレーク検出]の適切な値を選択します。以下が有効です。 • ブレーク検出が無効(0) • ブレーク検出が有効(1)
16#81AF	モジュールがこのプロトコルをサポートしていません。	モジュールの有効なプロトコルを選択します。
「送信構成」のエラーコード		
16#81B5	エンドデリミタが 2 つ以上、または終了シーケンスが 5 文字以上	[エンドデリミタ]と[終了シーケンス]に適切な値を選択します。 以下が有効です。 • 無効(0)、 • 1 つ (1)または 2 つ (2)のエンドデリミタ または • 無効(0)、 • 終了シーケンスに 1 文字 (1)から 5 文字 (5)
16#81B6	3964(R)プロトコルが選択されていたため、送信構成が拒否されました。	3964(R)プロトコルが設定されている場合は、送信構成が送信されないようにします。

「受信構成」のエラーコード		
16#81C0	開始条件が無効	適切な開始条件を選択します。 以下が有効です。 • フレーム開始前の送信中断 • アイドルラインの送信
16#81C1	終了条件が無効、または終了条件が選択されていません	適切な条件を選択します(Freeport によるデータ受信を参照)。
16#81C3	[最大メッセージ長]の値が無効	「最大メッセージ長」(MAXLEN)の適切な値を選択します。 値の有効な範囲(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#81C4	[メッセージ内の長さ指定のオフセット]の値が無効	[メッセージ内の長さ指定のオフセット]に有効な値を選択します。 値の有効な範囲(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#81C5	[長さフィールドのサイズ]の値が無効	「長さフィールドのサイズ」(LENGTHSIZE)の適切な値を選択します。 値の有効な範囲(バイト単位): • 1 (1) • 2 (2) • 4 (4)
16#81C6	[長さ指定でカウントされていない文字数]の値が無効	「長さ指定でカウントされていない文字数」(LENGTHM)の適切な値を選択します。 値の有効な範囲: 0～255 (バイト)
16#81C7	「メッセージ内のオフセット + 長さフィールドのサイズ + カウントされていない文字数」の合計が最大フレーム長を超えています	[メッセージ内のオフセット]、[長さフィールドのサイズ]、および[カウントされていない文字数]に適切な値を選択します。 値の有効な範囲: • メッセージ内のオフセット(モジュールによって異なります): 0-1024/2048/4096 (Byte) • 長さフィールドのサイズ: 1、2、または 4 (バイト) • カウントされていない文字数: 0～255 (バイト)
16#81C8	[応答タイムアウト]の値が無効	[応答タイムアウト]に適切な値を選択します。 値の有効な範囲: 1-65535 (ms)
16#81C9	[キャラクタ遅延時間]の値が無効	[キャラクタ遅延時間]に適切な値を選択します。 値の有効な範囲: 1～65535 (ビット時間)
16#81CB	フレーム終了シーケンスが有効になっていますが、チェック用の文字が有効になっていません	チェック用に 1 つ以上の文字を有効にします。

16#81CC	フレーム開始シーケンスが有効になっていますが、チェック用の文字が有効になっていません	チェック用に 1 つ以上の文字を有効にします。
16#81CD	[上書き禁止]の値が無効	[上書き禁止]に適切な値を選択します。 以下が有効です。 • 上書き禁止が無効(0)または • 上書き禁止が有効(1)
16#81CE	[スタートアップ時に受信バッファをクリアする]の値が無効	[スタートアップ時に受信バッファをクリアする]に適切な値を選択します。 以下が有効です。 • スタートアップ時に受信バッファをクリアが無効(0) • スタートアップ時に受信バッファをクリアが無効(1)
送信ステータスとエラーコード		
16#81D0	送信コマンドのランタイム中の送信要求の受信	送信コマンドのランタイム中に追加の送信要求を受信するのを防止します。
16#81D1	XON または CTS = ON の待機時間が経過しました。	通信パートナーで障害が発生したか、速度が遅すぎるか、またはオフラインになっています。 通信パートナーをチェックするか、または必要に応じてパラメータを変更します。
16#81D2	「ハードウェア RTS は常に ON」: DSR = ON から OFF への切り替えのために送信ジョブがキャンセルされました	通信パートナーをチェックします。 伝送時間全体にわたって、DSR が確実に ON になるようにします。
16#81D3	送信バッファオーバーフロー/送信フレームが長すぎます	短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1～1024/2048/4096 (バイト)
16#81D5	パラメータ変更、断線の検出、または CPU の STOP によって、伝送がキャンセルされました	パラメータ割り当て、断線、および CPU ステータスをチェックします。
16#81D6	終了識別子が受信されなかったため、伝送がキャンセルされました。	終了文字のパラメータ割り当てと通信パートナーのフレームをチェックします。
16#81D7	ユーザープログラムとモジュール間の通信エラー	通信をチェックします(たとえば、シーケンス番号の一致など)。
16#81D8	モジュールが構成定義されていないため、伝送の試行が拒否されました。	モジュールを構成定義します。
受信構成のエラーコード		
16#81E0	フレームが中断されました: 送信バッファオーバーフロー/送信フレームが長すぎます	ユーザープログラムで受信ファンクションの呼び出し率を上げるか、またはデータフロー制御による通信を構成定義する必要があります。
16#81E1	フレームが中断されました: パリティエラー	通信パートナーの接続線をチェックするか、または両方のデバイスが同じデータ伝送率、パリティ、ストップビット数に構成定義されているかどうかを検証します。

16#81E2	フレームが中断されました: キャラクタフレームエラー	スタートビット、データビット、パリティビット、データ伝送率、ストップビットの設定をチェックします。
16#81E3	フレームが中断されました: キャラクタオーバーフロエラー	ファームウェアエラー: カスタマーサポートに問い合わせてください。
16#81E4	フレームが中断されました: 「メッセージ内のオフセット + 長さフィールドのサイズ + カウントされていない文字数」の合計長が受信バッファを超えています	メッセージ内のオフセット、長さフィールドのサイズ、およびカウントされていない文字数に適切な値を選択します。
16#81E5	フレームが中断されました: break	パートナーへの受信ラインが中断しています。 再接続するか、またはパートナーのスイッチをオンにします。
16#81E6	[バッファリングされる受信フレーム]の最大数を超過しました	ユーザープログラムで命令を呼び出す回数を増やすか、データフロー制御による通信を構成定義するか、またはバッファリングされるフレームの数を増やします。
16#81E7	同期エラーモジュールと Receive_P2P	Receive_P2P のさまざまな異なるインスタンスが確実に同一モジュールにアクセスしないようにします。
16#81E8	フレームが中断されました: メッセージまたは条件が検出される前に、キャラクタ遅延時間が経過しました。	パートナーデバイスに障害が発生しているか、またはパートナーデバイスが遅すぎます。 必要に応じてインターフェーステスターを使用し、パートナーデバイスが伝送線で相互接続されていることをチェックします。
16#81E9	Modbus CRC エラー(Modbus をサポートしている通信モジュールのみ)	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#81EA	Modbus フレームが短すぎます(Modbus をサポートしている通信モジュールのみ)	Modbus フレームの最小長さが満たされていません 通信パートナーをチェックします。
16#81EB	フレームが中断されました: 最大フレーム長に達しました	通信パートナー側で短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte) フレーム終了検出のパラメータをチェックします。
エラーコード V24 に付随する信号		
16#81F0	モジュールが V24 に付随する信号をサポートしていません。	V24 に付随する信号をサポートしていないモジュールに対して付随する信号の設定を試みました。 必ず RS232 モジュールにするか、または RS232 モード(ET 200SP)を設定します。
16#81F1	V24 に付随する信号を操作できません	ハードウェアデータフロー制御が有効の場合、V24 に付随する信号を手動で操作することはできません。
受信構成のエラーコード		
16#8201 1)	Receive_Conditions は無効なデータタイプへのポインタです。	データタイプ

		DB、BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、DATE、TIME_OF_DAY、TIME、S5TIME、DATE_AND_TIME、STRING の 1 つを指すポインタを入力します。
16#8225	Receive_Conditions が、1 KB 以上の最適化されたメモリ領域をポイントしています。 または Receive_Conditions が最適化されたメモリ領域をポイントしており、受信長が Receive_Conditions でアドレス指定された領域を越えています。	以下の最大長を持つ領域へのポインタを入力します。 • 最適化されたメモリ領域: 1 kByte • 最適化されていない領域: 4 kByte 注記: ポインタが最適化されたメモリ領域を指す場合、1 kByte を超えるバイト数を送信してはいけません。
16#8229 1)	Receive_Conditions が、ビット数が n * 8 と等しくない BOOL へのポインタです。	BOOL へのポインタを使用している場合、ビット数は 8 の倍数とする必要があります。
エラーコード、一般		
16#8280	モジュールの読み出し時の否定確認	エラーの原因の詳細については、RDREC.STATUS 静的パラメータと、SFB RDREC の説明を参照してください。 • PORT パラメータの入力をチェックします。 • 初期呼び出しの前に、COM_RST パラメータを設定します。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、WRREC.STATUS 静的パラメータと、SFB WRREC の説明を参照してください。
16#8282	モジュールが使用不可	PORT パラメータの入力をチェックし、モジュールに確実にアクセスできるようにします。
受信構成のエラーコード		
16#82C1	[バッファリングされた受信フレーム]の値が無効	[バッファリングされた受信フレーム]に適切な値を選択します。 値の有効な範囲: 1-255
16#82C2	3964(R)プロトコルが選択されていたため、受信構成が拒否されました。	3964(R)プロトコルが設定されている場合は、受信構成が送信されないようにします。
16#8301 1)	Receive_Conditions は無効なデータタイプへのポインタです。	有効なデータタイプを選択します。 以下が有効です。 DB、BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、DATE、TIME_OF_DAY、TIME、S5TIME、DATE_AND_TIME、STRING
16#8322	パラメータ読み出し時の範囲長さエラー	Receive_Conditions パラメータの入力をチェックします。
16#8324	パラメータ読み出し時の範囲エラー	Receive_Conditions パラメータの入力をチェックします。
16#8328	パラメータ読み出し時の設定エラー	Receive_Conditions パラメータの入力をチェックします。

送信ステータスとエラーコード		
16#8328 1)	BUFFER が、ビット数が n * 8 と等しくない BOOL へのポインタです	BOOL へのポインタを使用している場合、ビット数は 8 の倍数とする必要があります。
受信構成のエラーコード		
16#8332	Receive_Conditions パラメータの無効なデータブロック	Receive_Conditions パラメータの入力をチェックします。
16#833A	Receive_Conditions パラメータのデータブロックの名称が、ロードされていないデータブロックを示しています。	Receive_Conditions パラメータの入力をチェックします。
16#8351	無効なデータタイプ	Receive_Conditions パラメータの入力をチェックします。
16#8352 1)	Receive_Conditions がデータブロックをポイントしていません。	Receive_Conditions へのポインタをチェックします。
16#8353 1)	Receive_Conditions タイプの構造をポイントしていません。 Receive_Conditions	Receive_Conditions へのポインタをチェックします。
エラーコード 3964(R)プロトコル		
16#8380	パラメータ割り当てエラー: [キャラクタ遅延時間]の値が無効	[キャラクタ遅延時間](CharacterDelay-Time)に適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8381	パラメータ割り当てエラー: [応答タイムアウト]の値が無効	[応答タイムアウト](AcknDelayTime)に適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8382	パラメータ割り当てエラー: [優先度]の値が無効	[優先度](Priority)に適切な値を選択します。 以下が有効です。 • 高い(1) • 低い(0)
16#8383	パラメータ割り当てエラー: [ブロックチェック]の値が無効	[ブロックチェック] (BCC)の適切な値を選択します。 以下が有効です。 • ブロックチェックあり(1) • ブロックチェックなし(0)
16#8384	パラメータ割り当てエラー: [接続試行]の値が無効.	[接続試行](BuildupAttempts)に適切な値を選択します。 値の有効な範囲: 1-255
16#8385	パラメータ割り当てエラー: [送信試行]の値が無効.	[送信試行](RepetitionAttempts)に適切な値を選択します。 値の有効な範囲: 1-255
16#8386	ランタイムエラー: 接続試行回数を超過しました	インターフェースケーブルおよび伝送パラメータをチェックします。 さらに、パートナーデバイス側で受信ファンクションが正しく構成定義されているかどうかもチェックします。

16#8387	ランタイムエラー: 送信試行回数を超過しました	インターフェースケーブル、伝送パラメータ、および通信パートナーの構成をチェックします。
16#8388	ランタイムエラー:「ブロックチェックキャラクタ」のエラー 内部的に計算したブロックチェックキャラクタの値が、接続の終了時にパートナーが受信したブロックチェックキャラクタに対応していません。	接続が致命的に中断されていないかどうかチェックします。その場合は、こまめにエラーコードをチェックすることをお勧めします。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#8389	ランタイムエラー: 空き受信バッファの待機中に無効な文字が受信されました	通信パートナーの送信要求(STX, 02H)は、受信バッファが空いている場合のみ、DLE で応答します。 この前に、追加文字を受信することはできません(再び、STX を除いて)。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838A	ランタイムエラー: 受信時の論理エラー DLE 受信後、追加のランダム文字(DLE または ETX を除く)を受信しました。	フレームヘッダー内およびデータ文字列内のパートナー DLE が常に重複しているか、または接続が DLE ETX でリリースされているかをチェックします。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838B	ランタイムエラー: キャラクタ遅延時間を超過しました	パートナーデバイスが遅すぎるか、またはパートナーデバイスに障害が発生しています。 必要に応じて、伝送線内に接続されたインターフェーステストデバイスを使用して検証します。
16#838C	ランタイムエラー: 空き受信バッファの待機時間が開始されました	ユーザープログラムで命令を呼び出す回数を増やすか、またはデータフロー制御による通信を構成定義します。
16#838D	ランタイムエラー: フレームの繰り返しが NAK 後、4 秒以内に開始されません。	通信パートナーをチェックします。 破壊されている可能性のある受信フレームをパートナーが 4 秒以内に繰り返す必要があります。
16#838E	ランタイムエラー: アイドルモードで、1 つ以上の文字(NAK、STX 以外)を受信しました。	パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838F	ランタイムエラー: 初期化の競合 - 両方のパートナーが高い優先度を設定しています。	パートナーのどちらかで、「低い」優先度を設定します。
16#8391	パラメータ割り当てエラー: Freeport が設定されているため、3964 構成データが拒否されました	Freeport プロトコルが設定されている場合は、3964 パラメータ割り当てデータが送信されないようにします。
[1] S7-300/400 CPU 用の命令の場合のみ		

エラーメッセージの概要 - Modbus

エラーコード	説明	対策
16#0000	エラーはありません。	-
インターフェースの構成エラー - Modbus_Comm_Load		
16#8181	モジュールがこのデータ伝送率をサポートしていません。	BAUD パラメータで、モジュールの有効なデータ伝送速度を選択します。
16#8182	モジュールがこのパリティ設定をサポートしていません。	PARITY パラメータの「パリティ」の適切な値を選択します。 以下が有効です。 • なし(1) • 偶数(2) • 奇数(3) • マーク(4) • スペース(5) • 任意(6)
16#8183	モジュールがこのタイプのデータフロー制御をサポートしていません。	FLOW_CTRL パラメータで、モジュールの有効なデータフロー制御を選択します。
16#8184	[応答タイムアウト]の値が無効	RESP_TO パラメータで、[応答タイムアウト]の適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、Send_Config.RDREC.STATUS/ Receive_Config.RDREC.STATUS 静的パラメータまたは RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、Send_Config.WRREC.STATUS/ Receive_Config.WRREC.STATUS 静的パラメータまたは WRREC.STATUS と、SFB WRREC の説明を参照してください。
16#8282	モジュールが使用不可	PORT パラメータの入力をチェックし、モジュールに確実にアクセスできるようにします。
構成エラー - Modbus_Slave		
16#8186	スレーブアドレスが無効	MB_ADDR パラメータで、適切なスレーブアドレスを選択します。 以下が有効です。 標準アドレス領域では 1～247; 拡張アドレス領域では 1～65535

		(0 はブロードキャスト用に予約済みです)
16#8187	MB_HOLD_REG パラメータの無効な値	MB_HOLD_REG パラメータで、保持レジスタの適切な値を選択します。
16#8188	動作モードまたはブロードキャストが無効(MB_ADDR = 0)および MODE パラメータ ≠ 1	ブロードキャストモードで MODE の値 1 を選択するか、または 異なる動作モードを選択します。
16#818C	MB_HOLD_REG 領域へのポインタは、データブロックまたはビットメモリアドレス領域であることが必要です。	MB_HOLD_REG 領域へのポインタの適切な値を選択します。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.RDREC.STATUS または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.WRREC.STATUS または Receive_P2P.WRREC.STATUS と、SFB WRREC の説明を参照してください。
16#8452 1)	MB_HOLD_REG が、DB またはビットメモリ領域へのポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8453 1)	MB_HOLD_REG が、タイプ BOOL または WORD のポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8454 1)	MB_HOLD_REG でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が読み出しまたは書き込みされるデータバイトの数に対して小さすぎます。	MB_HOLD_REG ポインタをチェックします
16#8455 1)	MB_HOLD_REG が書き込み禁止 DB を指しています	MB_HOLD_REG ポインタをチェックします
16#8456 1)	命令実行中のエラー エラーの原因は、STATUS パラメータに示されます。	SFCSTATUS パラメータの値を識別します。 SFC51、STATUS パラメータの説明でこの値が意味することをチェックします。
構成エラー - Modbus_Master		
16#8180	MB_DB パラメータの値が無効	Modbus_Comm_Load 命令で MB_DB(インスタンスデータ DB)に構成定義された値が有効ではありません。 Modbus_Comm_Load 命令とそのエラーメッセージの相互接続をチェックします。
16#8186	ステーションアドレスが無効	MB_ADDR パラメータで、適切なステーションアドレスを選択します。 以下が有効です。 標準アドレス領域では 1～247; 拡張アドレス領域では 1～65535 (0 はブロードキャスト用に予約済みです)

16#8188	動作モードまたはブロードキャストが無効(MB_ADDR = 0)および MODE パラメータ ≠ 1	ブロードキャストモードで MODE の値 1 を選択するか、または異なる動作モードを選択します。
16#8189	データアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。 情報システムの説明 Modbus_Master を選択します。
16#818A	長さが無効	DATA_LEN パラメータで、適切なデータ長を選択します。 情報システムの説明 Modbus_Master を選択します。
16#818B	DATA_PTR の値が無効	DATA_PTR パラメータで、データポインタの適切な値を選択します(M または DB アドレス)。 情報システムの説明 Modbus_Master を選択します。
16#818C	DATA_PTR パラメータの相互接続エラー	命令の相互接続をチェックしてください。
16#818D	DATA_PTR でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が 読み出しまたは書き込みされるデータバイトの数 に対して小さすぎます。	DATA_PTR ポインタをチェックします。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.RDREC.STATUS または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。. エラーの原因の詳細については、静的パラメータ Send_P2P.WRREC.STATUS、Receive_P2P.WRREC.STATUS 、または Receive_Reset と、SFB WRREC の説明を参照してください。
通信エラー - Modbus_Master および Modbus_Slave		
16#80D1	XON または CTS = ON の待機時間が経過しました。	通信パートナーで障害が発生したか、速度が遅すぎるか、またはオフラインになっています。 通信パートナーをチェックするか、または必要に応じてパラメータを変更します。
16#80D2	「ハードウェア RTS は常に ON」: DSR = ON から OFF への切り替えのために送信ジョブがキャンセルされました	通信パートナーをチェックします。 伝送時間全体にわたって、DSR が確実に ON になるようにします。
16#80E0	フレームが中断されました: 送信バッファオーバーフロー/送信フレームが長すぎます	ユーザープログラムで命令を呼び出す回数を増やすか、またはデータフロー制御による通信を構成定義します。

16#80E1	フレームが中断されました: パリティエラー	通信パートナーの接続線をチェックするか、または両方のデバイスが同じデータ伝送率、パリティ、ストップビット数に構成定義されているかどうかを検証します。
16#80E2	フレームが中断されました: キャラクタフレームエラー	スタートビット、データビット、パリティビット、データ伝送率、ストップビットの設定をチェックします。
16#80E3	フレームが中断されました: キャラクタオーバーフロエラー	通信パートナーのフレーム内のデータ数をチェックします。
16#80E4	フレームが中断されました: 最大フレーム長に達しました	通信パートナー側で短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1～1024/2048/4096 (バイト)
通信エラー - Modbus_Master		
16#80C8	スレーブが設定された時間内に応答しません。	スレーブのデータ伝送率、パリティ、および配線をチェックします。
16#80C9	スレーブが Blocked_Proc_Timeout によって設定された時間内に応答しません。	Blocked_Proc_Timeout の設定をチェックします。 モジュールが、Modbus_Comm_Load 命令を使用して構成定義されたかどうかをチェックします。 モジュールは、抜き差し後または電圧復旧後に Modbus_Comm_Load を使用して再構成しなければならない場合があります。
16#8200	インターフェースが出力要求でビジーになっています。	後からコマンドを繰り返します。 新しいコマンドを開始する前に、まだ実行中のコマンドが存在しないことを確認します。
プロトコルエラー - Modbus_Slave (Modbus をサポートしている通信モジュールの場合のみ)		
16#8380	CRC エラー	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#8381	ファンクションコードがサポートされていない、またはブロードキャストでサポートされていません。	通信パートナーをチェックして、有効なファンクションコードが送信されていることを確認します。
16#8382	要求フレーム内の情報の長さが無効	DATA_LEN パラメータで、適切なデータ長を選択します。
16#8383	要求フレーム内のデータアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。
16#8384	要求フレーム内のデータ値無効エラー	Modbus マスタの要求フレーム内のデータ値をチェックします。
16#8385	診断値が Modbus スレーブでサポートされません(ファンクションコード 08)	Modbus スレーブは、診断値 16#0000 および 16#000A のみをサポートします。
プロトコルエラー - Modbus_Master (Modbus をサポートしている通信モジュールの場合のみ)		
16#8380	CRC エラー	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#8381	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: ファンクションコードがサポートされていません。	通信パートナーをチェックして、有効なファンクションコードが送信されていることを確認します。

16#8382	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 長さが無効	適切なデータ長を選択します。
16#8383	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 要求フレーム内のデータアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。
16#8384	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: データ値エラー	Modbus スレーブへの要求フレームをチェックします。
16#8385	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 診断値が Modbus スレーブでサポートされていません。	Modbus スレーブは、診断値 16#0000 および 16#000A のみをサポートします。
16#8386	戻されたファンクションコードが要求されたファンクションコードと一致しません。	スレーブの応答フレームとアドレス指定をチェックします。
16#8387	応答を要求されなかったスレーブ	スレーブの応答フレームをチェックします。 スレーブのアドレス設定をチェックします。
16#8388	書き込み要求に対するスレーブの応答のエラー	スレーブの応答フレームをチェックします。
16#8828 [1)]	DATA_PTR が、n * 8 と等しくないビットアドレスを指しています	DATA_PTR ポインタをチェックします。
16#8852 [1)]	DATA_PTR が、DB またはビットメモリ領域へのポインタではありません	DATA_PTR ポインタをチェックします。
16#8853 [1)]	DATA_PTR が、タイプ BOOL または WORD のポインタではありません	DATA_PTR ポインタをチェックします。
16#8855 [1)]	DATA_PTR が書き込み禁止 DB を指しています	DATA_PTR ポインタをチェックします。
16#8856 [1)]	SFC51 呼び出し中のエラー	Modbus_Master 命令を再度呼び出してください。
エラー - Modbus_Slave (Modbus をサポートしている通信モジュールの場合のみ)		
16#8428 [1)]	MB_HOLD_REG が、n * 8 と等しくないビットアドレスを指しています	MB_HOLD_REG ポインタをチェックします
16#8452 [1)]	MB_HOLD_REG が、DB またはビットメモリ領域へのポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8453 [1)]	MB_HOLD_REG が、タイプ BOOL または WORD のポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8454 [1)]	MB_HOLD_REG でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が読み出しまたは書き込みされるデータバイトの数に対して小さすぎます。	MB_HOLD_REG ポインタをチェックします
16#8455 [1)]	MB_HOLD_REG が書き込み禁止 DB を指しています	MB_HOLD_REG ポインタをチェックします
16#8456 [1)]	SFC51 呼び出し中のエラー	Modbus_Slave 命令を再度呼び出してください。
[1)] S7-300/400 CPU 用の命令の場合のみ		

エラーメッセージの概要 - USS

エラーコード	説明	対策
16#0000	エラーはありません。	-
16#8180	ドライブの応答の長さエラー:	ドライブの応答フレームをチェックします。
16#8181	データタイプエラー	適切なデータタイプを選択します。 以下が有効です。 • REAL • ワード • ダブルワード
16#8182	データタイプエラー:「ダブルワード」または「実数」を「ワード」要求に対して返すことはできません。	ドライブの応答フレームをチェックします。
16#8183	データタイプエラー:「ワード」を「ダブルワード」または「実数」要求に対して返すことはできません。	ドライブの応答フレームをチェックします。
16#8184	ドライブの応答のチェックサムエラー:	ドライブと通信接続をチェックします。
16#8185	アドレス指定エラー	有効なドライブアドレス範囲: 1 ~ 16
16#8186	セットポイントエラー	有効なセットポイント範囲: -200 % ~ +200 %
16#8187	不正なドライブ番号が返されました	ドライブの応答フレームをチェックします。
16#8188	PZD 長さが無効	許可されている PZD 長さ: 2、4、6、8 ワード
16#8189	モジュールがこのデータ伝送率をサポートしていません。	モジュールの有効なデータ伝送率を選択します。
16#818A	このドライブに対する別の要求が現在有効です。	後から、パラメータ読み出しまたは書き込みコマンドを繰り返します。
16#818B	ドライブが応答しません。	ドライブをチェックします。
16#818C	ドライブがパラメータ要求に対してエラーメッセージで応答しました	ドライブの応答フレームをチェックします。 パラメータ要求をチェックします。 命令 USS_Read_Param、USS_Write_Param または USS_Port_Scan にエラーの報告があるかチェックします。 エラーの報告がある場合、USS_Drive_Control 命令の静的タグ USS_DB. w_USSExtendedError の値をチェックします。
16#818D	ドライブがパラメータ要求に対してアクセスエラーメッセージで応答しました	ドライブの応答フレームをチェックします。 パラメータ要求をチェックします。
16#818E	ドライブが初期化されていません。	ユーザープログラムをチェックして、USS_Drive_Control 命令がこのドライブに

		対して呼び出されていることを確認します。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Port_Config.RDREC.STATUS、Send_Config.RDREC.STATUS 、 Receive_Config.RDREC.STATUS 、Send_P2P.RDREC.STATUS 、または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。. エラーの原因の詳細については、静的パラメータ Port_Config.WRREC.STATUS、Send_Config.WRREC.STATUS 、 Receive_Config.WRREC.STATUS 、Send_P2P.RDREC.STATUS 、または Receive_P2P.RDREC.STATUS と、SFB WRREC の説明を参照してください。
[1] S7-300/400 CPU 用の命令の場合のみ		

MODBUS (RTU)

この章には下記に関する情報が記載されています：

- Modbus RTU 通信の概要 (S7-1200, S7-1500)
- Modbus_Comm_Load: Modbus 用の通信モジュールを設定 (S7-1200, S7-1500)
- Modbus_Master: Modbus マスタとして通信 (S7-1200, S7-1500)
- Modbus_Slave: Modbus スレーブとして通信 (S7-1200, S7-1500)
- フレーム構造 (S7-1200, S7-1500)
- エラーメッセージ (S7-1200, S7-1500)

Modbus RTU 通信の概要

Modbus RTU 通信

Modbus RTU (Remote Terminal Unit)はネットワーク内での通信用の標準プロトコルで、RS232 または RS422/485 接続を使用してネットワーク内の Modbus デバイス間のシリアルデータ伝送を行います。

Modbus RTU は、マスタ/スレーブネットワークを使用します。すべての通信が 1 台のマスタデバイスでトリガされ、スレーブはマスタの要求への応答のみを行います。 マスタはスレーブアドレスに要求を送信し、このスレーブアドレスを持つスレーブのみがコマンドに応答します。

例外: Modbus スレーブアドレス 0 は、すべてのスレーブにブロードキャストフレームを送信します(スレーブ応答なし)。

Modbus ファンクションコード

- Modbus RTU マスタとして操作される CPU は、通信接続によって接続されている Modbus RTU スレーブとの間でデータおよび I/O 状態の読み出しおよび書き込みを行うことができます。
- Modbus RTU スレーブとして操作される CPU により、通信接続で接続されている Modbus RTU マスタが自分の CPU 内のデータおよび I/O 状態の読み出しおよび書き込みを行うことができます。

データ読み出し用ファンクション リモート I/O とプログラムデータの読み出し

MODBUS ファンクションコード	スレーブ(サーバー)からのデータ読み出し用ファンクション - 標準アドレス指定
01	出力ビットの読み出し: 要求当たり 1～2000/1992[1)]ビット
02	入力ビットの読み出し: 要求当たり 1～2000/1992[1)]ビット
03	保持レジスタの読み出し 要求当たり 1～125/124[1)]ワード
04	入力ワードの読み出し: 要求当たり 1～125/124[1)]ワード
1) 拡張アドレス指定の場合	

データ書き込み用ファンクション リモート I/O とプログラムデータの変更

MODBUS ファンクションコード	スレーブ(サーバー)へのデータ書き込み用ファンクション - 標準アドレス指定
05	1 つの出力ビットの書き込み: 要求当たり 1 ビット
06	1 つの保持レジスタの書き込み: 要求当たり 1 ワード
15	1 つ以上の出力ビットの書き込み: 要求当たり 1～1960 ビット
16	1 つ以上の保持レジスタの書き込み: 要求当たり 1～122 ワード

- Modbus ファンクションコード 08 および 11 は、スレーブデバイスとの通信用診断オプションを提供します。
- Modbus スレーブアドレス 0 は、すべてのスレーブにブロードキャストフレームを送信します(スレーブ応答なし; ファンクションコード 5、6、15、16 の場合)。

Modbus ネットワーク内のステーションアドレス

ステーション		アドレス
RTU ステーション	標準ステーションアドレス	1～247 、およびブロードキャストの場合は 0
	拡張ステーションアドレス	1～65535、およびブロードキャストの場合は 0

Modbus メモリアドレス

実際に使用可能な Modbus メモリアドレス(入力/出力アドレス)の数は、CPU バージョンと使用可能なワークメモリに応じて異なります。

プログラムの Modbus RTU 命令

- Modbus_Comm_Load: Modbus_Comm_Load を実行して、データ伝送率、パリティ、データフロー制御などの PtP パラメータをセットアプする必要があります。 Modbus RTU プロトコル用の通信モジュールを設定すると、このモジュールは Modbus_Master 命令または Modbus_Slave 命令以外では使用できなくなります。
- Modbus_Master: Modbus マスタ命令で、CPU を Modbus RTU マスタデバイスとして、1 つ以上の Modbus スレーブデバイスとの通信に使用することができます。
- Modbus_Slave: Modbus スレーブ命令で、CPU を Modbus RTU スレーブデバイスとして、1 つの Modbus マスタデバイスとの通信に使用することができます。

Modbus_Comm_Load: Modbus 用の通信モジュールを設定

説明

Modbus_Comm_Load 命令は、Modbus RTU プロトコルを使用した通信用の通信モジュールを構成定義します。 使用しているプログラムに Modbus_Comm_Load 命令を追加すると、インスタンスデータブロックが自動的に割り当てられます。

Modbus_Comm_Load の構成の変更は CM に保存され、CPU には保存されません。 電圧復旧および抜き差しにより、CM はデバイス構成に保存されたデータで構成定義されます。 Modbus_Comm_Load 命令は、以下のシナリオで呼び出す必要があります。

パラメータ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200/1500	S7-300/400 / WinAC		
REQ	IN	Bool		FALSE	この入力の立ち上がりエッジ時に命令が開始されます。
PORT	IN	Port	Laddr	0	CM の構成定義が終了すると、CM ポート値はデバイス構成の[ハードウェア ID]プロパティに表示されます(S7-1200/1500) (入力アドレス; S7-300/400)。シンボルポート名は、PLC タグテーブルの[システム定数]タブで割り当てられます。 S7-300/400/WinAC システムでは、HWCN で割り当てられた入力アドレスを CM ポートに割り当てる必要があります。
BAUD	IN	UDInt	DWord	9600	データ伝送速度の選択 有効な値: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 bit/s.
PARITY	IN	UInt	Word	1	パリティの選択: • 0 – なし • 1 – 奇数 • 2 – 偶数
FLOW_CTRL	IN	UInt	Word	0	フロー制御の選択 • 0 – (既定)フロー制御なし • 1 – RTS が常にオンのハードウェアフロー制御(RS422/485 CM の場合、無効) • 2 – RTS が切り替えられるハードウェアフロー制御(RS422/485 CM の場合、無効)
RTS_ON_DLY	IN	UInt	Word	0	RTS オンディレイの選択 • 0 – フレームの最初の文字が送信されるまで、「RTS 有効」からの遅延なし。 • 1 ~ 65535 – フレームの最初の文字が送信されるまで、「RTS 有効」からの遅延あり(ミリ

					秒) (RS422/485 CM の場合、無効)。 RTS ディレーは、FLOW_CTRL の選択とは関わりなく、使用する必要があります。
RTS_OFF_DLY	IN	UInt	Word	0	RTS オフ遅延の選択: • 0 – 最後の文字が転送された後、「RTS 無効」までの遅延なし • 1 ~ 65535 – 最後の文字が転送された後、「RTS 無効」までの遅延あり(ミリ秒) (RS422/485 ポートの場合、無効)。 RTS ディレーは、FLOW_CTRL の選択とは関わりなく、使用する必要があります。
RESP_TO	IN	UInt	Word	1000	応答タイムアウト: 5 ms ~ 65535 ms - Modbus_Master がスレーブからの応答を待機する時間(ミリ秒)。 スレーブがこの時間内に応答しない場合、Modbus_Master は要求を繰り返すか、または指定された繰り返し回数(下記、RETRIES パラメータを参照)の要求が送信された場合には要求をエラーで終了します。
MB_DB	IN/OUT	MB_BASE		-	Modbus_Master または Modbus_Slave 命令のインスタンスデータブロックへの参照。 この MB_DB パラメータは、Modbus_Master または Modbus_Slave 命令の(静的であるため、命令に表示されない) MB_DB パラメータと接続する必要があります。
COM_RST	IN/OUT	---	Bool	FALSE	Modbus_Comm_Load 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DONE	OUT	Bool		FALSE	DONE ビットは、最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE です。
ERROR	OUT	Bool		FALSE	ERROR ビットは、最後の要求がエラーで完了した後、1 サイクルの間 TRUE です。 STATUS パラメータのエラーコードは、ERROR = TRUE のサイクルでのみ有効です。
STATUS	OUT	Word		16#7000	エラーコード(エラーメッセージを参照)

Modbus_Comm_Load は、Modbus RTU プロトコル用のポートを構成定義する場合に実行します。 Modbus RTU プロトコル用のポートを構成定義すると、このポートは命令 Modbus_Master または命令 Modbus_Slave 以外では使用できなくなります。

Modbus 通信に使用する通信ポートの構成ごとに、Modbus_Comm_Load を実行する必要があります。 使用する各ポートに、一意の Modbus_Comm_Load インスタンス DB を割り当てます。 データ伝送速度またはパリティなどの通信パラメータを変更する必要がある場合、またはネットワークが復旧した場合にのみ、再度 Modbus_Comm_Load を実行します。

たとえば、Modbus_Master または Modbus_Slave をプログラムに追加すると、インスタンスデータブロックが命令に割り当てられます。 Modbus_Comm_Load 命令の MB_DB パラメータを Modbus_Master または Modbus_Slave 命令の MB_DB パラメータに接続する必要があります。

Modbus_Comm_Load データブロックタグ

下の表に、プログラムで使用できる Modbus_Comm_Load のインスタンス DB 内のパブリックな静的タグを示します。

インスタンス DB の静的タグ

タグ	データタイプ		標準	説明
	S7-1200/1500	S7-300/400/ WinAC		
ICHAR_GAP	Word		0	キャラクタ間最大遅延時間。 このパラメータはミリ秒単位で指定され、受信文字間の予想時間を大きくします。 このパラメータに対応するビット時間数が、Modbus 既定値の 35 ビット時間(3.5 キャラクタ時間)に追加されます。
RETRIES	Word		2	「応答なし」のエラーコード 0x80C8 が返される前に、マスタが実行する再試行回数。
EN_SUPPLY_VOLT	Bool		0	電源電圧 L+が不明の診断の有効化
MODE	USInt	バイト	0	動作モード 有効な動作モード以下のとおりです。 • 0 = 全二重(RS232) • 1 = 全二重(RS422) 4 線式モード(ポイントツーポイント) • 2 = 全二重(RS422) 4 線式モード(マルチポイントマスタ; CM PtP (ET 200SP)) • 3 = 全二重(RS422) 4 線式モード(マルチポイントスレーブ; CM PtP (ET 200SP)) • 4 = 半二重(RS485) 2 線式モード [1)]
LINE_PRE	USInt	バイト	0	回線初期状態の受信 有効な初期状態は以下のとおりです。 • 0 = 初期状態「なし」 [1)] • 1 = シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) この初期状態では、ブレーク検出が可能です。 以下とともにのみ選択できます。 「全二重(RS422) 4 線式モード(ポイントツーポイント接続)」および「全二重(RS422) 4 線式モード(マルチポイントスレーブ)」 • 2 = シグナル R(A) 0 V/シグナル R(B) 5 V この初期設定はアイドル状態に相当します(送信動作がアクティブでない)。 この初期状態では、ブレーク検出は不可能です。
BRK_DET	USInt	バイト	0	ブレーク検出 以下が有効です。 • 0 = ブレーク検出が無効 • 1 = ブレーク検出が有効
EN_DIAG_ALARM	Bool		0	診断割り込みを有効にします。 • 0 - 有効になっていません • 1 - 有効になっています

STOP_BITS	USINT	バイト	1	ストップビット数: • 1 = 1 ストップビット • 2 = 2 ストップビット • 0、3 ~ 255 = 予約済み
1) RS485 の CM 1241 で PROFIBUS ケーブルを使用するために必要な設定。				

Modbus_Master: Modbus マスタとして通信

説明

Modbus_Master 命令により、Modbus_Comm_Load 命令で構成定義されたポート経由で Modbus マスタとして通信します。 使用しているプログラムに Modbus_Master 命令を追加すると、インスタンスデータブロックが自動的に割り当てられます。 Modbus_Comm_Load 命令の MB_DB パラメータを Modbus_Master 命令の(静的) MB_DB パラメータに接続する必要があります。

パラメータ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200/1500	S7-300/400 / WinAC		
REQ	IN	Bool		FALSE	FALSE = 要求なし TRUE = Modbus スレーブへのデータの送信要求
MB_ADDR	IN	UInt	Word	-	Modbus RTU ステーションアドレス: 標準アドレス指定の範囲(1 ~ 247、Broadcast の場合 0) 拡張アドレス指定の範囲(1 ~ 65535、Broadcast の場合 0) 値 0 は、すべての Modbus スレーブへのフレームのブロードキャスト用に予約されています。Modbus ファンクションコード 05、06、15、および 16 のみが、ブロードキャスト用にサポートされています。
MODE	IN	USInt	Byte	0	モードの選択: 要求のタイプ(読み出し、書き込み、または診断)を指定します。 追加情報については、下記の Modbus ファンクションの表を参照してください。
DATA_ADDR	IN	UDInt	DWord	0	スレーブの開始アドレス: Modbus スレーブでアクセスするデータの開始アドレスを指定します。 有効なアドレスは、下記の Modbus ファンクションの表に記載されています。
DATA_LEN	IN	UInt	Word	0	データ長: この要求でアクセスするビットまたはワードの数を指定します。 有効な長さは、下記の Modbus ファンクションの表に記載されています。
COM_RST	IN/OUT	---	Bool	FALSE	Modbus_Master 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
DATA_PTR	IN/OUT	Variant	Any	-	データポインタ: 書き込みまたは読み出しするデータのフラグまたは DB をポイントします。
DONE	OUT	Bool		FALSE	DONE ビットは、最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE です。

BUSY	OUT	Bool	-	• FALSE – Modbus_Master の有効なコマンドなし • TRUE – Modbus_Master のコマンドが進行中
ERROR	OUT	Bool	FALSE	ERROR ビットは、最後の要求がエラーで完了した後、1 サイクルの間 TRUE です。 STATUS パラメータのエラーコードは、ERROR = TRUE のサイクルでのみ有効です。
STATUS	OUT	Word	0	エラーコード(エラーメッセージを参照)

Modbus マスタのデータブロック内のタグ

下の表に、プログラムで使用できる Modbus_Master のインスタンス DB 内のパブリックな静的タグを示します。

インスタンス DB の静的タグ

タグ	データタイプ	標準	説明
Blocked_Proc_Time-out	Real	0.5	このインスタンスが ACTIVE として削除されるまで、ブロックされた Modbus マスタが待機する時間(秒単位)。 これは、たとえば、マスタ要求が出力され、マスタファンクションが要求を完了する前にプログラムがその呼び出しを停止したときに発生します。 時間値は、エラーの発生を避けるために 0 以上、55 未満であることが必要です。 「Modbus-Master による通信のルール」および「異なったパラメータ設定の Modbus_Master 命令の呼び出し」も参照してください。
Extended_Address-ing	Bool	FALSE	スレーブステーションアドレスをシングルまたはダブルバイトとして構成定義します。 • FALSE = 1 バイトアドレス; 0～247 • TRUE = 2 バイトアドレス(拡張アドレス指定に対応); 0～65535
Compatibility_Mode 1)	Bool	FALSE	CP 341 および CP 441-2 との互換モード、Mod-bus RTU 用ドライバとの互換モード、および Modbus 用 ET 200SP 1SI との互換モード。 既定値は 0 です。 • FALSE = Modbus 仕様により、互換性なし • TRUE = 互換性あり ○ FC1 および FC2 の場合: 受信フレームから読み出されたデータはアドレス指定された CPU メモリにワード単位で書き込まれ、バイト単位で交換されます。 伝送するビット数が 16 の倍数でない場合、関連のないビットは最後のワードで null に設定されます。 ○ FC15 の場合: 送信されるワードはアドレス指定されたメモリからワード単位で読み出され、送信フレームにバイト単位で書き込まれます。

			送信するビット数が 8 の倍数でない場合、最後のバイト内の関連のないビットはそのままアドレス指定されたメモリから読み出され、送信フレームに入力されます。
MB_DB	MB_BASE	-	Modbus_Comm_Load 命令の MB_DB パラメータを Modbus_Master 命令のこの MB_DB パラメータに接続する必要があります。
1) PtP 通信モジュールは、Modbus 指定での定義に従って応答します。 Modbus 用の CP 341, 、CP 441-2、および ET 200SP 1SI と同様に応答を保持するために、「Compatibility_Mode」パラメータを使用します。			

使用しているプログラムは、Blocked_Proc_Timeout および Extended_Addressing タグに値を書き込み、Modbus マスタの操作を制御できます。

Modbus-Master による通信のルール

- Modbus_Master 命令がポートと通信できるようにするには、Modbus_Comm_Load を実行してポートを構成定義する必要があります。
- Modbus マスタとして使用されるポートは、Modbus_Slave で使用できません。 このポートでは、1 つまたは複数の Modbus_Master のインスタンス 1)を使用できます。 ただし、Modbus_Master のすべてのバージョンがポートに対して同じインスタンス DB を使用する必要があります。
- Modbus 命令は、通信プロセスを制御するために、通信アラームイベントを使用しません。 ユーザープログラムは、Modbus_Master 命令を使用して、完了したコマンド(DONE、ERROR)を照会する必要があります。
- 特定のポートに対する Modbus_Master のすべての実行をプログラムサイクル OB から呼び出すことをお勧めします。Modbus マスタ命令は、1 つのプログラムサイクルまたは 1 つの周期/時間制御処理レベル以外では実行できません。 異なった処理レベルでは処理できません。 Modbus マスタ命令が、優先度の高い処理レベルで別のマスタ命令から優先度割り込みを受けると、正しく動作できなくなります。 Modbus マスタ命令は、スタートアップレベル、診断レベル、またはエラーレベルでは処理できません。

1) この場合の「Modbus マスタのインスタンス」は、Modbus_Comm_Load 命令との同一の相互接続と、MB_ADDR、MODE、DATA_ADDR、および DATA_LEN パラメータに対する同一の設定による Modbus_Master の呼び出しを示します。

例

Modbus_Master は、MODE = 0 および DATA_ADDR = 10 で呼び出されます

このジョブは、DONE = 1 または ERROR = 1 で完了するか、または Blocked_Proc_Timeout パラメータで構成定義されたタイムモニタリングの時間が経過するまで有効です。 ウォッチドッグタイムの経過後で、直前のコマンドが完了する前に新しいコマンドが開始された場合、直前のコマンドはエラーメッセージなしで中止されます。

このコマンドの実行中に、この命令の 2 回目の呼び出しが、インスタンスデータは同一であるが、MODE および DATA_ADDR パラメータ設定は異なるという条件で行われると、この 2 回目の呼び出しは ERROR = 1 および STATUS = 8200 で終了します。

異なったパラメータ設定の Modbus_Master 命令の呼び出し

MB_ADDR、MODE, DATA_ADDR、または DATA_LEN の設定が異なる Modbus_Master 命令の複数の呼び出しをユーザープログラムに配置した場合は、いつでもこれらの呼び出しの 1 つのみが必ず有効になるようにしてください。 これを行わない場合、エラーメッセージ 16#8200 が出力されます(インターフェースが進行中の要求でビジーです)。

呼び出しを完全に処理できない場合、ウォッチドッグが Blocked_Proc_Timeout パラメータによって有効になり、進行中のコマンドを終了します。

REQ パラメータ

FALSE = 要求なし; TRUE = Modbus スレーブへのデータの送信要求

要求された伝送を有効にします。 これにより、バッファの内容がポイントツーポイント通信インターフェースに伝送されます。

DATA_ADDR および MODE パラメータを使用して、Modbus ファンクションコードを選択します。

DATA_ADDR (スレーブの Modbus 開始アドレス): Modbus スレーブでアクセスするデータの開始アドレスを指定します。

Modbus_Master 命令は、ファンクションコード入力の代わりに MODE 入力を使用します。 MODE と DATA_ADDR の組み合わせによって、実際の Modbus フレームで使用されるファンクションコードが指定されます。 下の表に、MODE パラメータ、Modbus ファンクションコード、および DATA_ADDR の Modbus アドレス範囲の関係を示します。

Modbus ファンクション

MODE	DATA_ADDR (Modbus アドレス)			DATA_LEN (データ長)			Modbus ファンクションコード	操作およびデータ		
0				要求当たりのビット			01	出力ビットの読み出し:		
	1	~	9999	1	~	2000/1992 [1]		0	~	9998
0				要求当たりのビット			02	入力ビットの読み出し:		
	10001	~	19999	1	~	2000/1992 [1]		0	~	9998
0				要求当たりのワード			03	保持レジスタの読み出し		
	40001	~	49999	1	~	125/124 [1]		0	~	9998
	400001	~	465535	1	~	125/124 [1]		0	~	65534
0				要求当たりのワード			04	入力ワードの読み出し:		
	30001	~	39999	1	~	125/124 [1]		0	~	9998
1				要求当たりのビット			05	1 つの出力ビットの書き込み:		
	1	~	9999	1				0	~	9998
1				要求当たり 1 ワード			06	1 つの保持レジスタの書き込み:		
	40001	~	49999	1				0	~	9998
	400001	~	465535	1				0	~	65524
1				要求当たりのビット			15	複数の出力ビットの読み出し:		
	1	~	9999	2	~	1968/1960 [1]		0	~	9998
1				要求当たりのワード			16	複数の保持レジスタの書き込み		

	40001	~	49999	2	~	123/122		0	~	9998
	400001	~	465534	2	~	123/122 [1]		0	~	65534
2 [2]				要求当たりのビット			15	1 つ以上の出力ビットの書き込み:		
	1	~	9999	2	~	1968/1960 [1]		0	~	9998
2 [2]				要求当たりのワード			16	1 つ以上の保持レジスタの書き込み:		
	40001	~	49999	1	~	123		0	~	9998
	400001	~	465535	1	~	122 [1]		0	~	65534
11	このファンクションでは、Modbus_Master の DATA_ADDR オペランドと DATA_LEN オペランドの両方が無視されます。						11	スレーブ通信のステータスワードおよびイベントカウンタを読み出します。 ステータスワードはビジーを示します(0 – ビジーでない、0xFFFF - ビジー)。 イベントカウンタは、フレームの処理が正常に行われるたびにインクリメントされます。		
80				要求当たり 1 ワード			08	データ診断コード 0x0000 でスレーブのステータスをチェックします(ループバックテスト – スレーブが要求のエコーを返します)。		
	-			1				-		
81				要求当たり 1 ワード			08	データ診断コード 0x000A を使用してスレーブのイベントカウンタをリセットします。		
	-			1				-		
104 [3]				要求当たりのワード			04	入力ワードの読み出し		
	0	~	65535	1	~	125/124 [1]		0	~	65535
3 ~ 10、12 ~ 79、82 ~ 103、105 ~ 255	-			-				予備		

[1] 拡張アドレス指定では、Extended_Adressing パラメータを参照してください。最大データ長が、ファンクションのデータタイプに応じて 1 バイトまたは 1 ワード短くなります。

[2] MODE 2 では、Modbus ファンクション 15 および 16 を使用して、1 つ以上の出力ビットと 1 つ以上の保持レジスタを書き込むことができます。

MODE 1 は Modbus ファンクション 5 および 6 を使用して、1 つの出力ビットと 1 つの保持レジスタを書き込み、Modbus ファンクション 15 および 16 を使用して複数の出力ビットと複数の保持レジスタを書き込みます。

[3] モード 104 は S7-1200 でのみ使用可能です。

DATA_PTR パラメータ

DATA_PTR パラメータは、読み出し元または書き込み先の DB またはビットメモリアドレスを指します。 データブロックを使用する場合は、Modbus スレーブで読み出し処理と書き込み処理を行うためのデータメモリを提供するグローバルデータブロックを作成する必要があります。

注記

S7-1200/1500 - DATA_PTR を使用してアドレス指定されたデータブロックは、ダイレクトアドレス指定をサポートしている必要があります。

データブロックは、ダイレクト(絶対)およびシンボリックアドレスを許可する必要があります。

注記

ファンクションコード 5 を使用

ファンクションコード 5 は、個々のビットを設定または削除するために使用されます。

ビットがセットされると、値「16#FF00」は DATA_PTR によってアドレス指定された DB またはビットメモリ領域の先頭のワードに指定される必要があります。

- S7-1200 の場合、値「16#0100」はビットを設定するためにも指定できます。
- ビットをリセットするには、値「16#0000」は DATA_PTR によってアドレス指定された DB またはビットメモリ領域の先頭のワードに指定される必要があります。

その他のすべての値は、ERROR = TRUE および STATUS = 16#8384 により拒否されます。

DATA_PTR パラメータのデータブロック構造体

- 以下のデータタイプは、Modbus アドレス範囲(DATA_PTR) 30001 ~ 39999、40001 ~ 49999、および 400001 ~ 465535 からのワード読み出しで有効であるだけでなく、Modbus アドレス範囲(DATA_PTR パラメータ) 40001 ~ 49999 および 400001 ~ 465535 へのワード書き込みでも有効です。
  - データタイプ WORD、UINT、または INT の標準配列
  - WORD、UINT、または INT タイプの名前付き構造体。構造体内の各工レメントには、一意の名前と 16 ビットのデータタイプがあります。
  - 各工レメントが一意の名前と 16 ビットまたは 32 ビットのデータタイプを持つ、名前を付けられた複雑な構造体。
- Modbus アドレス範囲(DATA_PTR パラメータ) 00001 ~09999 に対するビットの読み出しおよび書き込みの場合と、10001 ~ 19999 からのビットの読み出しの場合。
  - Bool データタイプの標準フィールド
  - 一意の名前が付けられた Bool タグの名前が付けられた Bool 構造体。
- 絶対に必要というわけでありませんが、各 Modbus_Master に独自の個別のメモリ領域を割り当てることをお勧めします。 その理由は、複数の Modbus_Master 命令が同じメモリ領域で読み出しと書き込みを行うと、データが破損する確率が極めて高くなるためです。
- DATA_PTR の複数のデータ領域が、同じグローバルデータブロックにある必要はありません。Modbus 読み出し処理用に複数の領域を持つ 1 つのデータブロック、Modbus 書き込み処理用に 1 つのデータブロック、または各スレーブステーションに 1 つのデータブロックを作成することができます。

Modbus_Slave: Modbus スレーブとして通信

説明

ユーザープログラムは Modbus_Slave 命令を使用して、CM (RS422/485 または RS232).経由の Modbus スレーブとしての通信を行うことができます。 この命令を追加すると、STEP 7 は自動的にインスタンス DB を作成します。 Modbus_Comm_Load 命令の MB_DB パラメータを Modbus_Slave 命令の(静的) MB_DB パラメータに接続する必要があります。

パラメータ

パラメータ	宣言	データタイプ		標準	説明
		S7-1200/1500	S7-300/400 / WinAC		
MB_ADDR	IN	UInt	Word	-	Modbus スレーブの標準アドレス: 標準アドレス指定範囲(1～247) 拡張アドレス指定範囲(0～65535) 注記: 0 はブロードキャストアドレスです。
COM_RST	IN/OUT	---	Bool	FALSE	Modbus_Slave 命令の初期化 命令は TRUE に初期化されます。 その後、命令は COM_RST を FALSE にリセットします。
MB_HOLD_REG	IN/OUT	Variant	Any	-	Modbus 保持レジスタ DB へのポインタ: Modbus 保持レジスタは、フラグまたはデータブロックのメモリ領域です。
NDR	OUT	Bool		FALSE	新しいデータが使用可能: • FALSE – 新規データなし • TRUE – Modbus マスタによって新規のデータが書き込まれたことを示します。 NDR ビットは、最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE です。
DR	OUT	Bool		FALSE	データの読み出し: • FALSE – 読み出しデータなし • TRUE – この命令が、Modbus マスタによって受信されたデータをターゲット領域に格納したことを示します。 DR ビットは、最後の要求がエラーなしで完了した後、1 サイクルの間 TRUE です。
ERROR	OUT	Bool		FALSE	ERROR ビットは、最後の要求がエラーで完了した後、1 サイクルの間 TRUE です。 実行がエラーで終了した場合、STATUS パラメータのエラーコードは、ERROR = TRUE のサイクルでのみ有効です。
STATUS	OUT	Word		0	エラーコード(エラーメッセージを参照)

Modbus 通信のファンクションコード(1、2、4、5 、および 15)は、CPU のプロセスイメージ入力およびプロセスイメージ出力でビットとワードの読み書きができます。 MB_HOLD_REG パラメータは、これらのファンクションコードに対して 1 バイト以上のデータタイプとして定義する必要があります。 下の表に、CPU のプロセスイメージへの Modbus アドレスの割り当て例を示します。

Modbus アドレスのプロセスイメージへの割り当て

Modbus ファンクション						S7-1200			
コード	ファンクション	データ領域	アドレス領域			データ領域	CPU アドレス		
01	ビットの読み出し	出力	0	~	8191	プロセスイメージ出力	O0.0	~	O1023.7
02	ビットの読み出し	入力	0	~	8191	プロセスイメージ入力	I0.0	~	I1023.7
04	ワードの読み出し	入力	0	~	511	プロセスイメージ入力	IW0	~	IW1022
05	ビットの書き込み	出力	0	~	8191	プロセスイメージ出力	O0.0	~	O1023.7
15	ビットの書き込み	出力	0	~	8191	プロセスイメージ出力	O0.0	~	O1023.7

Modbus アドレスのプロセスイメージへの割り当て

Modbus ファンクション						S7-1500/S7-300/S7-400			
ファンクションコード	ファンクション	データ領域	アドレス領域			データ領域	CPU アドレス		
01	ビットの読み出し	出力	0	~	9998	プロセスイメージ出力	O0.0	~	O1248.6
02	ビットの読み出し	入力	0	~	9998	プロセスイメージ入力	I0.0	~	I1248.6
04	ワードの読み出し	入力	0	~	9998	プロセスイメージ入力	IW0	~	IW19996
05	ビットの書き込み	出力	0	~	9998	プロセスイメージ出力	O0.0	~	O1248.6
15	ビットの書き込み	出力	0	~	9998	プロセスイメージ出力	O0.0	~	O1248.6

注記
使用可能なアドレス領域は、CPU のメモリの構成に従って小さくなることがあります。

Modbus 通信のファンクションコード(3、6、16)は、フラグまたはデータブロックのメモリ領域内のアドレス領域である Modbus 保持レジスタを使用します。 保持レジスタのタイプは、Modbus_Slave 命令の MB_HOLD_REG パラメータで指定されます。

注記

S7-1200/1500 - MB_HOLD_REG データブロックのタイプ

Modbus 保持レジスタを持つデータブロックは、ダイレクト(絶対)およびシンボリックアドレス指定を許可する必要があります。

診断機能

S7-1200 Modbus_Slave の Modbus 診断機能		
ファンクションコード	**サブファンクション**	**説明**
08	0000H	エコーテストの出力要求データ: Modbus_Slave 命令が、受信したデータワードのエコーを Modbus マスタに返します。
08	000AH	通信イベントカウンタのクリア: Modbus_Slave 命令が、Modbus ファンクション 11 で使用される通信イベントカウンタをクリアします。
11		通信イベントカウンタの呼び出し: Modbus_Slave 命令は、内部通信イベントカウンタを使って、Modbus スレーブに送信された Modbus の読み出しおよび書き込み要求の成功した数を検出します。 ファンクション 8、ファンクション 11、およびブロードキャスト要求では、カウンタは繰り上げられません。 また、結果として通信エラー(たとえば、パリティまたは CRC エラー)になる要求でも繰り上げられません。

Modbus_Slave 命令は、Modbus マスタからのブロードキャスト書き込み要求に有効なアドレスへのアクセスが含まれている場合、この要求をサポートします。Modbus_Slave は、ブロードキャストファクションでサポートされていないファンクションコードに対して、エラーコード 16#8188 を生成します。

Modbus スレーブの変数

下の表に、プログラムで使用できる Modbus_Slave のインスタンスデータブロック内のパブリックな静的タグを示します。

Modbus スレーブの変数

タグ	**データタイプ**	**標準**	**説明**
HR_Start_Offset	Word	0	Modbus 保持レジスタの開始アドレスが指定されます (既定値 = 0)。
Extended_Address-ing	Bool	FALSE	拡張アドレス指定、スレーブアドレスをシングルまたはダブルバイトとして構成定義します。 (FALSE = シングルバイトアドレス、TRUE = ダブルバイトアドレス)
Request_Count	Word	0	このスレーブが受信したすべての要求の数
Slave_Mes-sage_Count	Word	0	この特定のスレーブに対して受信された要求の数
Bad_CRC_Count	Word	0	CRC エラーのある受信された要求の数
Broadcast_Count	Word	0	受信されたブロードキャスト要求の数
Exception_Count	Word	0	マスタを除いて確認が必要な Modbus 固有のエラー

Success_Count	Word	0	プロトコルエラーなしでこの特定のスレーブに対して受信した要求の数
MB_DB	MB_BASE	-	Modbus_Comm_Load 命令の MB_DB パラメータを Modbus_Master 命令のこの MB_DB パラメータに接続する必要があります。

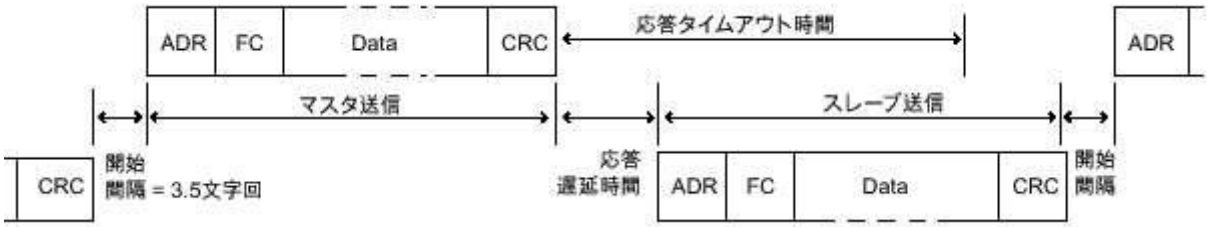
使用しているプログラムは、HR_Start_Offset および Extended_Addressing タグに値を書き込み、Modbus スレーブの操作を制御できます。 その他のタグを読み出して、Modbus ステータスをモニタすることができます。

Modbus スレーブ通信のルール

- Modbus_Slave 命令がポートと通信できるようにするには、Modbus_Comm_Load を実行してポートを構成定義する必要があります。
- ポートが Modbus マスタにスレーブとして応答しようとする場合、ポートは Modbus_Master 命令でプログラムされていない可能性があります。
- 特定のポートに対して、Modbus_Slave のインスタンスは 1 つしか使用できません。これに従わない場合、予期しない動作が行われることがあります。
- Modbus 命令は、通信プロセスを制御するために、通信アラームイベントを使用しません。 使用しているプログラムが、完了した送信および受信操作に対して Modbus_Slave 命令を照会して通信プロセスを制御する必要があります。
- Modbus_Slave 命令は、Modbus マスタの着信要求にタイムリーに応答できる頻度で定期的に実行する必要があります。 各サイクルごとに Modbus_Slave をプログラムサイクル OB から実行することをお勧めします。Modbus_Slave はサイクリック割り込み OB から実行できますが、割り込みプログラムでの遅延が大きくなると、他の割り込みプログラムの実行を一時的にブロックすることがあるため、お勧めできません。

Modbus 信号の時間制御

Modbus_Slave は、Modbus マスタの各要求を定期的に受信し、それに従って応答する必要があります。 Modbus_Slave が実行される頻度は、Modbus マスタからの応答に指定されたタイムアウト値によって異なります。 下の図にこれを示します。



(RESP_TO)応答のタイムアウト時間は、Modbus マスタが Modbus スレーブからの応答の開始まで待機する時間です。 この時間は Modobus プロトコルではなく、Modbus_Comm_Load 命令のパラメータによって定義されます。 フレームの受信と送信の両方で Modbus_Slave 命令の複数の呼び出し(最低 3 回)が必要なため、Modbus スレーブがタイムアウト時間で指定された回数の 2 倍の回数のデータの送受信を行うことができるように、Modbus マスタの応答のタイムアウト時間中に Modbus_Slave を最低 12 回実行する必要があります。

HR_Start_Offset

40001 または 400001 から開始される Modbus 保持レジスタのアドレス; これらのアドレスは、ターゲットシステムメモリの保持レジスタの開始アドレスに対応します。 ただし、HR_Start_Offset タグを使用して、40001 または 400001 と異なる Modbus 保持レジスタの開始アドレスを構成定義できます。

受信フレームのアドレス 0 は、ターゲットシステムメモリの保持レジスタの開始アドレスに対応します。 0 と異なる Modbus 保持レジスタの開始アドレスを構成定義するには、タグ HR_Start_Offset を使用します。

たとえば、MW100 から開始され、長さが 100 ワードの保持レジスタを構成定義することができます。 HR_Start_Offset = 20 の場合、受信フレームのアドレス 20 がターゲットメモリ(MW100)の保持レジスタの開始アドレスに対応します。 受信フレームの各アドレスが 20 未満であるか、または 119 を超えると、アドレス指定エラーが発生します。

DATA_PTR が長さが 100 ワードの MW100 へのポインタであるときの、Modbus 保持レジスタのアドレス指定の例

HR_Start_Offset	アドレス	最小値	最大
0	Modbus アドレス(ワード)	0	99
	S7-1500 アドレス	MW100	MW298
20	Modbus アドレス(ワード)	20	119
	S7-1500 アドレス	MW100	MW298

HR_Start_Offset は、Modbus 保持レジスタの開始アドレスを指定するワード値で、Modbus_Slave インスタンスデータブロックに保存されます。 このパブリックな静的タグは、プログラムに Modbus_Slave を追加した後で、パラメータのドロップダウンリストで選択します。

たとえば、Modbus_Slave を LAD ネットワークに追加した場合、前のネットワークに移動して移動コマンドを使用して値 HR_Start_Offset を割り当てることができます。 値は、Modbus_Slave を実行する前に割り当てる必要があります。

標準の DB 名を使用して Modbus スレーブタグを入力します。

1. カーソルを OUT1 パラメータフィールドに置いて、文字 m を入力します。
2. ドロップダウンリストから、Modbus_Slave 命令の必要なインスタンス DB を選択します。
3. カーソルを DB 名の右側(疑問符の後ろ)に置いて、ポイントを入力します。
4. ドロップダウンリストで[Modbus_Slave_DB.HR_Start_Offset]を選択します。

フレーム構造

Extended_Addressing

Extended_Addressing タグが Bool 値の場合を除き、説明に従って HR_Start_Offset リファレンスのために Extended_Addressing タグにアクセスします。

Modbus スレーブのアドレス指定で、Extended_Adressing = FALSE を使用してシングルバイト(Modbus 標準)を設定できます。また、Extended_Adressing = TRUE を使用して 2 バイトを設定することもできます。 拡張アドレス指定を使って、1 つのネットワークで 247 個以上のデバイスをアドレス指定できます。 Extended_Adressing = TRUE の場合、最大 65535 のアドレスを指定できます。 次の例は、Modbus フレームを示します。

1 バイトのスレーブアドレス(バイト 0)

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	
要求	[スレーブアドレス]	ファンクションコード	開始アドレス		DATA		
有効な応答	[スレーブアドレス]	ファンクションコード	長さ	データ...			
エラーメッセージ	[スレーブアドレス]	0xxx	例外コード				

2 バイトのスレーブアドレス(バイト 0 およびバイト 1)

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
要求	[スレーブアドレス]		ファンクションコード	開始アドレス		DATA	
有効な応答	[スレーブアドレス]		ファンクションコード	長さ	データ...		
エラーメッセージ	[スレーブアドレス]		0xxx	例外コード			

フレームの説明

マスタとスレーブ間/スレーブとマスタ間のデータトラフィックは、スレーブアドレスで開始され、その後にファンクションコードが続きます。 その後、データが転送されます。 データフィールドの構造は、使用するファンクションコードに依存します。 チェックサム(CRC)は、フレームの終了で伝送されます。

ファンクションコード 1 - このファンクションで、個々の出力ビットを読み出すことができます。

FC 1 - 出力ビットの読み出し

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 1	開始アドレス		出力の数	
有効な応答	[スレーブアドレス]	ファンクションコード 1	長さ: 1	出力データ 3)		
エラーメッセージ	[スレーブアドレス]	0x81	例外コード 2	---		

1 長さ: 出力数を 8 で除算した時に余りが出た場合、バイト数に 1 を足す必要があります。
2 E コード: 01、02、03、または 04
3 出力データは複数のバイトで構成できます。

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 1	開始アドレス		出力の数	
有効な応答	[スレーブアドレス]		ファンクションコード 1	長さ: 1	出力データ		
エラーメッセージ	[スレーブアドレス]		0x81	例外コード 2	---		

1 長さ: 出力数を 8 で除算した時に余りが出た場合、バイト数に 1 を足す必要があります。
2 E コード: 01、02、03、または 04
3 出力データは複数のバイトで構成できます。

ファンクションコード 2 - このファンクションで、個々の入力ビットを読み出すことができます。

FC 2 - 入力ビットの読み出し

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 2	開始アドレス		入力数	
有効な応答	[スレーブアドレス]	ファンクションコード 2	長さ: 1	入力データ		
エラーメッセージ	[スレーブアドレス]	0x82	例外コード 2	---		

1 長さ: 入力数を 8 で除算した時に余りが出た場合、バイト数に 1 を足す必要があります。
2 E コード: 01、02、03、または 04

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 2	開始アドレス		入力数	
有効な応答	[スレーブアドレス]		ファンクションコード 2	長さ: [1]	入力データ		
エラーメッセージ	[スレーブアドレス]		0x82	例外コード [2]	---		
[1] 長さ: 入力数を 8 で除算した時に余りが出た場合、バイト数に 1 を足す必要があります。 [2] E コード: 01、02、03、または 04							

ファンクションコード 3 - このファンクションで、個々のレジスタを読み出すことができます。

FC 3 - 保持レジスタの読み出し

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 3	開始アドレス		レジスタの数	
有効な応答	[スレーブアドレス]	ファンクションコード 3	長さ: [1]	レジスタデータ		
エラーメッセージ	[スレーブアドレス]	0x83	例外コード [2]	---		
[1] 長さ: バイト数 [2] E コード: 01、02、03、または 04						

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 3	開始アドレス		レジスタの数	
有効な応答	[スレーブアドレス]		ファンクションコード 3	長さ: [1]	レジスタデータ		
エラーメッセージ	[スレーブアドレス]		0x83	例外コード [2]	---		
[1] 長さ: バイト数 [2] E コード: 01、02、03、または 04							

ファンクションコード 4 - このファンクションで、個々のレジスタを読み出すことができます。

FC 4 - 入力ワードの読み出し

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 4	開始アドレス		入力ワードの数	
有効な応答	[スレーブアドレス]	ファンクションコード 4	長さ: [1]	入力データ		
エラーメッセージ	[スレーブアドレス]	0x84	例外コード [2]	---		
[1] 長さ: 2 * 入力ワードの数 [2] E コード: 01、02、03、または 04						

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 4	開始アドレス		入力ワードの数	
有効な応答	[スレーブアドレス]		ファンクションコード 4	長さ: [1]	入力データ		
エラーメッセージ	[スレーブアドレス]		0x84	例外コード [2]	---		
[1] 長さ: 2 * 入力ワードの数 [2] E コード: 01、02、03、または 04							

ファンクションコード 5 - このファンクションで、個々のビットを設定または削除できます。

FC 5 - 1 つの出力ビットの書き込み

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 5	開始アドレス		値	
有効な応答	[スレーブアドレス]	ファンクションコード 5	長さ	値		
エラーメッセージ	[スレーブアドレス]	0x85	例外コード [1]	---		
[1] E コード: 01、02、03、または 04						

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 5	開始アドレス		値	

有効な応答	[スレーブアドレス]	ファンクションコード 5	長さ	値
エラーメッセージ	[スレーブアドレス]	0x85	例外コード [1]	---
[1] E コード: 01、02、03、または 04				

ファンクションコード 6 - このファンクションで、個々のレジスタに書き込むことができます。

FC 6 - 保持レジスタの書き込み

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 6	アドレス		Register	
有効な応答	[スレーブアドレス]	ファンクションコード 6	アドレス		Register	
エラーメッセージ	[スレーブアドレス]	0x86	例外コード [1]	---		
[1] E コード: 01、02、03、または 04						

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 6	アドレス		Register	
有効な応答	[スレーブアドレス]		ファンクションコード 6	アドレス		Register	
エラーメッセージ	[スレーブアドレス]		0x86	例外コード [1]	---		
[1] E コード: 01、02、03、または 04							

ファンクションコード 8 - このファンクションを使用して、通信接続をチェックします。

FC 8 - スレーブステータス

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 8	診断コード		テスト値	
有効な応答	[スレーブアドレス]	ファンクションコード 8	診断コード		テスト値	

エラーメッセージ	[スレーブアドレス]	0x88	例外コード 1	---
1 E コード: 01、03、または 04				

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 8	診断コード		テスト値	
有効な応答	[スレーブアドレス]		ファンクションコード 8	診断コード		テスト値	
エラーメッセージ	[スレーブアドレス]		0x88	例外コード 1	---		
1 E コード: 01、03、または 04							

ファンクションコード 11 - このファンクションは、「ステータスワード」の 2 バイトまたは「イベントカウンタ」の 2 バイトを読み出すことができます。

FC 11 - スレーブ通信のイベントカウンタ

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
Query	[スレーブアドレス]	ファンクションコード 11	---			
有効な応答	[スレーブアドレス]	ファンクションコード 11	ステータス		イベントカウンタ	
エラーメッセージ	[スレーブアドレス]	0x8B	例外コード 1	---		
1 E コード: 01 または 04						

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
Query	[スレーブアドレス]		ファンクションコード 11	---			
有効な応答	[スレーブアドレス]		ファンクションコード 11	ステータス		イベントカウンタ	
エラーメッセージ	[スレーブアドレス]		0x8B	例外コード 1	---		
1 E コード: 01 または 04							

ファンクションコード 15 - このファンクションで、複数のビットを書き込むことができます。

FC 15 - 1 つ/複数の入力ビットの書き込み

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6	バイト 7	バイト n
Query	[スレーブアドレス]	ファンクションコード 15	開始アドレス		出力ワードの数		バイトカウンタ [1]	値	
有効な応答	[スレーブアドレス]	ファンクションコード 15	開始アドレス		出力ワードの数		---		
エラーメッセージ	[スレーブアドレス]	0x8F	例外コード [2]	---					
[1] バイトカウンタ: バイト数を 8 で除算した時に余りが出た場合、バイト数に 1 を足す必要があります。 [2] E コード: 01、02、03、または 04									

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6	バイト 7	バイト 8	バイト n
Query	[スレーブアドレス]		ファンクションコード 15	開始アドレス		出力ワードの数		バイトカウンタ [1]	値	
有効な応答	[スレーブアドレス]		ファンクションコード 15	開始アドレス		出力ワードの数		---		
エラーメッセージ	[スレーブアドレス]		0x8F	例外コード [2]		---				
[1] バイトカウンタ: バイト数を 8 で除算した時に余りが出た場合、バイト数に 1 を足す必要があります。 [2] E コード: 01、02、03、または 04										

ファンクションコード 16 - このファンクションで、個々のレジスタまたは複数のレジスタに書き込むことができます。

FC 16 - 1 つ/複数の保持レジスタの書き込み

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6	バイト 7	バイト n
Query	[スレーブアドレス]	ファンクションコード 16	開始アドレス		レジスタの数		バイトカウンタ [1]	値	
有効な応答	[スレーブアドレス]	ファンクションコード 16	開始アドレス		レジスタの数		---		
エラーメッセージ	[スレーブアドレス]	0x90	例外コード [2]	---					
[1] バイトカウンタ: レジスタの数 * 2 [2] E コード: 01、02、03、または 04									

	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6	バイト 7	バイト 8	バイト n
Query	[スレーブアドレス]		ファンクションコード 16	開始アドレス		レジスタの数		バイトカウンタ [1]	値	
有効な応答	[スレーブアドレス]		ファンクションコード 16	開始アドレス		レジスタの数		---		
エラーメッセージ	[スレーブアドレス]		0x90	例外コード [2]		---				

[1] バイトカウンタ: レジスタの数 * 2

[2] E コード: 01、02、03、または 04

エラーメッセージ

エラーメッセージの概要 - PtP

エラーメッセージは命令の STATUS 出力で提供され、そこで評価したりユーザープログラムで処理したりすることができます。

エラーコード	説明	対策
16#0000	エラーはありません。	-
受信ステータスとエラーコード		
16#0094	「固定/最大フレーム長の受信」に基づいて識別されたフレームの終了	-
16#0095	「メッセージタイムアウト」に基づいて識別されたフレームの終了	-
16#0096	「キャラクタ遅延時間」の経過に基づいて識別されたフレームの終了	-
16#0097	最大応答時間に達したためにフレームが中止されました。	-
16#0098	「メッセージからのメッセージ長の読み出し」条件の適合に基づいて識別されたフレームの終了	-
16#0099	「終了シーケンス」の受信に基づいて識別されたフレームの終了	-
送信ステータスとエラーコード		
16#7000	ブロックアイドル	-
16#7001	新しいフレームの最初の呼び出し データ伝送が開始されました	-
16#7002	中間呼び出し: データ伝送が実行中	-
16#8085	長さが無効	適切なフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8088	指定された長さが受信バッファの設定された範囲を超えています。 注記: データタイプ STRING が BUFFER パラメータで指定されている場合、現在の文字列が LENGTH パラメータで指定された長さよりも短いと、このエラーコードも表示されます。	受信バッファの範囲を変更するか、または受信バッファの設定された範囲に対応するフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8090	構成エラー: WString のバイト数が奇数です	偶数のバイト数を選択します。
受信ステータスとエラーコード		
16#8088	受信した文字数が BUFFER パラメータで指定された文字数を超えています。	適切なフレーム長を選択します。

		以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#8090	構成エラー: WString のバイト数が奇数です	偶数のバイト数を選択します。
特殊ファンクションのエラーメッセージコード		
16#818F	不正なパラメータ番号の設定(USS のみ)	適切なパラメータ番号(PARAM)を選択します。 次の番号が有効です。 0-2047
16#8190	CRC 計算の不正な設定	CRC 計算の適切な値を選択します。 以下が有効です。 無効または有効。 アドレス指定されたモジュールが CRC 計算をサポートしているかどうかチェックします。
16#8191	診断エラー割り込みの不正な設定	[診断エラー割り込み]の適切な値を選択します。 以下が有効です。 診断エラー割り込みが無効、または診断エラー割り込みが有効。 アドレス指定されたモジュールが診断割り込みの生成をサポートしているかどうかをチェックします。
「ポートの構成」のエラーメッセージコード		
16#81A0	モジュールがこのプロトコルをサポートしていません。	モジュールの有効なプロトコル(PROTOCOL)を選択します。
16#81A1	モジュールがこのデータ伝送率をサポートしていません。	モジュールの有効なデータ伝送速度(BAUD)を選択します。
16#81A2	モジュールがこのパリティ設定をサポートしていません。	「パリティ」(PARITY)の適切な値を選択します。 以下が有効です。 • なし(1) • 偶数(2) • 奇数(3) • マーク(4) • スペース(5) • 任意(6)
16#81A3	モジュールがこのデータビット数をサポートしていません。	「データビット数」(DATABITS)の適切な値を選択します。 以下が有効です。 • 7 (2) • 8 (1)
16#81A4	モジュールがこのストップビット数をサポートしていません。	「ストップビット数」(STOPBITS)の適切な値を選択します。 以下が有効です。 • 1 (1) • 2 (2)

16#81A5	モジュールがこのタイプのデータフロー制御をサポートしていません。	モジュールの有効なデータフロー制御(FLOWCTRL)を選択します。
16#81A7	XON または XOFF の無効な値	XON (XONCHAR)および XOFF (XOFF-CHAR)の適切な値を選択します。 値の有効な範囲: 0...255
16#81AA	動作モードが無効	有効な動作モード以下のとおりです。 • 全二重(RS232) (0) • 全二重(RS422) 4 線式モード(ポイントツーポイント) (1)/ (CM PtP (ET 200SP)) • 全二重(RS422) 4 線式モード(マルチポイントマスタ) (2)/ (CM PtP (ET 200SP)) • 全二重(RS 422) 4 線式モード(マルチポイントスレーブ) (3) • 半二重(RS485) 2 線式モード (4)
16#81AB	回線初期状態が無効	有効な初期状態は以下のとおりです。 • 初期設定[なし] (0) • シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) (1) 以下とともにのみ選択できます。「全二重(RS422) 4 線式モード(ポイントツーポイント接続)」および「全二重(RS422) 4 線式モード(マルチポイントスレーブ)」 • シグナル R(A) 0 V/シグナル R(B) 5 V (2): この初期設定はアイドル状態に相当します(送信動作がアクティブでない)。
16#81AC	[ブレーク検出]の値が無効	[ブレーク検出]の適切な値を選択します。以下が有効です。 • ブレーク検出が無効(0) • ブレーク検出が有効(1)
16#81AF	モジュールがこのプロトコルをサポートしていません。	モジュールの有効なプロトコルを選択します。
「送信構成」のエラーコード		
16#81B5	エンドデリミタが 2 つ以上、または終了シーケンスが 5 文字以上	[エンドデリミタ]と[終了シーケンス]に適切な値を選択します。 以下が有効です。 • 無効(0)、 • 1 つ (1)または 2 つ (2)のエンドデリミタ または • 無効(0)、 • 終了シーケンスに 1 文字 (1)から 5 文字 (5)
16#81B6	3964(R)プロトコルが選択されていたため、送信構成が拒否されました。	3964(R)プロトコルが設定されている場合は、送信構成が送信されないようにします。

「受信構成」のエラーコード		
16#81C0	開始条件が無効	適切な開始条件を選択します。 以下が有効です。 • フレーム開始前の送信中断 • アイドルラインの送信
16#81C1	終了条件が無効、または終了条件が選択されていません	適切な条件を選択します(Freeport によるデータ受信を参照)。
16#81C3	[最大メッセージ長]の値が無効	「最大メッセージ長」(MAXLEN)の適切な値を選択します。 値の有効な範囲(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#81C4	[メッセージ内の長さ指定のオフセット]の値が無効	[メッセージ内の長さ指定のオフセット]に有効な値を選択します。 値の有効な範囲(モジュールによって異なります): 1-1024/2048/4096 (Byte)
16#81C5	[長さフィールドのサイズ]の値が無効	「長さフィールドのサイズ」(LENGTHSIZE)の適切な値を選択します。 値の有効な範囲(バイト単位): • 1 (1) • 2 (2) • 4 (4)
16#81C6	[長さ指定でカウントされていない文字数]の値が無効	「長さ指定でカウントされていない文字数」(LENGTHM)の適切な値を選択します。 値の有効な範囲: 0～255 (バイト)
16#81C7	「メッセージ内のオフセット + 長さフィールドのサイズ + カウントされていない文字数」の合計が最大フレーム長を超えています	[メッセージ内のオフセット]、[長さフィールドのサイズ]、および[カウントされていない文字数]に適切な値を選択します。 値の有効な範囲: • メッセージ内のオフセット(モジュールによって異なります): 0-1024/2048/4096 (Byte) • 長さフィールドのサイズ: 1、2、または 4 (バイト) • カウントされていない文字数: 0～255 (バイト)
16#81C8	[応答タイムアウト]の値が無効	[応答タイムアウト]に適切な値を選択します。 値の有効な範囲: 1-65535 (ms)
16#81C9	[キャラクタ遅延時間]の値が無効	[キャラクタ遅延時間]に適切な値を選択します。 値の有効な範囲: 1～65535 (ビット時間)
16#81CB	フレーム終了シーケンスが有効になっていますが、チェック用の文字が有効になっていません	チェック用に 1 つ以上の文字を有効にします。

16#81CC	フレーム開始シーケンスが有効になっていますが、チェック用の文字が有効になっていません	チェック用に 1 つ以上の文字を有効にします。
16#81CD	[上書き禁止]の値が無効	[上書き禁止]に適切な値を選択します。 以下が有効です。 • 上書き禁止が無効(0)または • 上書き禁止が有効(1)
16#81CE	[スタートアップ時に受信バッファをクリアする]の値が無効	[スタートアップ時に受信バッファをクリアする]に適切な値を選択します。 以下が有効です。 • スタートアップ時に受信バッファをクリアが無効(0) • スタートアップ時に受信バッファをクリアが無効(1)
送信ステータスとエラーコード		
16#81D0	送信コマンドのランタイム中の送信要求の受信	送信コマンドのランタイム中に追加の送信要求を受信するのを防止します。
16#81D1	XON または CTS = ON の待機時間が経過しました。	通信パートナーで障害が発生したか、速度が遅すぎるか、またはオフラインになっています。 通信パートナーをチェックするか、または必要に応じてパラメータを変更します。
16#81D2	「ハードウェア RTS は常に ON」: DSR = ON から OFF への切り替えのために送信ジョブがキャンセルされました	通信パートナーをチェックします。 伝送時間全体にわたって、DSR が確実に ON になるようにします。
16#81D3	送信バッファオーバーフロー/送信フレームが長すぎます	短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1～1024/2048/4096 (バイト)
16#81D5	パラメータ変更、断線の検出、または CPU の STOP によって、伝送がキャンセルされました	パラメータ割り当て、断線、および CPU ステータスをチェックします。
16#81D6	終了識別子が受信されなかったため、伝送がキャンセルされました。	終了文字のパラメータ割り当てと通信パートナーのフレームをチェックします。
16#81D7	ユーザープログラムとモジュール間の通信エラー	通信をチェックします(たとえば、シーケンス番号の一致など)。
16#81D8	モジュールが構成定義されていないため、伝送の試行が拒否されました。	モジュールを構成定義します。
受信構成のエラーコード		
16#81E0	フレームが中断されました: 送信バッファオーバーフロー/送信フレームが長すぎます	ユーザープログラムで受信ファンクションの呼び出し率を上げるか、またはデータフロー制御による通信を構成定義する必要があります。
16#81E1	フレームが中断されました: パリティエラー	通信パートナーの接続線をチェックするか、または両方のデバイスが同じデータ伝送率、パリティ、ストップビット数に構成定義されているかどうかを検証します。

16#81E2	フレームが中断されました: キャラクタフレームエラー	スタートビット、データビット、パリティビット、データ伝送率、ストップビットの設定をチェックします。
16#81E3	フレームが中断されました: キャラクタオーバーフロエラー	ファームウェアエラー: カスタマーサポートに問い合わせてください。
16#81E4	フレームが中断されました: 「メッセージ内のオフセット + 長さフィールドのサイズ + カウントされていない文字数」の合計長が受信バッファを超えています	メッセージ内のオフセット、長さフィールドのサイズ、およびカウントされていない文字数に適切な値を選択します。
16#81E5	フレームが中断されました: break	パートナーへの受信ラインが中断しています。 再接続するか、またはパートナーのスイッチをオンにします。
16#81E6	[バッファリングされる受信フレーム]の最大数を超過しました	ユーザープログラムで命令を呼び出す回数を増やすか、データフロー制御による通信を構成定義するか、またはバッファリングされるフレームの数を増やします。
16#81E7	同期エラーモジュールと Receive_P2P	Receive_P2P のさまざまな異なるインスタンスが確実に同一モジュールにアクセスしないようにします。
16#81E8	フレームが中断されました: メッセージまたは条件が検出される前に、キャラクタ遅延時間が経過しました。	パートナーデバイスに障害が発生しているか、またはパートナーデバイスが遅すぎます。 必要に応じてインターフェーステスターを使用し、パートナーデバイスが伝送線で相互接続されていることをチェックします。
16#81E9	Modbus CRC エラー(Modbus をサポートしている通信モジュールのみ)	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#81EA	Modbus フレームが短すぎます(Modbus をサポートしている通信モジュールのみ)	Modbus フレームの最小長さが満たされていません 通信パートナーをチェックします。
16#81EB	フレームが中断されました: 最大フレーム長に達しました	通信パートナー側で短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1-1024/2048/4096 (Byte) フレーム終了検出のパラメータをチェックします。
エラーコード V24 に付随する信号		
16#81F0	モジュールが V24 に付随する信号をサポートしていません。	V24 に付随する信号をサポートしていないモジュールに対して付随する信号の設定を試みました。 必ず RS232 モジュールにするか、または RS232 モード(ET 200SP)を設定します。
16#81F1	V24 に付随する信号を操作できません	ハードウェアデータフロー制御が有効の場合、V24 に付随する信号を手動で操作することはできません。
受信構成のエラーコード		
16#8201 [1)]	Receive_Conditions は無効なデータタイプへのポインタです。	データタイプ

		DB、BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、DATE、TIME_OF_DAY、TIME、S5TIME、DATE_AND_TIME、STRING の 1 つを指すポインタを入力します。
16#8225	Receive_Conditions が、1 KB 以上の最適化されたメモリ領域をポイントしています。 または Receive_Conditions が最適化されたメモリ領域をポイントしており、受信長が Receive_Conditions でアドレス指定された領域を越えています。	以下の最大長を持つ領域へのポインタを入力します。 • 最適化されたメモリ領域: 1 kByte • 最適化されていない領域: 4 kByte 注記: ポインタが最適化されたメモリ領域を指す場合、1 kByte を超えるバイト数を送信してはいけません。
16#8229 1)	Receive_Conditions が、ビット数が n * 8 と等しくない BOOL へのポインタです。	BOOL へのポインタを使用している場合、ビット数は 8 の倍数とする必要があります。
エラーコード、一般		
16#8280	モジュールの読み出し時の否定確認	エラーの原因の詳細については、RDREC.STATUS 静的パラメータと、SFB RDREC の説明を参照してください。 • PORT パラメータの入力をチェックします。 • 初期呼び出しの前に、COM_RST パラメータを設定します。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、WRREC.STATUS 静的パラメータと、SFB WRREC の説明を参照してください。
16#8282	モジュールが使用不可	PORT パラメータの入力をチェックし、モジュールに確実にアクセスできるようにします。
受信構成のエラーコード		
16#82C1	[バッファリングされた受信フレーム]の値が無効	[バッファリングされた受信フレーム]に適切な値を選択します。 値の有効な範囲: 1-255
16#82C2	3964(R)プロトコルが選択されていたため、受信構成が拒否されました。	3964(R)プロトコルが設定されている場合は、受信構成が送信されないようにします。
16#8301 1)	Receive_Conditions は無効なデータタイプへのポインタです。	有効なデータタイプを選択します。 以下が有効です。 DB、BOOL、BYTE、CHAR、WORD、INT、DWORD、DINT、REAL、DATE、TIME_OF_DAY、TIME、S5TIME、DATE_AND_TIME、STRING
16#8322	パラメータ読み出し時の範囲長さエラー	Receive_Conditions パラメータの入力をチェックします。
16#8324	パラメータ読み出し時の範囲エラー	Receive_Conditions パラメータの入力をチェックします。
16#8328	パラメータ読み出し時の設定エラー	Receive_Conditions パラメータの入力をチェックします。

送信ステータスとエラーコード		
16#8328 1)	BUFFER が、ビット数が n * 8 と等しくない BOOL へのポインタです	BOOL へのポインタを使用している場合、ビット数は 8 の倍数とする必要があります。
受信構成のエラーコード		
16#8332	Receive_Conditions パラメータの無効なデータブロック	Receive_Conditions パラメータの入力をチェックします。
16#833A	Receive_Conditions パラメータのデータブロックの名称が、ロードされていないデータブロックを示しています。	Receive_Conditions パラメータの入力をチェックします。
16#8351	無効なデータタイプ	Receive_Conditions パラメータの入力をチェックします。
16#8352 1)	Receive_Conditions がデータブロックをポイントしていません。	Receive_Conditions へのポインタをチェックします。
16#8353 1)	Receive_Conditions タイプの構造をポイントしていません。 Receive_Conditions	Receive_Conditions へのポインタをチェックします。
エラーコード 3964(R)プロトコル		
16#8380	パラメータ割り当てエラー: [キャラクタ遅延時間]の値が無効	[キャラクタ遅延時間](CharacterDelay-Time)に適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8381	パラメータ割り当てエラー: [応答タイムアウト]の値が無効	[応答タイムアウト](AcknDelayTime)に適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8382	パラメータ割り当てエラー: [優先度]の値が無効	[優先度](Priority)に適切な値を選択します。 以下が有効です。 • 高い(1) • 低い(0)
16#8383	パラメータ割り当てエラー: [ブロックチェック]の値が無効	[ブロックチェック] (BCC)の適切な値を選択します。 以下が有効です。 • ブロックチェックあり(1) • ブロックチェックなし(0)
16#8384	パラメータ割り当てエラー: [接続試行]の値が無効.	[接続試行](BuildupAttempts)に適切な値を選択します。 値の有効な範囲: 1-255
16#8385	パラメータ割り当てエラー: [送信試行]の値が無効.	[送信試行](RepetitionAttempts)に適切な値を選択します。 値の有効な範囲: 1-255
16#8386	ランタイムエラー: 接続試行回数を超過しました	インターフェースケーブルおよび伝送パラメータをチェックします。 さらに、パートナーデバイス側で受信ファンクションが正しく構成定義されているかどうかもチェックします。

16#8387	ランタイムエラー: 送信試行回数を超過しました	インターフェースケーブル、伝送パラメータ、および通信パートナーの構成をチェックします。
16#8388	ランタイムエラー:「ブロックチェックキャラクタ」のエラー 内部的に計算したブロックチェックキャラクタの値が、接続の終了時にパートナーが受信したブロックチェックキャラクタに対応していません。	接続が致命的に中断されていないかどうかチェックします。その場合は、こまめにエラーコードをチェックすることをお勧めします。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#8389	ランタイムエラー: 空き受信バッファの待機中に無効な文字が受信されました	通信パートナーの送信要求(STX, 02H)は、受信バッファが空いている場合のみ、DLE で応答します。 この前に、追加文字を受信することはできません(再び、STX を除いて)。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838A	ランタイムエラー: 受信時の論理エラー DLE 受信後、追加のランダム文字(DLE または ETX を除く)を受信しました。	フレームヘッダー内およびデータ文字列内のパートナー DLE が常に重複しているか、または接続が DLE ETX でリリースされているかをチェックします。 パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838B	ランタイムエラー: キャラクタ遅延時間を超過しました	パートナーデバイスが遅すぎるか、またはパートナーデバイスに障害が発生しています。 必要に応じて、伝送線内に接続されたインターフェーステストデバイスを使用して検証します。
16#838C	ランタイムエラー: 空き受信バッファの待機時間が開始されました	ユーザープログラムで命令を呼び出す回数を増やすか、またはデータフロー制御による通信を構成定義します。
16#838D	ランタイムエラー: フレームの繰り返しが NAK 後、4 秒以内に開始されません。	通信パートナーをチェックします。 破壊されている可能性のある受信フレームをパートナーが 4 秒以内に繰り返す必要があります。
16#838E	ランタイムエラー: アイドルモードで、1 つ以上の文字(NAK、STX 以外)を受信しました。	パートナーデバイスが正しく動作していることをチェックします。伝送線内に接続されたインターフェーステストデバイスを使用するとチェックできる場合があります。
16#838F	ランタイムエラー: 初期化の競合 - 両方のパートナーが高い優先度を設定しています。	パートナーのどちらかで、「低い」優先度を設定します。
16#8391	パラメータ割り当てエラー: Freeport が設定されているため、3964 構成データが拒否されました	Freeport プロトコルが設定されている場合は、3964 パラメータ割り当てデータが送信されないようにします。
[1] S7-300/400 CPU 用の命令の場合のみ		

エラーメッセージの概要 - Modbus

エラーコード	説明	対策
16#0000	エラーはありません。	-
インターフェースの構成エラー - Modbus_Comm_Load		
16#8181	モジュールがこのデータ伝送率をサポートしていません。	BAUD パラメータで、モジュールの有効なデータ伝送速度を選択します。
16#8182	モジュールがこのパリティ設定をサポートしていません。	PARITY パラメータの「パリティ」の適切な値を選択します。 以下が有効です。 • なし(1) • 偶数(2) • 奇数(3) • マーク(4) • スペース(5) • 任意(6)
16#8183	モジュールがこのタイプのデータフロー制御をサポートしていません。	FLOW_CTRL パラメータで、モジュールの有効なデータフロー制御を選択します。
16#8184	[応答タイムアウト]の値が無効	RESP_TO パラメータで、[応答タイムアウト]の適切な値を選択します。 値の有効な範囲: 1～65535 (ms)
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、Send_Config.RDREC.STATUS/ Receive_Config.RDREC.STATUS 静的パラメータまたは RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、Send_Config.WRREC.STATUS/ Receive_Config.WRREC.STATUS 静的パラメータまたは WRREC.STATUS と、SFB WRREC の説明を参照してください。
16#8282	モジュールが使用不可	PORT パラメータの入力をチェックし、モジュールに確実にアクセスできるようにします。
構成エラー - Modbus_Slave		
16#8186	スレーブアドレスが無効	MB_ADDR パラメータで、適切なスレーブアドレスを選択します。 以下が有効です。 標準アドレス領域では 1～247; 拡張アドレス領域では 1～65535

		(0 はブロードキャスト用に予約済みです)
16#8187	MB_HOLD_REG パラメータの無効な値	MB_HOLD_REG パラメータで、保持レジスタの適切な値を選択します。
16#8188	動作モードまたはブロードキャストが無効(MB_ADDR = 0)および MODE パラメータ ≠ 1	ブロードキャストモードで MODE の値 1 を選択するか、または 異なる動作モードを選択します。
16#818C	MB_HOLD_REG 領域へのポインタは、データブロックまたはビットメモリアドレス領域であることが必要です。	MB_HOLD_REG 領域へのポインタの適切な値を選択します。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.RDREC.STATUS または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.WRREC.STATUS または Receive_P2P.WRREC.STATUS と、SFB WRREC の説明を参照してください。
16#8452 1)	MB_HOLD_REG が、DB またはビットメモリ領域へのポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8453 1)	MB_HOLD_REG が、タイプ BOOL または WORD のポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8454 1)	MB_HOLD_REG でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が読み出しまたは書き込みされるデータバイトの数に対して小さすぎます。	MB_HOLD_REG ポインタをチェックします
16#8455 1)	MB_HOLD_REG が書き込み禁止 DB を指しています	MB_HOLD_REG ポインタをチェックします
16#8456 1)	命令実行中のエラー エラーの原因は、STATUS パラメータに示されます。	SFCSTATUS パラメータの値を識別します。 SFC51、STATUS パラメータの説明でこの値が意味することをチェックします。
構成エラー - Modbus_Master		
16#8180	MB_DB パラメータの値が無効	Modbus_Comm_Load 命令で MB_DB(インスタンスデータ DB)に構成定義された値が有効ではありません。 Modbus_Comm_Load 命令とそのエラーメッセージの相互接続をチェックします。
16#8186	ステーションアドレスが無効	MB_ADDR パラメータで、適切なステーションアドレスを選択します。 以下が有効です。 標準アドレス領域では 1～247; 拡張アドレス領域では 1～65535 (0 はブロードキャスト用に予約済みです)

16#8188	動作モードまたはブロードキャストが無効(MB_ADDR = 0)および MODE パラメータ ≠ 1	ブロードキャストモードで MODE の値 1 を選択するか、または異なる動作モードを選択します。
16#8189	データアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。 情報システムの説明 Modbus_Master を選択します。
16#818A	長さが無効	DATA_LEN パラメータで、適切なデータ長を選択します。 情報システムの説明 Modbus_Master を選択します。
16#818B	DATA_PTR の値が無効	DATA_PTR パラメータで、データポインタの適切な値を選択します(M または DB アドレス)。 情報システムの説明 Modbus_Master を選択します。
16#818C	DATA_PTR パラメータの相互接続エラー	命令の相互接続をチェックしてください。
16#818D	DATA_PTR でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が 読み出しまたは書き込みされるデータバイトの数 に対して小さすぎます。	DATA_PTR ポインタをチェックします。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Send_P2P.RDREC.STATUS または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。. エラーの原因の詳細については、静的パラメータ Send_P2P.WRREC.STATUS、Receive_P2P.WRREC.STATUS 、または Receive_Reset と、SFB WRREC の説明を参照してください。
通信エラー - Modbus_Master および Modbus_Slave		
16#80D1	XON または CTS = ON の待機時間が経過しました。	通信パートナーで障害が発生したか、速度が遅すぎるか、またはオフラインになっています。 通信パートナーをチェックするか、または必要に応じてパラメータを変更します。
16#80D2	「ハードウェア RTS は常に ON」: DSR = ON から OFF への切り替えのために送信ジョブがキャンセルされました	通信パートナーをチェックします。 伝送時間全体にわたって、DSR が確実に ON になるようにします。
16#80E0	フレームが中断されました: 送信バッファオーバーフロー/送信フレームが長すぎます	ユーザープログラムで命令を呼び出す回数を増やすか、またはデータフロー制御による通信を構成定義します。

16#80E1	フレームが中断されました: パリティエラー	通信パートナーの接続線をチェックするか、または両方のデバイスが同じデータ伝送率、パリティ、ストップビット数に構成定義されているかどうかを検証します。
16#80E2	フレームが中断されました: キャラクタフレームエラー	スタートビット、データビット、パリティビット、データ伝送率、ストップビットの設定をチェックします。
16#80E3	フレームが中断されました: キャラクタオーバーフロエラー	通信パートナーのフレーム内のデータ数をチェックします。
16#80E4	フレームが中断されました: 最大フレーム長に達しました	通信パートナー側で短いフレーム長を選択します。 以下が有効です(モジュールによって異なります): 1 ~ 1024/2048/4096 (バイト)
通信エラー - Modbus_Master		
16#80C8	スレーブが設定された時間内に応答しません。	スレーブのデータ伝送率、パリティ、および配線をチェックします。
16#80C9	スレーブが Blocked_Proc_Timeout によって設定された時間内に応答しません。	Blocked_Proc_Timeout の設定をチェックします。 モジュールが、Modbus_Comm_Load 命令を使用して構成定義されたかどうかをチェックします。 モジュールは、抜き差し後または電圧復旧後に Modbus_Comm_Load を使用して再構成しなければならない場合があります。
16#8200	インターフェースが出力要求でビジーになっています。	後からコマンドを繰り返します。 新しいコマンドを開始する前に、まだ実行中のコマンドが存在しないことを確認します。
プロトコルエラー - Modbus_Slave (Modbus をサポートしている通信モジュールの場合のみ)		
16#8380	CRC エラー	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#8381	ファンクションコードがサポートされていない、またはブロードキャストでサポートされていません。	通信パートナーをチェックして、有効なファンクションコードが送信されていることを確認します。
16#8382	要求フレーム内の情報の長さが無効	DATA_LEN パラメータで、適切なデータ長を選択します。
16#8383	要求フレーム内のデータアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。
16#8384	要求フレーム内のデータ値無効エラー	Modbus マスタの要求フレーム内のデータ値をチェックします。
16#8385	診断値が Modbus スレーブでサポートされません(ファンクションコード 08)	Modbus スレーブは、診断値 16#0000 および 16#000A のみをサポートします。
プロトコルエラー - Modbus_Master (Modbus をサポートしている通信モジュールの場合のみ)		
16#8380	CRC エラー	Modbus フレームのチェックサムエラー. 通信パートナーをチェックします。
16#8381	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: ファンクションコードがサポートされていません。	通信パートナーをチェックして、有効なファンクションコードが送信されていることを確認します。

16#8382	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 長さが無効	適切なデータ長を選択します。
16#8383	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 要求フレーム内のデータアドレスが無効	DATA_ADDR パラメータで、データアドレスの適切な値を選択します。
16#8384	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: データ値エラー	Modbus スレーブへの要求フレームをチェックします。
16#8385	Modbus Slave からの応答フレームに以下のエラーメッセージがあります: 診断値が Modbus スレーブでサポートされていません。	Modbus スレーブは、診断値 16#0000 および 16#000A のみをサポートします。
16#8386	戻されたファンクションコードが要求されたファンクションコードと一致しません。	スレーブの応答フレームとアドレス指定をチェックします。
16#8387	応答を要求されなかったスレーブ	スレーブの応答フレームをチェックします。 スレーブのアドレス設定をチェックします。
16#8388	書き込み要求に対するスレーブの応答のエラー	スレーブの応答フレームをチェックします。
16#8828 [1)]	DATA_PTR が、n * 8 と等しくないビットアドレスを指しています	DATA_PTR ポインタをチェックします。
16#8852 [1)]	DATA_PTR が、DB またはビットメモリ領域へのポインタではありません	DATA_PTR ポインタをチェックします。
16#8853 [1)]	DATA_PTR が、タイプ BOOL または WORD のポインタではありません	DATA_PTR ポインタをチェックします。
16#8855 [1)]	DATA_PTR が書き込み禁止 DB を指しています	DATA_PTR ポインタをチェックします。
16#8856 [1)]	SFC51 呼び出し中のエラー	Modbus_Master 命令を再度呼び出してください。
エラー - Modbus_Slave (Modbus をサポートしている通信モジュールの場合のみ)		
16#8428 [1)]	MB_HOLD_REG が、n * 8 と等しくないビットアドレスを指しています	MB_HOLD_REG ポインタをチェックします
16#8452 [1)]	MB_HOLD_REG が、DB またはビットメモリ領域へのポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8453 [1)]	MB_HOLD_REG が、タイプ BOOL または WORD のポインタではありません	MB_HOLD_REG ポインタをチェックします
16#8454 [1)]	MB_HOLD_REG でアドレス指定された領域が DB よりも長いか、またはアドレス指定された領域が読み出しまたは書き込みされるデータバイトの数に対して小さすぎます。	MB_HOLD_REG ポインタをチェックします
16#8455 [1)]	MB_HOLD_REG が書き込み禁止 DB を指しています	MB_HOLD_REG ポインタをチェックします
16#8456 [1)]	SFC51 呼び出し中のエラー	Modbus_Slave 命令を再度呼び出してください。
[1)] S7-300/400 CPU 用の命令の場合のみ		

エラーメッセージの概要 - USS

エラーコード	説明	対策
16#0000	エラーはありません。	-
16#8180	ドライブの応答の長さエラー:	ドライブの応答フレームをチェックします。
16#8181	データタイプエラー	適切なデータタイプを選択します。 以下が有効です。 • REAL • ワード • ダブルワード
16#8182	データタイプエラー:「ダブルワード」または「実数」を「ワード」要求に対して返すことはできません。	ドライブの応答フレームをチェックします。
16#8183	データタイプエラー:「ワード」を「ダブルワード」または「実数」要求に対して返すことはできません。	ドライブの応答フレームをチェックします。
16#8184	ドライブの応答のチェックサムエラー:	ドライブと通信接続をチェックします。
16#8185	アドレス指定エラー	有効なドライブアドレス範囲: 1 ~ 16
16#8186	セットポイントエラー	有効なセットポイント範囲: -200 % ~ +200 %
16#8187	不正なドライブ番号が返されました	ドライブの応答フレームをチェックします。
16#8188	PZD 長さが無効	許可されている PZD 長さ: 2、4、6、8 ワード
16#8189	モジュールがこのデータ伝送率をサポートしていません。	モジュールの有効なデータ伝送率を選択します。
16#818A	このドライブに対する別の要求が現在有効です。	後から、パラメータ読み出しまたは書き込みコマンドを繰り返します。
16#818B	ドライブが応答しません。	ドライブをチェックします。
16#818C	ドライブがパラメータ要求に対してエラーメッセージで応答しました	ドライブの応答フレームをチェックします。 パラメータ要求をチェックします。 命令 USS_Read_Param、USS_Write_Param または USS_Port_Scan にエラーの報告があるかチェックします。 エラーの報告がある場合、USS_Drive_Control 命令の静的タグ USS_DB. w_USSExtendedError の値をチェックします。
16#818D	ドライブがパラメータ要求に対してアクセスエラーメッセージで応答しました	ドライブの応答フレームをチェックします。 パラメータ要求をチェックします。
16#818E	ドライブが初期化されていません。	ユーザープログラムをチェックして、USS_Drive_Control 命令がこのドライブに

		対して呼び出されていることを確認します。
16#8280	モジュールの読み出し時の否定確認	PORT パラメータの入力をチェックします。 エラーの原因の詳細については、静的パラメータ Port_Config.RDREC.STATUS、Send_Config.RDREC.STATUS 、 Receive_Config.RDREC.STATUS 、Send_P2P.RDREC.STATUS 、または Receive_P2P.RDREC.STATUS と、SFB RDREC の説明を参照してください。
16#8281	モジュールの書き込み時の否定確認	PORT パラメータの入力をチェックします。. エラーの原因の詳細については、静的パラメータ Port_Config.WRREC.STATUS、Send_Config.WRREC.STATUS 、 Receive_Config.WRREC.STATUS 、Send_P2P.RDREC.STATUS 、または Receive_P2P.RDREC.STATUS と、SFB WRREC の説明を参照してください。
[1] S7-300/400 CPU 用の命令の場合のみ		

ポイントツーポイント

この章には下記に関する情報が記載されています：

PORT_CFG: 動的に通信パラメータを設定

説明

「PORT_CFG」命令を使用して、ポイントツーポイント通信ポートの通信パラメータを動的に設定することができます。

ハードウェアコンフィグレーションで、ポートのオリジナルの静的構成を設定します。「PORT_CFG」命令を実行することによって、この構成を変更できます。 また、このファンクションを使用して、作成済みブロックをライブラリに保存し、再使用時にハードウェアコンフィグレーションでの構成を避けることもできます。

「PORT_CFG」を使用して、次の通信パラメータ設定に影響を与えることができます。

- パリティ
- ボーレート
- 文字当たりのビット数
- ストップビット数
- フロー制御のタイプとプロパティ

「PORT_CFG」命令で行われた変更は、ターゲットシステムに恒久的には格納されません。

シリアルデータは、電気的接続の RS-232 (半二重および全二重)および RS-485 (半二重)で転送することができます。

パラメータ

次の表に、「PORT_CFG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで構成の変更を有効にします。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
PROTO-COL	Input	UINT	I、Q、M、D、L、または定数	伝送プロトコル: • 0: ポイントツーポイント通信プロトコル • 1..n: 特定の伝送プロトコルの将来の定義
BAUD	Input	UINT	I、Q、M、D、L、または定数	ポートのボーレート: • 1: 300 ボー • 2: 600 ボー • 3: 1200 ボー • 4: 2400 ボー • 5: 4800 ボー • 6: 9600 ボー(デフォルト)

				• 7: 19200 ボー • 8: 38400 ボー • 9: 57600 ボー • 10: 76800 ボー • 11: 115200 ボー
PARITY	Input	UINT	I、Q、M、D、L、または定数	ポートのパリティ: • 1: パリティなし(デフォルト) • 2: 偶数パリティ • 3: 奇数パリティ • 4: マークパリティ • 5: スペースパリティ
DATABITS	Input	UINT	I、Q、M、D、L、または定数	文字当たりのビット数: • 1: 文字当たり 8 ビット(デフォルト) • 2: 文字当たり 7 ビット
STOPBITS	Input	UINT	I、Q、M、D、L、または定数	ストップビット数: • 1: 1 ストップビット(デフォルト) • 2: 2 ストップビット
FLOWCTRL	Input	UINT	I、Q、M、D、L、または定数	データフロー制御 • 1: なし(デフォルト) • 2: XON/XOFF • 3: ハードウェアフロー制御((RTS は常に有効) • 4: ハードウェアフロー制御(送信中に RTS を無効にすることが可能)
XONCHAR	Input	CHAR	I、Q、M、D、L、または定数	XON 文字として使用される文字を示します。 文字 DC1 (11H)がデフォルトとして設定されます。
XOFF-CHAR	Input	CHAR	I、Q、M、D、L、または定数	XOFF 文字として使用される文字を示します。 文字 DC3 (13H)がデフォルトとして設定されます。
WAITTIME	Input	UINT	I、Q、M、D、L、または定数	送信開始後の XON または CTS に対する待ち時間を指定します。 指定された値は、デフォルトとして設定されている 0. 2000 ミリ秒を超えてはなりません。
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#...)	説明
80A0	指定されたプロトコルが無効です。
80A1	指定されたボーレートが無効です。
80A2	指定されたパリティレートが無効です。
80A3	指定された文字当たりのビット数が無効です。
80A4	指定されたストップビット数が無効です。
80A5	指定されたフロー制御のタイプが無効です。
80A6	WAITTIME パラメータの値が正しくありません。 データフロー制御が有効の場合、WAITTIME パラメータの値は 0 よりも大きくなければなりません。
80A7	XONCHAR および XOFFCHAR パラメータの値が無効です。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "通信ブロックの全般的なステータス情報".

SEND_CFG: 動的にシリアル伝送パラメータを設定

説明

「SEND_CFG」命令を使用して、ポイントツーポイント通信ポートのシリアル伝送パラメータを動的に設定することができます。 転送待機中のすべてのメッセージは、SEND_CFG の実行後に破棄されます。

ハードウェアコンフィグレーションで、ポートのオリジナルの静的構成を設定します。「SEND_CFG」命令を実行することによって、この構成を変更できます。 また、このファンクションを使用して、作成済みブロックをライブラリに保存し、再使用時にハードウェアコンフィグレーションでの構成を避けることもできます。 「SEND_CFG」を使用して、次の伝送パラメータ設定に影響を与えることができます。

- RTS (Request to Send)の有効化から伝送の開始までの時間
- 伝送の終了から RTS の無効化までの時間
- ブレークのビット時間の定義

「SEND_CFG」命令で行われた変更は、ターゲットシステムに恒久的には格納されません。

シリアルデータは、電気的接続の RS-232 (半二重および全二重)および RS-485 (半二重)で転送することができます。

パラメータ

次の表に、「SEND_CFG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで構成の変更を有効にします。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
RTSONDLY	Input	UINT	I、Q、M、D、L、または定数	RTS が有効になってから伝送が開始されるまでの経過時間。 このパラメータに有効な値は、次の通りです。 • 0 (デフォルト) • 1 ms 単位で 0～65535 ms このパラメータは、RS-485 モジュールには適用されません。
RTSOFFDLY	Input	UINT	I、Q、M、D、L、または定数	伝送が終了してから RTS が無効になるまでの経過時間。 このパラメータに有効な値は、次の通りです。 • 0 (デフォルト) • 1 ms 単位で 0～65535 ms

				このパラメータは、RS-485 モジュールには適用されません。
BREAK	Input	UINT	I、Q、M、D、L、または定数	メッセージの開始時に送信される、改行のビット時間を指定します。 デフォルトとして、12 ビット時間が設定されます。 最大で 25000 ビット時間を指定することができます。
IDLELINE	Input	UINT	I、Q、M、D、L、または定数	メッセージの開始時の改行後のアイドルラインのビット時間を指定します。 デフォルトとして、12 ビット時間が設定されます。 最大で 25000 ビット時間を指定することができます。
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード * (W#16#...)	説明
80B0	送信中断の構成は行えません。
80B1	指定された改行時間が、最大許容の 25000 ビット時間を超えています。
80B2	アイドルラインに指定された時間が、最大許容の 25000 ビット時間を超えています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "通信ブロックの全般的なステータス情報".

RCV_CFG: 動的にシリアル受信パラメータを設定

説明

「RCV_CFG」命令を使用して、ポイントツーポイント通信ポートのシリアル受信パラメータを動的に設定することができます。 この命令を使用して、メッセージ送信の開始と終了を指定する条件を構成できます。 この条件に対応するメッセージの受信は、「RCV_PTP 」命令で有効にすることができます。

ハードウェアコンフィグレーションのプロパティで、ポートのオリジナルの静的な構成を行います。構成を変更するには、プログラムで「RCV_CFG」命令を実行します。 また、このファンクションを使用して、作成済みブロックをライブラリに保存し、再使用時にハードウェアコンフィグレーションでの構成を避けることもできます。 「RCV_CFG」命令で行われた変更は、ターゲットシステムに恒久的には格納されません。

転送待機中のすべてのメッセージは、「RCV_CFG」命令の実行後に破棄されます。

パラメータ

次の表に、「RCV_CFG」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで構成の変更を有効にします。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
CONDI-TIONS	Input	CONDI-TIONS	D、L	データ伝送の開始および終了条件を定義するデータ構造体。
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

データタイプ CONDITIONS

CONDITIONS 構造体を使用して、メッセージ送信の開始および終了条件を定義できます。 構造体 CONDITIONS は、「RCV_CFG」命令のインスタンス DB に含まれています。 構造体 CONDITIONS を使用して、メッセージの送信がいつ完了するか、また、次のメッセージ転送をいつ開始するかという開始条件および終了条件を定義します。

- START 構造体でデータ転送の開始条件を定義します。
- END 構造体でデータ転送の終了条件を定義します。

1 つまたは複数の開始および終了条件を定義することができます。 複数の開始または終了条件を指定した場合、条件は OR 論理命令でリンクされます。

次の表に、「CONDITIONS」構造体を示します。

パラメータ		データタイプ	説明
START		STRUCT	開始条件
	START-COND	UINT	開始条件を指定します(詳細については、下記を参照)。 開始条件を 16 ビットの 16 進値で指定できます。 開始条件に使用可能な値は次の通りです。 • 1: 開始文字 • 2: 任意の文字(デフォルト) • 4: 改行 • 8: アイドルライン • 16: 文字列 1 • 32: 文字列 2 • 64: 文字列 3 • 128: 文字列 4 STARTCOND パラメータで複数の開始条件を定義することもできます。 この場合は、個々の条件の値の合計を指定します。 たとえば、「アイドルライン」OR「文字列 1」OR「文字列 4」を開始条件として定義したい場合、値「152」を指定する必要があります。
	IDLETIME	UINT	受信が開始される前の行の最大アイドル時間を指定します。 このパラメータに有効な値は、次の通りです。 • 40 ビット時間(デフォルト) • 0～2500 ビット時間
	STARTCH-AR	BYTE	開始文字を指定します。 この設定は、構成された開始条件が「開始文字」の場合にのみ有効です。 このパラメータに有効な値は、次の通りです。 • 02 (STX): デフォルト設定 • B#16#00 ～ B#16#FF
	SEQ[1].CTL	BYTE	文字列 1: 文字ごとのコントロールシーケンス 文字のビット位置を使用して、文字列のどの文字を考慮または無視するかを定義することができます。 文字を評価するには、対応するビットを設定する必要があります。 • ビット 0: 1 文字 • ビット 1: 2 文字 • ビット 2: 3 文字 • ビット 3: 4 文字

			• ビット 4: 5 文字 対応するビットがリセットされると、文字は無視されます。
	SEQ[1].STR	CHAR[5]	文字列 1: 開始文字(5 文字)
	SEQ[2].CTL	BYTE	文字列 2: 文字ごとのコントロールシーケンスを無視/比較します。
	SEQ[2].STR	CHAR[5]	文字列 2: 開始文字(5 文字)
	SEQ[3].CTL	BYTE	文字列 3: 文字ごとのコントロールシーケンスを無視/比較します。
	SEQ[3].STR	CHAR[5]	文字列 3: 開始文字(5 文字)
	SEQ[4].CTL	BYTE	文字列 4: 文字ごとのコントロールシーケンスを無視/比較します。
	SEQ[4].STR	CHAR[5]	文字列 4: 開始文字(5 文字)
END		STRUCT	終了条件
	ENDCOND	UINT	終了条件を指定します(詳細については、下記を参照)。 終了条件を 16 ビットの 16 進値として指定することができます。終了条件に使用可能な値は以下の通りです。 • 1: 応答タイムアウト • 2: メッセージタイムアウト • 4: 文字列内でのタイムアウト • 8: 最大長 • 16: N+LEN+M。メッセージの長さに関する情報がメッセージに統合され、評価されます。 • 32: 文字列 1 ENDCOND パラメータで複数の終了条件を定義することもできます。 この場合は、個々の終了条件の値の合計を指定します。 たとえば、終了条件「最大長」OR「シーケンス 1」を定義したい場合は、値「40」を指定する必要があります。
	MAXLEN	UINT	メッセージ内の最大文字数を指定します。 このパラメータに有効な値*は、次の通りです。 • 1 文字(デフォルト) • 0 ~ 1024 文字 この設定は、「最大長」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
	N	UINT	メッセージ内の長さフィールドのオフセット このパラメータに有効な値は、次の通りです。 • 0 文字(デフォルト) • 0 ~ 1024 文字 この設定は、「N+LEN+M」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
	LENGTH-SIZE	UINT	長さフィールドのバイト単位のサイズ このパラメータに有効な値*は、次の通りです。 • 0 バイト(デフォルト) • 1 バイト • 2 バイト • 4 バイト

		この設定は、「N+LEN+M」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
LENGTHM	UINT	長さフィールドの後に続いているが、メッセージの長さには含まれない終了文字の数を指定します。 このパラメータに有効な値は、次の通りです。 • 0 文字(デフォルト) • 0～255 文字 この設定は、「N+LEN+M」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
RCVTIME	UINT	メッセージの最初の文字の受信について最大継続時間を指定します。 このパラメータに有効な値は、次の通りです。 • 200 ms (デフォルト) • 1 ms 単位で 0～65535 ms この設定は、「応答タイムアウト」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
MSGTIME	UINT	メッセージの受信の最大継続時間を指定します。 このパラメータに有効な値は、次の通りです。 • 200 ms (デフォルト) • 1 ms 単位で 0～65535 ms この設定は、「メッセージタイムアウト」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
CHARGAP	UINT	連続する文字と文字の間の受信時間間隔を指定します。 このパラメータに有効な値は、次の通りです。 • 12 ビット時間(デフォルト) • 0～2500 ビット時間 この設定は、「文字列内でのタイムアウト」終了条件が ENDCOND パラメータで設定されている場合にのみ有効です。
SEQ.CTL	BYTE	文字列: 文字ごとのコントロールシーケンス 文字のビット位置を使用して、文字列のどの文字を考慮または無視するかを定義することができます。 文字を評価するには、対応するビットを設定する必要があります。 • ビット 0: 1 文字 • ビット 1: 2 文字 • ビット 2: 3 文字 • ビット 3: 4 文字 • ビット 4: 5 文字 対応するビットがリセットされると、文字は無視されます。
SEQ.STR	CHAR[5]	文字列: 開始文字(5 文字)
* 上記の値の範囲は、メッセージの終了を指定するための、対応するハードウェアの設定にも適用されます。		

メッセージ受信の開始条件(STARTCOND パラメータ)

メッセージの開始は、構成された開始条件が適用される場合に受信側に認識されます。 次の条件が、メッセージ受信の開始条件として定義可能です。

- 開始文字: メッセージの開始は、特定の文字が現れた時に認識されます。 この文字は、メッセージの最初の文字として保存されます。 開始文字の前に受信された文字はすべて拒否されます。
- 任意の文字: 任意の文字をメッセージの開始として定義できます。 この文字は、メッセージの最初の文字として保存されます。
- 改行: メッセージの開始は、受信したデータストリームに 1 文字の長さよりも長い中断があった場合に認識されます。
- アイドルライン: 送信伝送ラインが特定の時間(ビット時間で指定)アイドル状態になり、その後に文字の新たな伝送が続いたときに、メッセージの開始が認識されます。
- 文字列(シーケンス): データストリームに指定された文字シーケンスが現れたときに、メッセージの開始が認識されます。 最大で 4 つまでの文字シーケンスをそれぞれ 5 文字までで指定することができます。

  例: 受信した 16 進メッセージに次の文字が含まれています。 「68 10 aa 68 bb 10 aa 16」 設定済みの開始文字シーケンスを以下の

  表に示します。 最初の 68H 文字が正常に受信されると、開始文字シーケンスが評価されます。

  4 番目の文字(

  2 番目の 68H)が正常に受信された後に、開始条件「1」が満たされます。 開始条件が満たされると、

  終了条件の評価が開始します。

  開始文字シーケンスの処理は、各種パリティエラー、フレームエラー、文字間の時間間隔エラーが原因で終了する可能性があります。

  開始条件が満たされていないため、これらのエラーによって、メッセージを受信できなくなります。

開始条件	最初の文字	最初の文字+1	最初の文字+2	最初の文字+3	最初の文字+4
1	68H	xx	xx	68H	xx
2	10H	aaH	xx	xx	xx
3	dcH	aaH	xx	xx	xx
4	e5H	xx	xx	xx	xx

メッセージ受信の終了条件(ENDCOND パラメータ)

メッセージの開始は、構成された終了条件が適用される場合に受信側に認識されます。 次の条件が、メッセージ受信の終了条件として定義可能です。

- 応答タイムアウト: 文字の受信について指定された最大継続時間を超えると、メッセージの受信が終了します。 最大継続時間は、RCVTIME パラメータで定義されます。 定義された時間は、最後の送信が完了して RCV_PTP 命令によってメッセージの受信が可能になると、カウントダウンが直ちに開始されます。 定義された時間(RCVTIME)内に文字が受信されなかった場合、RCV_PTP 命令によって エラーがレポートされます。
- メッセージタイムアウト: メッセージの受信について指定された最大継続時間を超えると、メッセージの受信が終了します。 最大継続時間は、MSGTIME パラメータで定義されます。 メッセージの最初の文字が受信されると、定義された時間のカウントダウンが直ちに開始されます。

- 文字列内でのタイムアウト: 2 つの連続する文字の受信間の時間間隔が CHARGAP パラメータの値よりも長くなると、メッセージの受信が終了します。
- 最大長: MAXLEN パラメータで定義されたメッセージの長さを超えると、メッセージの受信が終了します。
- 読み出しメッセージの長さ(N+LEN+M): 特定のメッセージ長に達したときに、メッセージの受信が終了します。 この長さは、以下のパラメータの値で計算されます。
  - N: 長さフィールドが始まる、メッセージ内の文字の位置。
  - LENGTHSIZE: 長さフィールドのバイト単位のサイズ。
  - LENGTHM: 長さフィールドの後に続く終了文字の数。 この文字は、メッセージ長の計算で考慮されません。
- 文字列: 定義された文字シーケンスが受信されたときに、メッセージの受信が終了します。 文字列には、最大で 5 文字を入れることができます。 文字列の各文字ごとに、ビット位置を使用して、この文字を計算で考慮するか無視するかを定義することができます。

STATUS パラメータ

エラーコード* (W#16#...)	説明
80C0	開始条件のエラー
80C1	• 終了条件のエラー • 終了条件が定義されていません。
80C2	割り込みの受信が有効です。
80C3	0 に等しいか 4132 よりも大きい値が MAXLEN パラメータに入力される一方で、「最大長」終了条件が設定されています。
80C4	4131 よりも大きい値が N パラメータに入力される一方で、「N+LEN+M」終了条件が設定されています。
80C5	0 に等しいか無効な値が LENGTHSIZE パラメータに入力される一方で、「N+LEN+M」終了条件が設定されています。
80C6	255 よりも大きい値が LENGTHM パラメータに入力される一方で、「N+LEN+M」終了条件が設定されています。
80C7	4132 よりも大きいメッセージ長が計算される一方で、「N+LEN+M」終了条件が設定されています。
80C8	0 に等しい値が RCVTIME パラメータに入力される一方で、「応答タイムアウト」終了条件が設定されています。
80C9	0 に等しいかまたは 2500 よりも大きい値が CHARGAP パラメータに入力される一方で、「文字列内でのタイムアウト」終了条件が設定されています。
80CA	0 に等しいかまたは 2500 よりも大きい値が IDLETIME パラメータに入力される一方で、「アイドルライン」開始条件が設定されています。
80CB	「文字列」が終了条件として設定されているにもかかわらず、文字列のすべての文字が「Don't care」としてマークされています。
80CC	「文字列」が開始条件として設定されているにもかかわらず、文字列のすべての文字が「Don't care」としてマークされています。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "通信ブロックの全般的なステータス情報".

SEND_PTP: 送信バッファデータの送信

説明

「SEND_PTP」命令を使用してデータの伝送を開始します。 「SEND_PTP」命令は、実際のデータ伝送を実行しません。 送信バッファのデータは、関連するポイントツーポイント接続用通信モジュール(CM)に送信されます。 CM が実際の送信を実行します。

パラメータ

次の表に、「SEND_PTP」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	このイネーブル入力信号の立ち上がり工ッジで要求された送信を有効にします。 バッファの内容がポイントツーポイント接続用通信モジュール(CM)に送信されます。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
BUFFER	Input	VARIANT	I、Q、M、D、L、または定数	送信バッファの開始アドレスへのポインタ。 ブール値または Array of BOOL はサポートされていません。
LENGTH	Input	UINT	I、Q、M、D、L、または定数	送信バッファの長さ
PTRCL	Input	BOOL	I、Q、M、D、L、または定数	このパラメータは、接続されている CM に実装される通常のポイントツーポイント通信プロトコルまたは特定の Siemens プロトコルについてバッファを選択します。 FALSE = ユーザープログラムによって制御されるポイントツーポイント操作(唯一の有効なオプション)
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード * (W#16#...)	説明
7000	送信操作が行われていません。
7001	送信操作が最初の呼び出しを処理中です。
7002	送信操作が後続の呼び出し(最初の呼び出しに続く問い合わせ)を処理中です。
8080	通信ポート番号について入力された識別子が無効です。
8088	LENGHT パラメータの長さが、送信データの長さと一致しません。 関連項目 LENGHT および BUFFER パラメータ。
80D0	送信の実行中に、新しい送信要求が受信されました。
80D1	指定された待機時間内に CTS 信号が確認されなかったため、送信が中断されました。
80D2	通信パートナー(DCE)から受信を望まないことを信号で通知されたため(DSR)、送信要求が中断されました。
80D3	待機ループの最大サイズを超えた(1024 バイトを超える)ため、送信要求が中断されました。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "通信ブロックの全般的なステータス情報".

LENGTH および BUFFER パラメータ

「PTP_SEND」命令によって送信できる最小データサイズは、1 バイトです。 BUFFER パラメータは、送信されるデータのサイズを定義します。 BUFFER パラメータには BOOL も Array of BOOL データタイプも使用できません。

LENGTH パラメータ	BUFFER パラメータ	説明
LENGTH = 0	未使用	BUFFER パラメータで定義された完全なデータが送信されます。 LENGTH = 0 の場合、転送されるバイト数を指定する必要はありません。
LENGTH > 0	基本データタイプ	LENGTH 値には、このデータタイプのバイト数を含める必要があります。 それ以外の場合、データは転送されず、エラー 8088 が出力されます。
	STRUCT	LENGTH 値には、構造体の完全なバイト長よりも小さいバイト数を含めることができます。 この場合、最初の LENGTH バイトのみが転送されます。
	ARRAY	LENGTH 値には、フィールドの完全なバイト長よりも小さいバイト数を含めることができます。 この場合、LENGTH バイト内に完全に収まるフィールド工レメントのみが転送されます。 LENGTH 値は、データ工レメントのバイト数の倍数である必要があります。 それ以外の場合、STATUS = 8088、ERROR = 1 となり、データは転送されません。
	STRING	文字列の最大長および文字列の実際の長さに関する情報とともに、文字シーケンスフォーマットの完全なメモリ配列が送信されます。

		LENGTH 値には、最大長、実際の長さおよび文字列の文字のバイトを含める必要があります。 データタイプ STRING では、すべての長さと文字のサイズは 1 バイトです。 文字列が BUFFER パラメータに使用された場合、LENGTH 値にも 2 つの長さフィールドについて 2 バイトを含める必要があります。

RCV_PTP: 受信メッセージの有効化

説明

RCV_PTP 命令を使用して、送信メッセージの受信を有効にします。 各メッセージを個別に有効にする必要があります。 メッセージが対応する通信パートナーによって確認応答済みの場合にのみ、送信データは受信領域内で使用可能です。

パラメータ

次の表に、「RCV_PTP」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
EN_R	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで受信を有効にします。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
BUFFER	Input	VAR-IANT	I、Q、M、D、L、または定数	受信バッファの開始アドレスをポイントします。 受信バッファで STRING タイプのタグを使用しないでください。
NDR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス
LENGTH	Output	UINT	I、Q、M、D、L	受信バッファ内のメッセージの長さ

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード * (W#16#....)	説明
80E0	受信バッファが一杯のため、メッセージの受信が終了しました。
80E1	パリティエラーの結果、メッセージの受信が終了しました。
80E2	フレームエラーの結果、メッセージの受信が終了しました。
80E3	オーバーフローエラーの結果、メッセージの受信が終了しました。

80E4	計算されたメッセージ長(N+LEN+M)が受信バッファのサイズを超えているため、メッセージの受信が終了しました。
8080	通信ポート番号について入力された識別子が無効です。
8088	データタイプ STRING が BUFFER パラメータを介して参照されています。
0094	最大文字長が受信されたため、メッセージの受信が終了しました。
0095	タイムアウトの結果、メッセージの受信が終了しました。
0096	文字列内でのタイムアウトのため、メッセージの受信が終了しました。
0097	応答タイムアウトの結果、メッセージの受信が終了しました。
0098	「N+LEN+M」の長さ条件が満たされる場合、メッセージの受信が終了しました。
0099	終了条件として定義されている文字列が受信されたため、メッセージの受信が終了しました。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "**通信ブロックの全般的なステータス情報**".

RCV_RST:受信バッファの削除

説明

「RCV_RST」命令を使用して、通信パートナーの受信バッファを削除します。

パラメータ

次の表に、「RCV_RST」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで受信バッファの削除を有効にします。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス 通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。"通信ブロックの全般的なステータス情報".

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

SGN_GET: RS-232 信号の問い合わせ

説明

「SGN_GET」命令を使用して、RS-232 通信モジュールの複数の信号の現在の状態を問い合わせます。

パラメータ

次の表に、「SGN_GET」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで問い合わせを有効にします。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID)
NDR	Output	BOOL	I、Q、M、D、L	新しいデータの送信が準備完了し、命令がエラーなしで実行された場合に、1 サイクルに対して設定されます。
DTR	Output	BOOL	I、Q、M、D、L	データ端末準備完了、モジュール準備完了
DSR	Output	BOOL	I、Q、M、D、L	データセット準備完了、通信パートナー準備完了
RTS	Output	BOOL	I、Q、M、D、L	送信要求、モジュールの送信準備完了
CTS	Output	BOOL	I、Q、M、D、L	送信可、通信パートナーはデータを受信可能(モジュールの RTS = ON への応答)。
DCD	Output	BOOL	I、Q、M、D、L	データキャリア検出、受信信号レベル
RING	Output	BOOL	I、Q、M、D、L	リング表示、着呼の表示
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード* (W#16#....)	説明
80F0	通信モジュールが RS-485 モジュールで、信号は使用できません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "**通信ブロックの全般的なステータス情報**".

SGN_SET: RS-232 信号の設定

説明

「SGN_SET」命令を使用して、RS-232 通信モジュールの出力信号のステータスを設定します。

パラメータ

次の表に、「SGN_SET」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L、または定数	信号立ち上がりエッジで操作を起動します。 初期値:FALSE
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID (HW-ID) 初期値: FALSE
SIGNAL	Input	BYTE	I、Q、M、D、L、または定数	設定する信号を指定します。 • 01H = RTS を設定 • 02H = DTR を設定 • 04H = DSR を設定 初期値: FALSE
RTS	Input	BOOL	I、Q、M、D、L、または定数	送信要求、モジュールの送信準備完了 初期値: FALSE
DTR	Input	BOOL	I、Q、M、D、L、または定数	データ端末準備完了、モジュール準備完了 初期値: FALSE
DSR	Input	BOOL	I、Q、M、D、L、または定数	データの設定準備完了(DCE タイプのインターフェースにのみ適用されます) 初期値: FALSE
DONE	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: ジョブがまだ開始されていないか、または実行中です。 • 1: ジョブはエラーなく実行されました 初期値: FALSE
ERROR	Output	BOOL	I、Q、M、D、L	以下の値を持つステータスパラメータ: • 0: エラーは発生していません。 • 1: エラーが発生しました。 初期値: FALSE
STATUS	Output	WORD	I、Q、M、D、L	命令のステータス 初期値:0

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

エラーコード * (W#16#...)	説明
80F0	通信モジュールが RS-485 モジュールで、信号は使用できません。
80F1	H/W フロー制御が有効になっているため、信号を設定できません。
80F2	モジュールが DTE デバイスであるため、DSR 信号を設定できません。
80F3	モジュールが DCE デバイスであるため、DTR 信号を設定できません。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

通信命令の一般エラーコードに関する詳細情報については、次の項目を参照してください。 "通信ブロックの全般的なステータス情報".

通信ブロックの全般的なステータス情報

通信ブロックの実行ステータスに関する全般情報

次の表に、通信ブロックの STATUS パラメータに出力できる全般情報を示します。

エラーコード* (W#16#....)	説明
0000	エラーは発生していません。
7000	ジョブ処理が行われていません。
7001	ジョブ処理の開始。 パラメータ BUSY = 1、DONE = 0
7002	中間呼び出し(REQ は対象外): 命令が既に有効です。BUSY の値は「1」です。
8x3A	パラメータ x のポインタが無効です。
8070	すべての内部インスタンスメモリが使用されます。
8080	通信ポートに入力された識別子が無効です。
8081	タイムアウト、モジュールエラー、内部エラー
8082	現在、パラメータ割り当てがバックグラウンドで実行されているため、パラメータ割り当てに失敗しました。
8083	バッファオーバーフロー: CM または CB が長さパラメータによって許可されているよりも長い受信メッセージを返しました。
8085	LENGHT パラメータで長さを指定しているときにエラーが発生しました。 指定された長さが「0」であるか、最大許容値よりも大きくなっています。
8090	メッセージ長が無効、モジュールが無効、メッセージが無効
8091	パラメータ設定メッセージ内のバージョンが正しくありません。
8092	パラメータ設定メッセージ内のレコード長が無効です。
* エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えについては、「関連項目」を参照してください。	

USS

この章には下記に関する情報が記載されています：

USS 命令の概要

概要

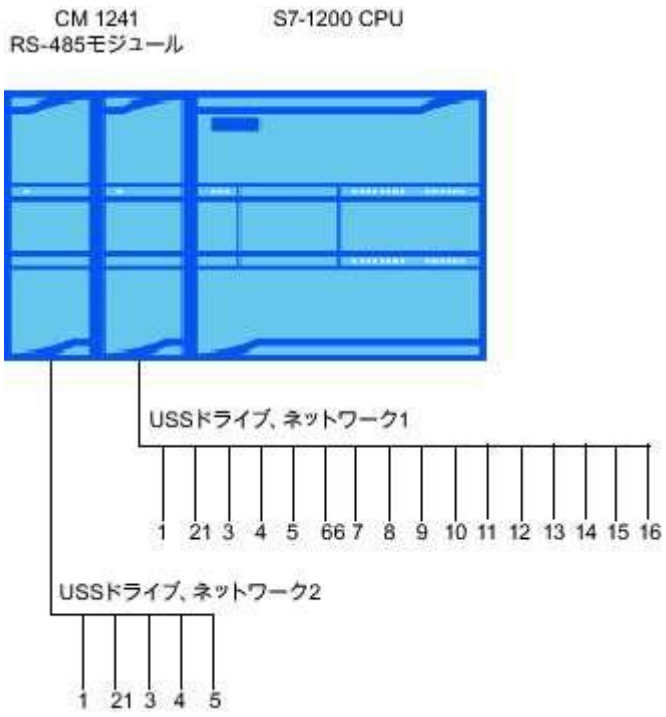
USS 命令は、ユニバーサルシリアルインターフェースをサポートするドライブ操作を制御します (USS)。 USS 命令を使用して、RS-485 接続を経由して複数のドライブと通信できます。

通信するには、CM 1241 RS-485 通信モジュールまたは CB 1241 RS-485 通信ボードが必要です。 S7-1200 CPU には、最大 3 つの CM 1241 RS-485 モジュールと 1 つの CB 1241 RS-485 をインストールできます。

各 RS-485 ポートは最大 16 ドライブを操作できます。

USS プロトコルは、シリアルバスを経由して通信にマスタ/スレーブネットワークを使用します。マスタはアドレスパラメータを使用してメッセージを選択されたスレーブに送信します。 スレーブ自体は前もって要求を受信せずに、送信することはありません。 スレーブ間のメッセージの直接交換はできません。 USS 通信は半二重モードで動作します。

次の図に、USS ネットワークダイアグラムの例を示します。



USS プロトコル使用の必要条件

ドライブ設定の一般的な必要条件

- 4 PKW ワードを使用するにはドライブを設定する必要があります。
- 2、4、6 または 8 PZD ワードに対して、ドライブを設定できます。
- ドライブの PZD ワード数は、ドライブの「USS_DRIVE」命令の PZD_LEN 入力と一致している必要があります。
- すべてのドライブのボーレートは、「USS_PORT」命令の BAUD 入力パラメータのボーレートと一致している必要があります。
- リモート制御にドライブを設定しなければなりません。
- USS は、ドライブの COM 接続では、目的の周波数値を指定する必要があります。
- ドライブアドレスとして 1~16 をセットしなければなりません。 このアドレスは、「USS_DRV」命令の DRIVE 入力パラメータのアドレスと一致している必要があります。
- ドライブの方向を制御するには、ドライブの極性に目的の値をセットしなければなりません。
- RS-485 ネットワークを正確に接続する必要があります。

定義: PKW / PZD 領域

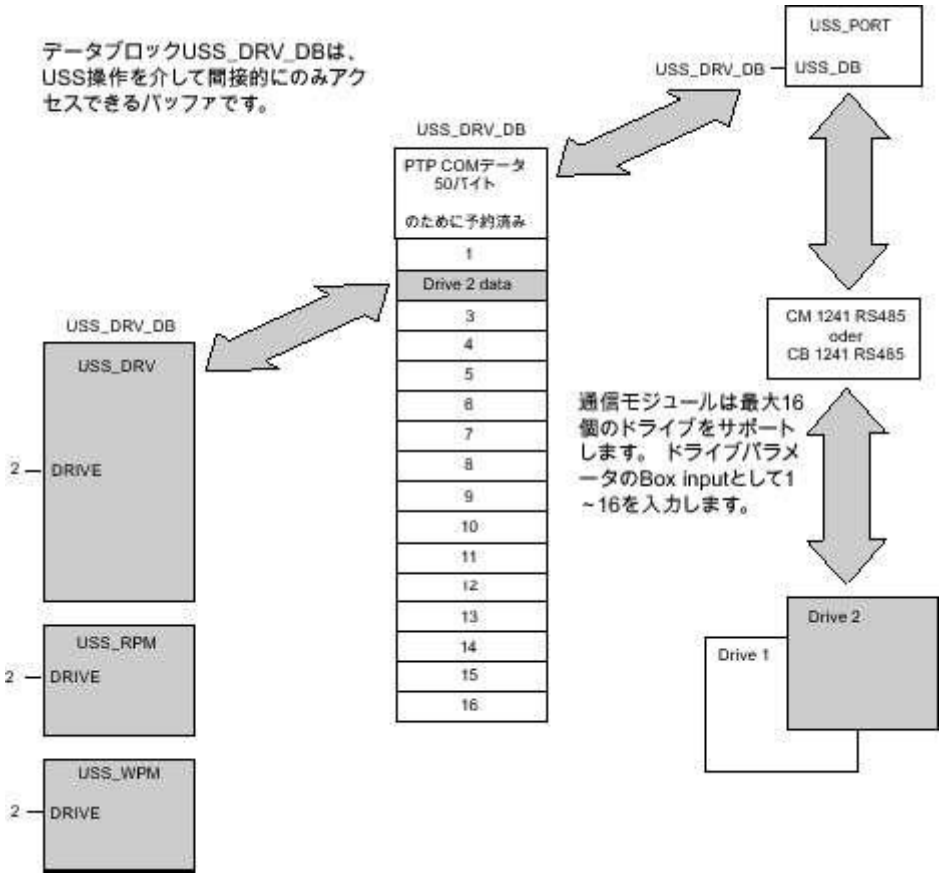
- PKW 領域は、パラメータ-識別子-値インターフェースの処理に関連します。

  PKW インターフェースは物理的インターフェースではありませんが、2 つの通信パートナー間のパラメータ交換を制御するメカニズムを示しています。言い換えれば、パラメータ値の読み出しと書き込み、パラメータの記述と対応するテキスト、および未処理メッセージのためのパラメータ変更の処理です。 PKW インターフェースを経由して処理されたタスクはすべて、基本的には、オペレータコントロールおよびモニタリング、サービスと診断のためのタスクです。
- PZD 領域には、自動化に必要なシグナルが含まれます。
  - コントロールワード、マスタからスレーブへのセットポイント
  - ステータスワード、スレーブからマスタへの現在値。

両方の領域は共に、結果としてユーザーデータフィールドになります。 これは、マスタによってジョブフレームとしてスレーブに転送され、スレーブによって応答フレームとしてマスタに転送されます。

説明

各通信モジュール CM 1241 RS485 は、最大 16 ドライブをサポートします。 単一のインスタンスデータブロックには、インストール済みの PtP 通信モジュールに接続される USS ネットワークのすべてのドライブに対する一時メモリとバッファファンクションが含まれます。 これらのドライブの USS 命令はこのデータブロックの情報へのアクセスを共有しています。



- RS485 ポートに接続されたすべてのドライブ(最大 16)は、同じ USS ネットワークの一部です。 異なる RS485 ポートに接続されあたすべてのドライブは、異なる USS ネットワークの一部です。 S7-1200 は最大 3 つの CM 1241 RS485 モジュールをサポートするので、最大 3 つの USS ネットワークを設定できます。それぞれ最大 16 ドライブを保持するので、合計 48 USS のドライブがサポートされます。
- 各 USS ネットワークは固有のデータブロックによって管理されます(3 つの CM 1241 RS485 モジュール付き 3 つの USS ネットワークには 3 つのデータブロックが必要です)。 USS ネットワークに属するすべての命令はこのデータブロックを共有しなければなりません。 これには、USS ネットワークのすべてのドライブを制御するために必要な、「USS_DRIVE」、「USS_PORT」、「USS_RPM」および「USS_WPM」命令がすべて含まれています。
- 「USS_DRIVE」命令はファンクションブロック(FB)です。 エディタで「USS_DRIVE」命令を挿入する場合、DB を命令に割り当てるか[呼び出しオプション]ダイアログで問われます。
  - この USS ネットワークのプログラム内で最初の「USS_DRIVE」命令の場合、デフォルトの DB 割り当てを使用することができ(必要な場合は名前を変更)、新規の DB が作成されます。
  - ただし、このネットワークの最初の「USS_DRIVE」命令でない場合、[呼び出しオプション]ダイアログのドロップダウンリストで、以前この USS ネットワークに割り当てられた DB の中から適切な DB を選択する必要があります。
- すべての「USS_PORT」、「USS_RPM」、および「USS_WPM」命令はファンクション(FC)です。 エディタでこれらの FC を挿入する場合、DB は割り当てられません。 代わりに、当該の DB をこれらの命令の USS_DB 入力に割り当てる必要があります(パラメータフィールドをダブルクリックし、次にアイコンをダブルクリックして、使用可能な DB を表示します)。
- 「USS_PORT」命令は、PtP 通信モジュールを経由して CPU とドライブ間の通信を制御します。 この命令を呼び出す時は必ず、ドライブによる通信が処理されます。 ドライブがタイムアウトをレポートしないように、このファンクションを素早く呼び出す必要があります。 この命令は、メインプログラムまたは任意の割り込み OB から呼び出すことができます。
- ファンクションブロック「USS_DRIVE」を使用すると、USS ネットワーク内の指定したドライブにアクセスできます。 入力と出力はドライブの状態およびオペレーティングファンクションに対

応します。ネットワーク内に 16 のドライブがある場合、プログラム内で「USS_DRIVE」を少なくとも 16 回呼び出す必要があります。つまりドライブごとに 1 回です。 ブロックをどの程度素早く呼び出す必要があるかは、ドライブファンクションを制御するのに必要な速度によって異なります。

「USS_DRIVE」命令は、メインプログラム OB からのみ呼び出すことができます。

注意

「USS_DRIVE」、「USS_RPM」、「USS_WPM」は、メインプログラム OB からのみ呼び出してください。「USS_PORT」命令は任意の OB から呼び出すことができますが、通常は遅延割り込み OB から呼び出されます。「USS_PORT」命令が実行中に割り込まれた場合、予期しないエラーが発生する可能性があります。

「USS_RPM」および「USS_WPM」命令は、ドライブのオペレーティングパラメータを読み出す/書き込むために使用されます。 これらのパラメータは、ドライブ操作の内部モードを制御します。 パラメータの定義はドライブマニュアルに記載されています。

プログラムにはこれらの命令の番号も含まれていることがあります。しかし、ドライブでは 1 つの読み出し要求または 1 つの書き込み要求のみ有効です。 メインプログラム OB から「USS_RPM」および「USS_WPM」命令のみ呼び出しできます。

ドライブとの通信時間の計算

ドライブとの通信は、S7-1200 のサイクルと非同期に実行されます。S7-1200 は複数のサイクルを実行してから、ドライブとの通信を完了します。

「USS_PORT」の間隔はドライブトランザクションに必要な時間です。 次の表に、ボーレートごとの「USS_PORT」の最小間隔を示します。 「USS_PORT」間隔よりも頻繁に「USS_PORT」を呼び出しても、トランザクション数は増えません。 ドライブのタイムアウト間隔とは、通信エラーのためトランザクションを完了するには 3 回の試行が必要な場合のトランザクションに使用できる時間です。 デフォルトでは、USS プロトコルを使用したトランザクションごとに最大 2 回の試行が行われます。

ボーレート	USS_PORT を呼び出すための計算済み最小間隔(ミリ秒)	ドライブあたりのドライブメッセージ間隔タイムアウト(ミリ秒)
1200	790	2370
2400	405	1215
4800	212.5	638
9600	116.3	349
19200	68.2	205
38400	44.1	133
57600	36.1	109
115200	28.1	85

USS_PORT: USS ネットワーク経由の通信編集

説明

「USS_PORT」命令は USS ネットワーク上の通信を処理します。 プログラムでは、PtP 通信ポートあたり 1 つの「USS_PORT」命令を使用して、1 つのドライブへの転送や 1 つのドライブからの転送を制御します。

1 つの USS ネットワークと 1 つの PtP 通信ポートに割り当てられるすべての USS 命令は、おなじインスタンスデータブロックを使用する必要があります。

呼び出し

ドライブのタイムアウトを防ぐため、「USS_PORT」命令を頻繁に実行しなければなりません。 したがって、サイクル割り込み OB から「USS_PORT」命令を呼び出し、最新の USS データを「USS_DRIVE」呼び出しに使用できるよう更新する必要があります。

パラメータ

次の表に、「USS_PORT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
PORT	Input	PORT	D、L または定数	PtP 通信ポート識別子 デフォルトタグテーブルの[定数]タブ内で参照できる定数。
BAUD	Input	DINT	I、Q、M、D、L または定数	USS 通信のボーレート。
USS_DB	InOut	USS_BASE	D	「USS_DRIVE」命令のインスタンス DB への参照
ERROR	Output	BOOL	I、Q、M、D、L	エラーが発生する場合、ERROR は TRUE にセットされる。 対応するエラーコードは STATUS 出力で出力されます。
STATUS	Output	WORD	I、Q、M、D、L	要求のステータス値。 サイクルまたは初期設定の結果を示す。 一部のステータスコードの追加情報については、「USS_Extended_Error」タグを参照してください。

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

USS_DRIVE: データとドライブをスワップ

説明

「USS_DRIVE」命令は、要求メッセージを作成し、ドライブ応答メッセージを割り込んでドライブとデータを交換します。 ドライブごとに個別の命令を使用する必要がありますが、1 つの USS ネットワークおよび 1 つの PtP 通信モジュールに割り当てられたすべての USS 命令は同じインスタンスデータブロックを使用しなければなりません。最初の「USS_DRIVE」命令を設定する時に DB 名を作成する必要があります。 初期命令が挿入された時に作成されたこの DB を再使用します。

「USS_DRIVE」命令が最初に実行される時、USS アドレス(パラメータ DRIVE)によって指示されたドライブは、インスタンス DB で初期化されます。 この初期設定の後、後続の「USS_PORT」命令は、このドライブ番号のドライブと通信を開始できます。

ドライブ番号を変更する場合、PLC STOP をインスタンス DB を初期化する RUN モードに移行する必要があります。 入力パラメータは USS 送信バッファで設定され、出力がある場合、出力は「以前の」有効な応答バッファから読み取られます。「USS_DRIVE」命令の実行中は、データ転送は行われません。 「USS_PORT」が実行される時、ドライブとの通信が行われます。 "「USS_DRIVE」は、送信されるメッセージの設定、および以前の要求で受信したデータの解釈のみを行います。

DIR (BOOL)入力を使用するか、または SPEED_SP (REAL)入力で符号(正または負)を使用すると、ドライブの回転方向を制御できます。 次の表に、これらの入力が共に作動し、モータが前転するよう配線されていると想定した場合のドライブ方向の決定方法を示します。

SPEED_SP	DIR	ドライブの回転方向
値 > 0	0	反転
値 > 0	1	前転
値< 0	0	前転
値< 0	1	反転

パラメータ

ボックスの下をクリックして、ボックスを拡大しすべてのパラメータを表示します。 グレイ表示されているパラメータ接続はオプションで、割り当てる必要はありません。

次の表に、「USS_DRIVE」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
RUN	Input	BOOL	I、Q、M、D、L または定数	ドライブのスタートビット: このパラメータ値が TRUE の場合、この入力によってドライブを事前設定された速度で実行できます。
OFF2	Input	BOOL	I、Q、M、D、L または定数	「電気停止」ビット。 このパラメータ値が FALSE の場合、このビットによってドライブはブレーキをかけることなく停止に向かって惰行運転します。

OFF3	Input	BOOL	I、Q、M、D、L または定数	即時停止ビット - パラメータの値が FALSE の場合、このビットによってドライブにブレーキをかけて即時停止させます。
F_ACK	Input	BOOL	I、Q、M、D、L または定数	故障確認ビット - このビットによってドライブで故障ビットがリセットされます。このビットは故障が除去された後にセットされ、前の故障を示す必要がないことをドライブに指示します。
DIR	Input	BOOL	I、Q、M、D、L または定数	ドライブの方向制御 - このビットは方向が前転であることを示すためにセットされます(SPEED_SP が正の場合)。
DRIVE	Input	USINT	I、Q、M、D、L または定数	ドライブアドレス: この入力は USS ドライブのアドレスです。 有効な範囲はドライブ 1 からドライブ 16 です。
PZD_LEN	Input	USINT	I、Q、M、D、L または定数	ワード長さ - これは PZD データのワード数です。 有効な値は、2、4、6 または 8 ワードです。 デフォルトは 2 です。
SPEED_SP	Input	REAL	I、Q、M、D、L または定数	速度セットポイント - これはドライブの速度で、設定した頻度のパーセンテージとして表されます。 正の値は前転方向を指定します(DIR の値が TRUE の場合)。
CTRL3	Input	WORD	I、Q、M、D、L または定数	コントロールワード 3 - ドライブ上でユーザーが設定可能なパラメータに書き込まれた値。 ユーザーはドライブにこの値を設定する必要があります。 オプションパラメータ
CTRL4	Input	WORD	I、Q、M、D、L または定数	コントロールワード 4 - ドライブ上でユーザーが設定可能なパラメータに書き込まれた値。 ユーザーはドライブにこの値を設定する必要があります。 オプションパラメータ
CTRL5	Input	WORD	I、Q、M、D、L または定数	コントロールワード 5 - ドライブ上でユーザーが設定可能なパラメータに書き込まれた値。 ユーザーはドライブにこの値を設定する必要があります。 オプションパラメータ
CTRL6	Input	WORD	I、Q、M、D、L または定数	コントロールワード 6 - ドライブ上でユーザーが設定可能なパラメータに書き込まれた値。 ユーザーはドライブにこの値を設定する必要があります。
CTRL7	Input	WORD	I、Q、M、D、L または定数	コントロールワード 7 - ドライブ上でユーザーが設定可能なパラメータに書き込まれた値。 ユーザーはドライブにこの値を設定する必要があります。 オプションパラメータ
CTRL8	Input	WORD	I、Q、M、D、L または定数	コントロールワード 8 - ドライブ上でユーザーが設定可能なパラメータに書き込まれた値。 ユーザーはドライブにこの値を設定する必要があります。 オプションパラメータ
NDR	Output	BOOL	I、Q、M、D、L	新規データ準備完了 - このパラメータの値が TRUE の場合、このビットは出力に新規

				通信要求からのデータが含まれることを示します。
ERROR	Output	BOOL	I、Q、M、D、L	エラー発生 - このパラメータの値が TRUE の場合、エラーが発生し、STATUS 出力は有効であることを示します。 他のすべて出力はエラーにゼロがセットされます。通信エラーは、「USS_PORT」命令の ERROR と STATUS 出力でのみ報告されます。
STATUS	Output	WORD	I、Q、M、D、L	要求のステータス値。 サイクルの結果を示します。 これはドライブから返されるステータスワードではありません。
RUN_EN	Output	BOOL	I、Q、M、D、L	実行が有効 - このビットは、ドライブが実行中かどうかを示します。
D_DIR	Output	BOOL	I、Q、M、D、L	ドライブ方向 - このビットは、ドライブが前転の実行中かどうかを示します。
INHIBIT	Output	BOOL	I、Q、M、D、L	ドライブ禁止 - このビットはドライブでの禁止ビットの状態を示します。
FAULT	Output	BOOL	I、Q、M、D、L	ドライブ故障 - このビットはドライブが故障を登録したことを示します。 故障を除去し、F_ACK ビットをセットして、このビットを消去します。
SPEED	Output	REAL	I、Q、M、D、L	現在のドライブ速度(ドライブステータスワード 2 のスケールされた値) - 設定された速度のパーセンテージとして表わされるドライブの速度値。
STATUS1	Output	WORD	I、Q、M、D、L	ドライブステータスワード 1 - この値には、ドライブの固定ステータスビットが含まれます。
STATUS3	Output	WORD	I、Q、M、D、L	ドライブステータスワード 3 - この値には、ドライブ上でユーザーが設定可能なステータスワードが含まれます。
STATUS4	Output	WORD	I、Q、M、D、L	ドライブステータスワード 4 - この値には、ドライブ上でユーザーが設定可能なステータスワードが含まれます。
STATUS5	Output	WORD	I、Q、M、D、L	ドライブステータスワード 5 - この値には、ドライブ上でユーザーが設定可能なステータスワードが含まれます。
STATUS6	Output	WORD	I、Q、M、D、L	ドライブステータスワード 6 - この値には、ドライブ上でユーザーが設定可能なステータスワードが含まれます。
STATUS7	Output	WORD	I、Q、M、D、L	ドライブステータスワード 7 - この値には、ドライブ上でユーザーが設定可能なステータスワードが含まれます。
STATUS8	Output	WORD	I、Q、M、D、L	ドライブステータスワード 8 - この値には、ドライブ上でユーザーが設定可能なステータスワードが含まれます。

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

USS_RPM: ドライブからパラメータの読み出し

説明

「USS_RPM」命令はドライブからパラメータを読み出します。 1 つの USS ネットワークと 1 つの PtP 通信モジュールに割り当てられるすべての USS ファンクションは同じインスタンスデータブロックを使用する必要があります。 "「USS_RPM」はメインプログラム OB から呼び出さなければなりません。

パラメータ

次の表に、「USS_RPM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	要求の送信: このパラメータの値が TRUE の場合、新規の読み出し要求が求められていることを示します。 このパラメータの要求がすでに保留中であれば、これは無視されます。
DRIVE	Input	USINT	I、Q、M、D、L または定数	ドライブアドレス: この入力は USS ドライブのアドレスです。 有効な範囲はドライブ 1 からドライブ 16 です。
PARAM	Input	UINT	I、Q、M、D、L または定数	パラメータ番号 この入力は、書き込まれるドライブパラメータを指定します。 このパラメータの範囲は 0～2047 です。この範囲を超えるパラメータにアクセスする方法の詳細については、ドライブのマニュアルを参照してください。
INDEX	Input	UINT	I、Q、M、D、L または定数	パラメータインデックス: この入力は、書き込まれるドライブパラメータインデックスを指定します。 これは 16 ビットの値で、最下位ビットは範囲(0～255)の実際のインデックス値です。 最上位ビットもドライブによって使用されることがあり、ドライブ固有です。 追加情報については、ドライブマニュアルを参照してください。
USS_DB	InOut	USS_BASE	D	「USS_DRIVE」命令がプログラムに挿入される時に作成され、初期化されるインスタンス DB への参照。
DONE	Output	BOOL	I、Q、M、D、L	このパラメータの値が TRUE の場合、VALUE 出力は以前に要求した読み出しパラメータ値を保持することを示します。 「USS_DRIVE」命令がドライブからの読み出し応答を認識する時に、このビットがセットされます。 次のいずれかの場合、このビットはリセットされます。

				• 別の「USS_RPM」ポーリングを経由して応答データを要求する場合 または • 「USS_DRIVE」の次の 2 つの呼び出しの 2 番目の呼び出しが実行された場合
ERROR	Output	BOOL	I、Q、M、D、L	エラー発生 - このパラメータの値が TRUE の場合、エラーが発生し、STATUS 出力は有効であることを示します。 他のすべて出力はエラーにゼロがセットされます。通信エラーは、「USS_PORT」命令の ERROR と STATUS 出力でのみ報告されます。
STATUS	Output	WORD	I、Q、M、D、L	これは、要求のステータス値です。 読み出し要求の結果を示します。 一部のステータスコードの追加情報については、「USS_Extended_Error」タグを参照してください。
VALUE	Output	VARIANT	I、Q、M、D、L	これは読み出されたパラメータの値であり、DONE ビットの値が TRUE の場合のみ有効です。

有効なデータタイプの詳細については、「**有効なデータタイプの概要**」を参照してください。

USS_WPM: ドライブのパラメータ変更

説明

「USS_WPM」命令はドライブのパラメータを変更します。 1 つの USS ネットワークと 1 つの PtP 通信モジュールに割り当てられるすべての USS ファンクションは同じインスタンスデータブロックを使用する必要があります。 "USS_WPM」はメインプログラム OB から呼び出さなければなりません。

注記

EEPROM 書き込み操作

EEPROM 書き込み操作の乱用に気をつけてください。 EEPROM 書き込み操作数を最小限に抑え、EEPROM 寿命を延長します。

パラメータ

次の表に、「USS_WPM」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	要求の送信: このパラメータの値が TRUE の場合、新規の書き込み要求が求められていることを示します。 このパラメータの要求がすでに保留中であれば、これは無視されます。
DRIVE	Input	USINT	I、Q、M、D、L または定数	ドライブアドレス: この入力は USS ドライブのアドレスです。 有効な範囲はドライブ 1 からドライブ 16 です。
PARAM	Input	UINT	I、Q、M、D、L または定数	パラメータ番号 この入力は、書き込まれるドライブパラメータを指定します。 このパラメータの範囲は 0～2047 です。この範囲を超えるパラメータにアクセスする方法の詳細については、ドライブのマニュアルを参照してください。
INDEX	Input	UINT	I、Q、M、D、L または定数	パラメータインデックス: この入力は、書き込まれるドライブパラメータインデックスを指定します。 これは 16 ビットの値で、最下位ビットは範囲(0～255)の実際のインデックス値です。 最上位ビットもドライブによって使用されることがあり、ドライブ固有です。 追加情報については、ドライブマニュアルを参照してください。
EEPROM	Input	BOOL	I、Q、M、D、L または定数	ドライブ EEPROM への保存: このパラメータの値が TRUE の場合、ドライブパラメータに書き込まれた値はドライブ EEPROM に保存されます。 このパラメータの値が FALSE の場合、書き込まれた値のみが一時的に保存され、ドライブが次にオンになった時に失われます。

VALUE	Input	VAR-IANT	I、Q、M、D、L または定数	書き込まれるパラメータの値。 REQ の移行時に有効でなければなりません。
USS_DB	InOut	USS_BASE	D	これは、「USS_DRIVE」命令がプログラムに挿入される時に作成され、初期化されるインスタンス DB への参照です。
DONE	Output	BOOL	I、Q、M、D、L	このパラメータの値が TRUE の場合、VALUE 入力がドライブに書き込まれました。 「USS_DRIVE」命令がドライブからの書き込み応答を認識する時に、このビットがセットされます。 次のいずれかの場合、このビットはリセットされます。 別の「USS_WPM」照会を使用して書き込み操作が完了しているか、または「USS_DRIVE」の次の 2 つの呼び出しの 2 番目の呼び出しが実行された時に、ドライブの確認が必要になります。
ERROR	Output	BOOL	I、Q、M、D、L	エラー発生: このパラメータの値が TRUE の場合、エラーが発生し、STATUS 出力は有効であることを示します。 他のすべて出力はエラーにゼロがセットされます。 通信エラーは、「USS_PORT」命令の ERROR と STATUS 出力でのみ報告されます。
STATUS	Output	WORD	I、Q、M、D、L	これは、要求のステータス値です。 書き込み要求の結果を示します。 一部のステータスコードの追加情報については、「USS_Extended_Error」タグを参照してください。

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

USS 命令のパラメータ STATUS

STATUS パラメータ

次の表は、USS 命令の STATUS 出力で出力される USS 操作のステータスコードを示しています。

STATUS* (W#16#....)	説明
0000	エラーは発生していません。
8180	ドライブ応答の長さは、ドライブから受信された文字に一致していません。 エラーが発生したドライブ番号が「USS_Extended_Error」タグに返されます。 この表の下の詳細なエラー説明を参照してください。
8181	パラメータ VALUE は、データタイプ WORD、REAL または DWORD ではありません。
8182	ユーザーはタイプワードのパラメータ値を提供し、応答でドライブから DWORD または REAL を受信しました。
8183	ユーザーはタイプ DWORD または REAL のパラメータ値を提供し、応答でドライブからワードを受信しました。
8184	ドライブからの応答テレグラムに不正なチェックサムがありました。 エラーが発生したドライブ番号が「USS_Extended_Error」タグに返されます。 この表の下の詳細なエラー説明を参照してください。
8185	無効なドライブアドレス(有効なドライブアドレス範囲: 1-16)
8186	速度のセットポイントが有効な範囲外(有効な速度 SP 範囲: -200% ~ 200%)
8187	送信された要求に対応していたドライブ番号が誤りでした。 エラーが発生したドライブ番号が「USS_Extended_Error」タグに返されます。 この表の下の詳細なエラー説明を参照してください。
8188	無効な PZD ワード長さを指定しました(有効な範囲= 2、4、6 または 8 ワード)
8189	無効なボーレートを指定しました。
818A	パラメータ要求チャンネルはこのドライブの別の要求によって使用されています。
818B	ドライブが要求と再試行に対応していませんでした。 エラーが発生したドライブ番号が「USS_Extended_Error」タグに返されます。 この表の下の詳細なエラー説明を参照してください。
818C	ドライブはパラメータ要求操作で拡張エラーを返しました。 この表の下の詳細なエラー説明を参照してください。
818D	ドライブは、パラメータ要求操作で無効なアクセスエラーを返しました。 パラメータアクセスが制限される理由については、ドライブマニュアルを参照してください。
818E	ドライブは初期化されませんでした: 「USS_DRIVE」命令がこのドライブに対して少なくとも 1 回呼び出されなかった場合、このエラーコードは「USS_RPM」または「USS_WPM」に出力されます。 これによって、「USS_DRIVE」の最初のサイクルの初期設定が保持され、新規のエントリとしてドライブを初期化して以降、保留中のパラメータ読み出し要求または書き込み要求が上書きされません。 エラーを除去するには、このドライブの「USS_DRIVE」命令を呼び出します。
80Ax-80Fx	特定のエラーは、USS ライブラリによって呼び出された PtP(ポイントツーポイント)通信命令から返されました。 これらのエラーコード値は USS ライブラリでは変更されず、PtP 命令の記述に定義されます。

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。

USS_Extended_Error - USS ドライブはエラーコードを拡張しました。

USS ドライブはドライブの内部パラメータへの読み出しアクセスと書き込みアクセスをサポートしています。 この機能によって、ドライブの構成やコントロールを分散できます。 ドライブパラメータアクセス操作は失敗しますが、値が範囲外またはドライブの現在のモードでの無効な要求などのエラーのためです。 ドライブは「USS_DRIVE」命令のインスタンス DB 内の「USS_Extended_Error」変数内に出力されるエラーコードを生成します。 このエラーコード値は、「USS_RPM」または「USS_WPM」命令の最終実行にのみ有効です。 ドライブエラーコードは、STATUS の値が 16 進数 818C の場合、「USS_Extended_Error」タグに置かれます。 「USS_Extended_Error」のエラーコードは、ドライブの種類によって異なります。 読み出しと書き込みパラメータ操作の拡張エラーコードの説明については、ドライブのマニュアルを参照してください。

MODBUS (RTU)

この章には下記に関する情報が記載されています：

- MB_COMM_LOAD: MODBUS RTU の PtP モジュールの設定ポート (S7-1200)
- MB_MASTER: PtP ポート経由で Modbus マスタとして通信 (S7-1200)
- MB_SLAVE: Modbus スレーブとして PtP ポート経由の通信 (S7-1200)

MB_COMM_LOAD: MODBUS RTU の PtP モジュールの設定ポート

説明

「MB_COMM_LOAD」命令は、MODBUS RTU プロトコルを使った通信のためのポートを設定します。 次のハードウェアをこの操作に使用できます。

- 最大 3 つのポイントツーポイントモジュール(PtP) CM 1241 RS485 または CM 1241 RS232
- これに追加される通信ボード CB 1241 RS485

ポートの設定後、「MB_SLAVE」または「MB_MASTER」命令を実行して Modbus 経由で通信します。

呼び出し

"MODBUS RTU プロトコル用にポートを設定するために、「MB_COMM_LOAD」を 1 回呼び出す必要があります。 設定が完了すると、「MB_MASTER」および「MB_SLAVE」命令でポートを使用できます。

"いずれかの通信パラメータを修正する必要がある場合、「MB_COMM_LOAD」を再度呼び出す必要があります。 「MB_COMM_LOAD」を呼び出すたびに、通信バッファが削除されます。 通信中のデータの損失を防ぐため、必要のない場合はこの命令は呼び出さないようにしてください。

「MB_COMM_LOAD」の 1 つのインスタンスを使って、MODBUS 通信に使用される各通信モジュールのポートを設定する必要があります。 使用する各ポートに、一意の「MB_COMM_LOAD」インスタンスデータブロックを割り当てます。 S7-1200 CPU は、3 つの通信モジュールに限られます。

「MB_MASTER」または「MB_SLAVE」命令を挿入すると、インスタンスデータブロックが割り当てられます。 MB_DB パラメータを「MB_COMM_LOAD」命令に指定すると、このインスタンスデータブロックが参照されます。

パラメータ

次の表に、「MB_COMM_LOAD」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	信号立ち上がりエッジでの命令の実行。
PORT	Input	PORT	I、Q、M、D、L、または定数	通信ポートの ID: 通信モジュールをデバイス構成に挿入した後、PORT ボックス接続のドロップダウンリストにポート ID が表示されます。この定数は、タグテーブルの「定数」タブでも参照できます。
BAUD	Input	UDINT	I、Q、M、D、L、または定数	ボーレートの選択: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200 その他すべての値は無効です。
PARITY	Input	UINT	I、Q、M、D、L、または定数	パリティの選択: • 0 – なし

				• 1 – 奇数 • 2 – 偶数
FLOW_CTRL	Input	UINT	I、Q、M、D、L、または定数	フロー制御の選択: • 0 – (デフォルト)フロー制御なし • 1 – RTS が常にオンのハードウェアフロー制御(RS485 ポートには適用されません) • 2 - RTS 切り替えのハードウェアフロー制御
RTS_ON_DLY	Input	UINT	I、Q、M、D、L、または定数	RTS オンディレーの選択: • 0 – (デフォルト) メッセージの最初の文字が転送されるまで RTS の有効な遅延なし。 • 1～65535 – メッセージの最初の文字が転送されるまでの「RTS 有効」のミリ秒単位の遅延(RS-485 ポートには適用されません)。RTS 遅延は、FLOW_CTRL の選択とは独立して適用する必要があります。
RTS_OFF_DLY	Input	UINT	I、Q、M、D、L、または定数	RTS オフ遅延の選択: • 0 – (デフォルト)最後の文字が送信されてから「RTS 無効」までの遅延なし。 • 1～65535 – 最後の文字が転送されてから「RTS 無効」までのミリ秒単位の遅延(RS-485 ポートには適用されません)。RTS 遅延は、FLOW_CTRL の選択とは独立して適用する必要があります。
RESP_TO	Input	UINT	I、Q、M、D、L、または定数	応答タイムアウト: 「MB_MASTER」がスレーブが応答するまで許容するミリ秒単位の時間。 スレーブがこの時間内に応答しない場合、「MB_MASTER」が要求を繰り返すか、指定された数の再試行が既に送信された場合はエラーで要求を終了します。 5 ミリ秒～65535 ミリ秒(デフォルト= 1000 ミリ秒)。
MB_DB	Input	MB_BASE	D	「MB_MASTER」または「MB_SLAVE」命令のインスタンスデータブロックの参照。 使用しているプログラムに「MB_SLAVE」または「MB_MASTER」を挿入した後、MB_DB ボックス接続のドロップダウンリストに DB 識別子が表示されます。
DONE	Output	BOOL	I、Q、M、D、L	命令の実行がエラーなしで完了しました。
ERROR	Output	BOOL	I、Q、M、D、L	エラー: • 0 – エラーが検出されませんでした。

				• 1 – エラーが検出されたことを示します。 STATUS パラメータでエラーコードが出力されました。
STATUS	Output	WORD	I、Q、M、D、L	ポートの構成エラーコード

有効なデータタイプの追加情報については、「有効なデータタイプの概要」を参照してください。

パラメータ STATUS

エラーコード * (W#16#...)	説明
0000	エラーは発生していません。
8180	ポート ID の無効な値(通信モジュールの誤ったアドレス)。
8181	無効なボーレート値。
8182	無効なパリティ値。
8183	無効なフロー制御値。
8184	応答のタイムアウトの無効な値(タイムアウトがレポートされるまでの時間は少なくとも 25 ミリ秒必要です)。
8185	MB_DB パラメータの「MB_MASTER」または「MB_SLAVE」命令のインスタンス DB への不正なポインタ。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

MB_COMM_LOAD データブロックタグ

以下の表に、プログラムで使用できる MB_COMM_LOAD のインスタンス DB 内の公開静的タグを示します。

インスタンス DB の静的タグ

タグ	データタイプ	デフォルト	説明
ICHAR_GAP	WORD	0	文字間の文字間隔の遅延。 このパラメータはミリ秒単位で指定され、受信文字間の予想時間を大きくするために使用されます。 このパラメータに対応するビット時間数が、Modbus 標準値の 35 ビット時間(3.5 文字時間)に追加されます。
RETRIES	WORD	2	「応答なし」のエラーコード 0x80C8 が返されるまでのマスタによる試行の繰り返し回数。
MODE	USINT	0	モード 以下のモードが許可されています。 • 0 = 全二重 (RS232) • 1 = 全二重 (RS422) 4 線式モード(ポイントツーポイント) • 2 = 全二重 (RS422) 4 線式モード(マルチポイントマスタ) • 3 = 全二重 (RS422) 4 線式モード(マルチポイントスレーブ) • 4 = 半二重 (RS485) 2 線式モード

LINE_PRE	USINT	0	回線初期状態の受信 以下の既定値が許可されています。 • 0 = 既定値「なし」 • 1 = シグナル R(A) 5 V、シグナル R(B) 0 V (ブレーク検出) この既定では、ブレーク検出は使用できません。 以下とともにのみ選択できます。[全二重 (RS422) 4 線式モード(ポイントツーポイント接続]および[全二重 (RS422) 4 線式モード(マルチポイントスレーブ)] • 2 = シグナル R(A) 0 V/シグナル R(B) 5 V この既定は休止状態に相当します(送信元が動作中でない)。 この既定では、ブレーク検出は使用できません。
CABLE-BREAK	USINT	0	ケーブル断線検出のアクティブ化: • 0 - 有効になっていません • 1 - 有効になっています

MB_MASTER: PtP ポート経由で Modbus マスタとして通信

この章には下記に関する情報が記載されています：

MB_MASTER の説明

説明

「MB_MASTER」命令によって、ポイントツーポイントモジュール(CM)または通信ボード(CB)を使って、使用しているプログラムが Modbus マスタとして通信できます。 1 つ以上の Modbus スレーブデバイスのデータにアクセスすることができます。

「MB_MASTER」命令がポートで通信できるためには、まず「MB_COMM_LOAD」を実行する必要があります。

使用しているプログラムに「MB_MASTER」命令を挿入すると、インスタンス DB が作成されます。このインスタンス DB は、「MB_COMM_LOAD」命令の MB_DB 入力パラメータで指定します。

Modbus マスタ通信のルール

- Modbus マスタ要求で使用するポートは、「MB_SLAVE」には使用できません。
- 同じインスタンス DB を使用している場合、1 つのポートを 1 つ以上の「MB_MASTER」呼び出しに使用することができます。
- Modbus 命令は、通信プロセスを制御するために、通信割り込みイベントを使用しません。 使用しているプログラムが完了した送信および受信操作の「MB_MASTER」命令をポーリングする必要があります。
- 命令の呼び出し:
  - 可能な場合は、周期プログラム OB で「MB_MASTER」命令を呼び出します。この命令は、遅延またはサイクリック割り込み OB のみで呼び出せます。
  - 異なる優先度クラスのオーガニゼーションブロックで複数の「MB_MASTER」命令を呼び出さないでください。 「MB_MASTER」命令がより高い優先度クラスから「先行して」呼び出された場合、この命令は不正に実行される場合があります。
  - 「MB_MASTER」命令は、スタートアップ、診断または時刻エラー OB で呼び出さないでください。
- 転送が開始された後、命令によって DONE または ERROR 出力パラメータが「1」にセットされるまで、EN パラメータ(LAD/FBD)が値「1」にセットされた状態であることが必要です。 命令の実行中に REQ パラメータによる新たな呼び出しを行うと、エラーの原因になります。 命令の実行後、REQ パラメータのビットがインスタンス DB の BLOCKED_PROC_TIMEOUT パラメータで指定された時間に設定された状態が継続します。
- 「MB_MASTER」がスレーブに要求を送信する場合、スレーブからの応答が届くまで「MB_MASTER」の実行が継続するようにしてください。

パラメータ

次の表に、「MB_MASTER」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L	要求入力: • 0 – 要求なし • 1 – Modbus スレーブへのデータ転送の要求

MB_ADDR	Input	UINT	I、Q、M、D、L または定数	MODBUS RTU ステーションアドレス: • デフォルトのアドレス範囲: 0 ~ 247 • 拡張されたアドレス範囲: 0 ~ 65535 値「0」は、すべての Modbus スレーブへのメッセージのブロードキャスト用に予約されています。 MODBUS ファンクションコード 05、06、15、および 16 のみが、ブロードキャストでサポートされるファンクションコードです。
MODE	Input	USINT	I、Q、M、D、L または定数	モードの選択: 要求のタイプを指定します: 読み出し、書き込み、または診断: 詳細については、MODBUS ファンクションテーブルを参照してください。
DATA_ADDR	Input	UDINT	I、Q、M、D、L または定数	スレーブの開始アドレス: Modbus スレーブでアクセスするデータの開始アドレスを指定します。 有効なアドレスについては、MODBUS ファンクションテーブルを参照してください。
DATA_LEN	Input	UINT	I、Q、M、D、L または定数	データ長: この要求でアクセスするビットまたはワードの数を指定します。 有効な長さについては、MODBUS ファンクションテーブルを参照してください。
DATA_PTR	Input	VARIANT	M、D	読み出す、または書き込むデータの DB または CPU のビットメモリアドレスをポイントします。 DB の場合、これは「標準 (S7-300/400 との互換性あり)」アクセスタイプで作成する必要があります。
DONE	Output	BOOL	I、Q、M、D、L	• 0: トランザクションが未完了 • 1: トランザクションがエラーなしで完了
BUSY	Output	BOOL	I、Q、M、D、L	• 0: 進行中の「MB_MASTER」トランザクションなし • 1: "「MB_MASTER」トランザクションが進行中
ERROR	Output	BOOL	I、Q、M、D、L	• 0: エラーは発生していません。 • 1: エラー、エラーコードは STATUS パラメータで示されます。
STATUS	Output	WORD	I、Q、M、D、L	実行条件コード

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

命令の通信および構成エラーメッセージ

エラーコード * (W#16#....)	説明
0000	エラーは発生していません。

80C8	スレーブタイムアウト スレーブのボーレート、パリティおよびコネクタをチェックしてください。
80D1	受信元が有効な転送を保留するためにフロー制御要求を発行し、転送は待機時間内に再度有効にはなりません。 受信元が待機時間内に CTS を検出しない場合、このエラーはハードウェアフロー制御中にも生成されます。
80D2	DCE から DSR シグナルが受信されなかったため、送信要求が中止されました。
80E0	受信バッファが不足しているため、メッセージが終了しました。
80E1	パリティエラーの結果、メッセージが終了しました。
80E2	フレームエラーの結果、メッセージが終了しました。
80E3	オーバーランエラーの結果、メッセージが終了しました。
80E4	指定された長さが合計バッファサイズを超えた結果、メッセージが終了しました。
8180	ポート ID の無効な値。
8186	無効な MODBUS ステーションアドレス
8188	MODE パラメータが無効な値がブロードキャスト呼び出しの不正な値を持ちます。
8189	無効なデータアドレス値。
818A	無効なデータ長の値。
818B	ローカルデータソース/宛先への無効なポインタ: サイズが不正
818C	DATA_PTR パラメータに不正なポインタがあります。 ビットメモリエリアへのポインタまたは「標準 (S7-300/400 との互換性あり)」アクセスタイプの DB を使用します。
8200	リクエストの送信の処理でポートが使用中
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

MODBUS プロトコルのエラーメッセージ

エラーコード * (W#16#....)	**スレーブの応答コード**	**説明**
8380	-	CRC エラー
8381	01	ファンクションコードがサポートされていません。
8382	03	データ長エラー
8383	02	データアドレスのエラー、またはアドレスが DATA_PTR の有効範囲外
8384	> 03	データ値エラー
8385	03	データ診断コード値がサポートされていません(ファンクションコード 08)
8386	-	応答のファンクションコードが、照会のファンクションコードと一致しません。
8387	-	不正なスレーブからの応答
8388	-	スレーブの書き込み呼び出しに対する応答が不正です。 スレーブによって送信されたデータが、マスターからの照会と一致しません。

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。

パラメータ REQ

説明

- REQ = FALSE: 要求なし
- REQ = TRUE: データの Modbus スレーブへの転送の要求

レベル制御接点またはエッジ制御接点によって、この入力を制御できます。

同一のインスタンス DB を使用する別の命令「MB_MASTER」が、現在の要求が処理されたときのみ、要求を出すことができるようにするために、この入力が有効にされるたびに、状態マシンが開始されます。 応答が受信されるか、エラーが検出されるまで、現在の要求の他のすべての入力状態が記録され、内部に保存されます。

現在の要求が完全に処理される前に、「MB_MASTER」の同一のインスタンスが REQ 入力 = 1 で再び実行されると、それ以降の転送は行われません。 ただし、要求の処理が終了した場合は、「MB_MASTER」が REQ 入力 = 1 で再び実行されるときに、新しい要求が出されます。

DATA_ADDR および MODE パラメータ

説明

DATA_ADDR パラメータを使って、Modbus スレーブのデータアクセスの開始アドレスを指定します。

MODE パラメータと MODBUS アドレスを使って、Modbus スレーブに転送されるファンクションコードを指定します。 次の表は、MODE パラメータ、ファンクションコードおよび MODBUS アドレス範囲の関係を示します。

MODE	MOD-BUS ファンクション	データ長	操作およびデータ	MODBUS アドレス
0	01	1 ~ 2000 1 ~ 1992 (1)	出力ビットの読み出し: 照会あたり 1 ~ (1992 または 2000)ビット	1 ~ 9999
0	02	1 ~ 2000 1 ~ 1992 (1)	入力ビットの読み出し: 照会あたり 1 ~ (1992 または 2000)ビット	10001 ~ 19999
0	03	1 ~ 125 1 ~ 124 (1)	保持レジスタの読み出し: 照会あたり 1 ~ (124 または 125) WORD	40001 ~ 49999 または 400001 ~ 465535
0	04	1 ~ 125 1 ~ 124 (1)	入力 WORD の読み出し: 照会あたり 1 ~ (124 または 125) WORD	30001 ~ 39999
1	05	1	出力ビットの書き込み: 照会あたり 1 ビット	1 ~ 9999
1	06	1	保持レジスタの書き込み: 1 照会あたり WORD	40001 ~ 49999 または 400001 ~ 465535
1	15	2 ~ 1968 2 ~ 1960 (1)	複数の出力ビットの書き込み 照会あたり 2 ~ (1960 または 1968)ビット	1 ~ 9999
1	16	2 ~ 123 2 ~ 122 (1)	複数の保持レジスタの書き込み: 照会あたり 2 ~ (122 または 123) WORD	40001 ~ 49999 または 400001 ~ 465535
2	15	1 ~ 1968 2 ~ 1960 (1)	1 つ以上の出力ビットの書き込み: 照会あたり 1 ~ (1960 または 1968)ビット	1 ~ 9999
2	16	1 ~ 123 2 ~ 122 (1)	1 つ以上の保持レジスタの書き込み: 照会あたり 1 ~ (122 または 123) WORD	40001 ~ 49999 または 400001 ~ 465535
11	11	0	スレーブおよびイベントカウンタの通信ステータスワードの読み出し:	-

			ステータスワードは、命令の実行を示します(0: 実行されていない、0xFFFF: 実行中)。 イベントカウンタは、メッセージの転送が成功するたびに繰り上げられます。 「MB_MASTER」命令の DATA_ADDR および DATA_LEN パラメータは、このファンクションでは無視されます。	
80	08	1	エラーコード(0x0000)の読み出しによるスレーブステータスをチェック: 1 照会あたり WORD	-
81	08	1	診断コード 0x000A によるスレーブのイベントカウンタのリセット: 1 照会あたり WORD	-
3～10、12～79、82～2555			予備	-
(1)「拡張アドレス範囲」の場合、ファンクションに使用しているデータタイプによって最大データ長は 1 バイトまたは 1 WORD 減少します。				

パラメータ DATA_PTR

説明

DATA_PTR パラメータは、データを読み書きするデータブロックまたはビットメモリへのポインタです。 データブロックを使用する場合、「標準 (S7-300/400 との互換性あり)」アクセスタイプのグローバルデータブロックを作成します。

DATA_PTR パラメータのデータブロック構造体

- これらのデータタイプは、MODBUS アドレス 30001 ~ 39999、40001 ~ 49999、および 400001 ~ 465536 の**ワードの読み出し**に、また MODBUS アドレス 40001 ~ 49999 および 400001 ~ 465536 の**ワードの書き込み**に有効です。
  - WORD、UINT、または INT データタイプの標準配列(以下参照)。
  - 各エレメントが一意の名前と 16 ビットのデータタイプを持つ、名前を付けられた WORD、UINT、または INT 構造体。
  - 各エレメントが一意の名前と 16 ビットまたは 32 ビットのデータタイプを持つ、名前を付けられた複雑な構造体。
- MODBUS アドレス 00001 ~ 09999 および 10001 ~ 19999 のビットの読み出しおよび書き込み。
  - Bool データタイプの標準配列。
  - 一意の名前が付けられた Bool 変数の名前が付けられた Bool 構造体。
- 必須ではないものの、各「MB_MASTER」命令が、グローバルデータブロック内に独自に独立したメモリ領域を持つようにすることを推奨します。 この推奨の理由は、複数の「MB_MASTER」命令がグローバルデータブロックの同じ領域で読み書きをするとデータが破損される可能性が高くなるからです。
- DATA_PTR のメモリ領域が、同じグローバルデータブロックにある必要はありません。 MODBUS 読み出し操作用に複数のエリアを持つ 1 つのデータブロック、MODBUS 書き込み操作用に 1 つのデータブロック、または各スレーブステーションに 1 つのデータブロックを作成することができます。

「MB_MASTER」命令のインスタンス DB

インスタンス DB の静的変数

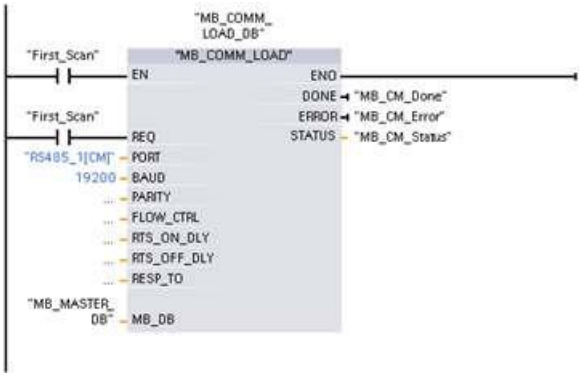
次の表は、ユーザープログラムで使用できる命令のインスタンス DB の静的変数を説明します。

変数	データタイプ	説明
MB_STATE	UINT	MODBUS 命令の内部ステータス
BLOCKED_ PROC_TIMEOUT	REAL	命令呼び出しの完了からインスタンス DB の ACTIVE ビットのリセットまでの時間。 タイムバッファを使って、ジョブが完全に送信される前に命令の実行が終了されるのを防ぎます。 デフォルトの時間は 500 ミリ秒です。
EXTENDED_ AD-DRESSING	BOOL	アドレス指定の設定: • 0: デフォルトのアドレス領域(1 バイト) • 1: 拡張アドレス領域(2 バイト) 追加情報については、セクション「EXTENDED_ADDRESSING」を参照してください。 「MB_SLAVE」命令のインスタンス DB

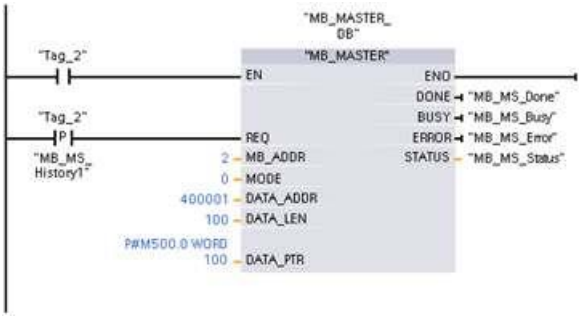
Modbus マスタのサンプルプログラム

ネットワーク(LAD)

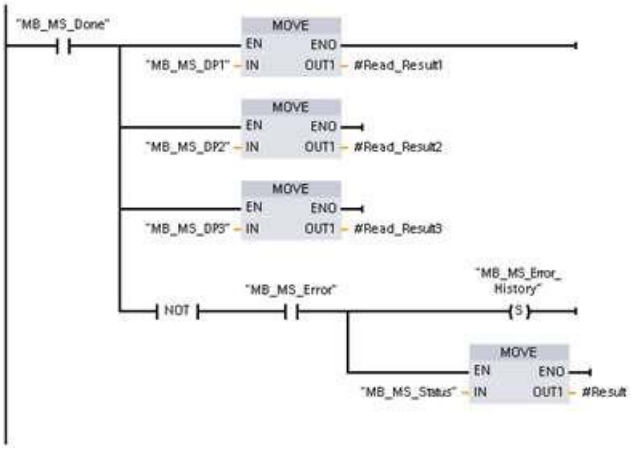
ネットワーク 1: 最初の周期中に RS-485 モジュールのパラメータを 1 回のみ初期化します。



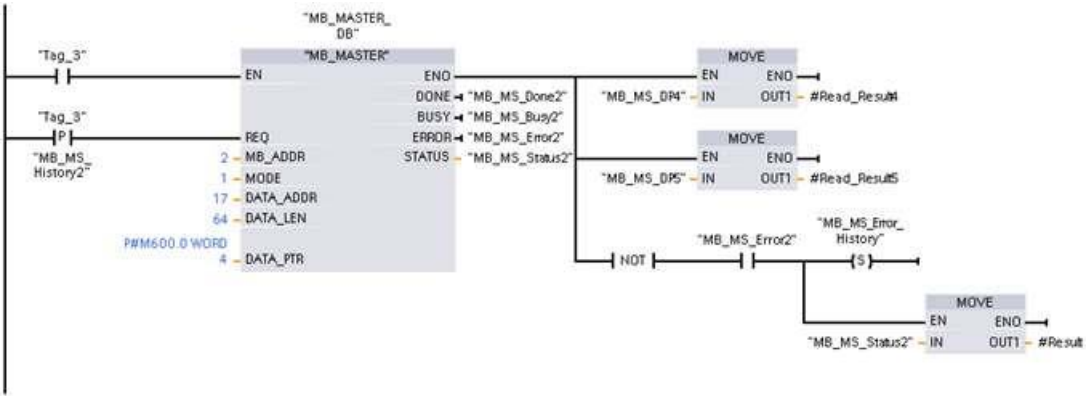
ネットワーク 2: スレーブの保持レジスタから 100 ワード読み出します。



ネットワーク 3: これは、読み出し操作が実行されてからすぐに最初の 3 ワードの値を表示するオプションのネットワークです。



ネットワーク 4: スレーブアドレス Q2.0 から開始して、64 ビットをプロセス画面出力に書き込みます。



MB_SLAVE: Modbus スレーブとして PtP ポート経由の通信

この章には下記に関する情報が記載されています：

- MB_SLAVE の説明 (S7-1200)
- 「MB_SLAVE」命令のインスタンス DB (S7-1200)
- Modbus スレーブのサンプルプログラム (S7-1200)

MB_SLAVE の説明

説明

「MB_SLAVE」命令によって、ポイントツーポイントモジュール(PtP)または通信ボード(CB)を使って、使用しているプログラムが Modbus スレーブとして通信できます。 MODBUS RTU マスタは要求を発行でき、その後、「MB_SLAVE」の実行によって使用しているプログラムが応答します。

「MB_SLAVE」命令を使用しているプログラムに挿入する場合は、一意のインスタンスデータブロックを割り当てる必要があります。 このインスタンスデータブロックは、これを「MB_COMM_LOAD」命令の MB_DB パラメータで指定する場合に使用します。

MODBUS 通信ファンクションコード(1、2、4、5、および 15)は、ターゲットシステムのプロセスイメージ入力およびプロセスイメージ出力でビットとワードの読み書きができます。 次の表は、MODBUS アドレスの CPU のプロセスイメージへのマッピングを示します。

「MB_SLAVE」の MODBUS ファンクション						S7-1200	
コード	ファンクション	データ領域	アドレス範囲			データ領域	CPU アドレス
01	ビットの読み出し	出力	1	~	8192	プロセスイメージ出力	Q0.0 ~ Q1023.7
02	ビットの読み出し	入力	10001	~	18192	プロセスイメージ入力	I0.0 ~ I1023.7
04	ワードの読み出し	入力	30001	~	30512	プロセスイメージ入力	IW0 ~ IW1022
05	ビットの書き込み	出力	1	~	8192	プロセスイメージ出力	Q0.0 ~ Q1023.7
15	ビットの書き込み	出力	1	~	8192	プロセスイメージ出力	Q0.0 ~ Q1023.7

MODBUS 通信ファンクションコード(ファンクションコード 3、6、16)は個別の保持レジスタを使用します。 これを行うには、「標準 (S7-300/400 との互換性あり)」アクセスタイプのビットメモリまたはデータブロックを使用することができます。

MB_SLAVE 命令の MB_HOLD_REG パラメータを使用して保持レジスタのタイプを指定します。 次の表は、MODBUS 保持レジスタからターゲットシステムの MB_HOLD_REG の DB アドレスへのマッピングを示します。

「MB_SLAVE」の MODBUS ファンクション			S7-1200		
コード	ファンクション	データ領域	アドレス範囲 (WORD 番号)	DB のアドレス (BYTE 番号)	ビットメモリアドレス (BYTE 番号)
03	ワードの読み出し	保持レジスタ	40001 ~ 49999 または	DW0 ~ DW19998 または	MW0 ~ CPU 限界値
			400001 ~ 465535	DW0 ~ DW131068	
06	書き込みワード	保持レジスタ	40001 ~ 49999 または	DW0 ~ DW19998 または	

			400001 ~ 465535	DW0 ~ DW131068
16	ワードの書き込み	保持レジスタ	40001 ~ 49999 または	DW0 ~ DW19998 または
			400001 ~ 465535	DW0 ~ DW131068

次の表に、サポートされている MODBUS 診断ファンクションを示します。

S7-1200「MB_SLAVE」MODBUS 診断ファンクション		
コード	**サブファンクション**	**説明**
08	0000H	照会データエコーの返しテスト:「MB_SLAVE」命令が、受信したデータワードのエコーを Modbus マスタに返します。
08	000AH	通信イベントカウンタのクリア:「MB_SLAVE」命令が、MODBUS ファンクション 11 で使用される通信イベントカウンタをクリアします。
11	-	通信イベントカウンタの取得:「MB_SLAVE」命令は、内部通信イベントカウンタを使って、Modbus スレーブに送信された MODBUS の読み出しおよび書き込み要求の成功した数を記録します。 いずれのファンクション 8、ファンクション 11、またはブロードキャスト要求でも、カウンタは繰り上げられません。 また、結果として通信エラー(たとえば、パリティまたは CRC エラー)になるあらゆる要求でも繰り上げられません。

「MB_SLAVE」命令は、要求に有効なアドレスへのアクセスが含まれる限り、Modbus マスタからのブロードキャスト書き込み要求をサポートします。

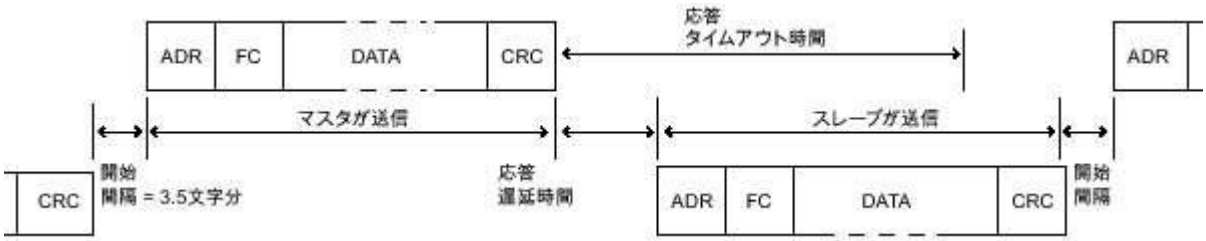
要求の有効性に関係なく、「MB_SLAVE」はブロードキャスト要求の結果として Modbus マスタに応答しません。

Modbus スレーブ通信のルール

- "「MB_SLAVE」命令がポートと通信できるようにするには、「MB_COMM_LOAD」を実行してポートを設定する必要があります。
- ポートがスレーブとして Modbus マスタに応答する場合、「MB_MASTER」はそのポートを使用できません。 特定のポートで使用できるのは、「MB_SLAVE」のインスタンスのみです。
- Modbus 命令は、通信プロセスを制御するために、通信割り込みイベントを使用しません。 使用しているプログラムが、完了した送信および受信操作に対して「MB_SLAVE」命令をポーリングして通信プロセスを制御する必要があります。
- 「MB_SLAVE」命令は、Modbus マスタから着信する要求にタイミング良く応答することができるレートで定期的に実行する必要があります。 したがって、この命令を周期プログラム OB で呼び出すことを推奨します。割り込み OB での「MB_SLAVE」命令の呼び出しは可能ですが、実行が大幅に遅延する可能性があるため、推奨されません。

「MB_SLAVE」の実行頻度

Modbus マスタからの要求を受信し、必要に応じて応答するため、「MB_SLAVE」命令を定期的に実行する必要があります。 「MB_SLAVE」の実行頻度は、Modbus マスタの指定された応答タイムアウト時間によって異なります。 これは、次の図で説明されています。



応答タイムアウト時間は、Modbus マスタが Modbus スレーブからの応答の開始まで待機する時間です。 この時間は、MODBUS プロトコルで定義するのではなく、各 Modbus マスタのパラメータで定義します。 「MB_SLAVE」の実行の頻度(ある実行と次の実行の間の時間)は、使用している Modbus マスタの専用のパラメータに基づく必要があります。 最低でも、Modbus マスタの応答タイムアウト時間内に「MB_SLAVE」を 2 回実行してください。

パラメータ

次の表に、「MB_SLAVE」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
MB_ADDR	Input	USINT	I、Q、M、D、L、または定数	Modbus スレーブのステーションアドレス(アドレス範囲:0～255)
MB_HOLD_REG	Input	VAR-IANT	D	MODBUS 保持レジスタ DB へのポインタ。 DB は、「標準 (S7-300/400 との互換性あり)」アクセスタイプで作成する必要があります。
NDR	Output	BOOL	I、Q、M、D、L	新規データの準備完了: • 0: 新規データなし • 1: Modbus マスタによって新規のデータが書き込まれたことを示します。
DR	Output	BOOL	I、Q、M、D、L	データの読み出し: • 0: データ読み出しなし • 1: Modbus マスタによってデータが読み出されたことを示します。
ERROR	Output	BOOL	I、Q、M、D、L	• 0: エラーが検出されませんでした。 • 1: エラー、対応するエラーコードが STATUS に出力されます。
STATUS	Output	WORD	I、Q、M、D、L	エラーコード

有効なデータタイプの詳細情報については、「有効なデータタイプの概要」を参照してください。

STATUS パラメータ

STATUS*(W#16#....)	説明
80C8	指定された応答タイムアウト(RCVTIME または MSGTIME を参照)が「0」になっています。

80D1	受信元が有効な転送を保留するためにフロー制御要求を発行し、転送は待機時間内に再度有効にはなりません。 受信元が待機時間内に CTS を検出しない場合、このエラーはハードウェアフロー制御中にも生成されます。	
80D2	DCE から DSR シグナルが受信されなかったため、送信要求が中止されました。	
80E0	受信バッファが不足しているため、メッセージが終了しました。	
80E1	パリティエラーの結果、メッセージが終了しました。	
80E2	メッセージフレームエラーの結果、メッセージが終了しました。	
80E3	オーバーランエラーの結果、メッセージが終了しました。	
80E4	指定された長さが合計バッファサイズを超えた結果、メッセージが終了しました。	
8180	ポート ID の無効な値。	
8186	無効な MODBUS ステーションアドレス	
8187	MB_HOLD_REG-DB への無効なポインタ	
818C	タイプセーフな DB タイプ MB_HOLD_REG へのポインタ(クラシック DB タイプであることが必要)	
応答コードが Modbus マスタに送信された(B#16#...)		
8380	応答なし	CRC エラー
8381	01	ファンクションコードがサポートされていない、またはブロードキャストでサポートされていない
8382	03	データ長エラー
8383	02	データアドレスのエラー、またはアドレスが MB_HOLD_REG の有効範囲外
8384	03	データ値エラー
8385	03	データ診断コード値がサポートされていません(ファンクションコード 08)
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

「MB_SLAVE」命令のインスタンス DB

インスタンス DB の静的変数

次の表は、ユーザープログラムで使用できる命令のインスタンス DB の静的変数を説明します。 使用しているプログラムは、HR_Start_Offset および Extended_Addressing 変数に値を書き込み、Modbus スレーブの操作を制御できます。

その他の変数を読み出して、MODBUS ステータスをモニタすることができます。

変数	データタイプ	説明
HR_Start_Offset	WORD	MODBUS 保持レジスタの開始アドレス(デフォルト=「0」)
Extended_ Addressing	BOOL	アドレス指定の設定: • 0: デフォルトのアドレス領域(1 バイト) • 1: 拡張アドレス領域(2 バイト)
Request_Count	WORD	スレーブが受信した照会の合計数
Slave_Message_Count	WORD	この特定のスレーブに送信された照会の数
Bad_CRC_Count	WORD	受信した CRC エラー付きのエラーの数
Broadcast_Count	WORD	受信されたブロードキャスト照会の数
Exception_Count	WORD	例外を返す必要のある Modbus 固有のエラーの数
Success_Count	WORD	プロトコルエラーなしでこの特定のスレーブについて受信した要求の数

HR_Start_Offset

40001 または 400001 から開始される MODBUS 保持レジスタのアドレス。これらのアドレスは、ターゲットシステムメモリの保持レジスタの開始アドレスに対応します。 HR_Start_Offset 変数を使って、異なる開始アドレスに対するオフセットを設定できます。

例: 保持レジスタは、MW 100 から開始され、長さは 100 WORD です。 HR_Start_Offset パラメータのオフセットが 20 の場合、保持レジスタはアドレス 40001 ではなく 40021 から開始されます。40021 および未満および 400119 を超えるそれぞれのアドレスは、アドレス指定エラーの原因になります。

	HR_Start_Offset = 0			HR_Start_Offset = 20	
	MODBUS ワードアドレス	S7-1200 バイトアドレス		MODBUS ワードアドレス	S7-1200 バイトアドレス
最小値	40001	MW100		40021	MW100
最大	40099	MW198		40119	MW198

Extended_Addressing

Modbus スレーブのアドレス指定をするため、シングルバイト(デフォルトアドレス範囲)またはダブルバイト(拡張アドレス範囲)を設定できます。 拡張アドレス指定を使って、1 つのネットワークで 247 個以上のデバイスをアドレス指定できます。 拡張アドレス指定を使用する場合、最大 64,000 個のアドレスをアドレス指定できます。 次に、MODBUS ファンクション 1 のフレームを例として示します。

1 バイトのスレーブアドレス(バイト 0)

ファンクション 1	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	
要求	[スレーブアドレス]	F コード	開始アドレス		コイルの長さ		
有効な応答	[スレーブアドレス]	F コード	長さ	コイルのデータ			
エラー応答	[スレーブアドレス]	0x81	E コード				

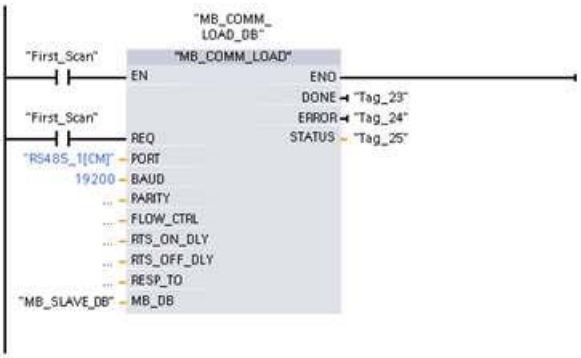
2 バイトのスレーブアドレス(バイト 0 およびバイト 1)

ファンクション 1	バイト 0	バイト 1	バイト 2	バイト 3	バイト 4	バイト 5	バイト 6
要求	[スレーブアドレス]		F コード	開始アドレス		コイルの長さ	
有効な応答	[スレーブアドレス]		F コード	長さ	コイルのデータ		
エラー応答	[スレーブアドレス]		0x81	F コード			

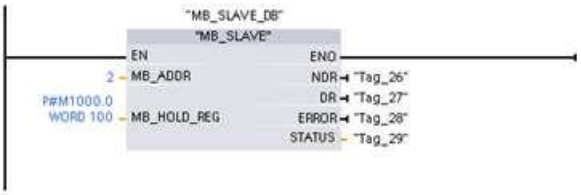
Modbus スレーブのサンプルプログラム

ネットワーク(LAD)

ネットワーク 1: 最初の周期中に RS-485 モジュールのパラメータを 1 回のみ初期化します。



ネットワーク 2: 各サイクルで Modbus マスタの要求をチェックします。 MW1000 で始まる 100 ワードは、MODBUS 保持レジスタ用に設定されます。



MODBUS (TCP)

この章には下記に関する情報が記載されています：

- MODBUS (TCP) (S7-1200)
- MODBUS (TCP) (S7-1200, S7-1500)

MODBUS (TCP)

この章には下記に関する情報が記載されています：

- MB_CLIENT: PROFINET を介した Modbus TCP クライアントとしての通信 (S7-1200)
- MB_SERVER: PROFINET を介した Modbus TCP サーバーとしての通信 (S7-1200)
- 例 (S7-1200)

MB_CLIENT: PROFINET を介した Modbus TCP クライアントとしての通信

この章には下記に関する情報が記載されています：

- MB_CLIENT の説明 (S7-1200)
- REQ および DISCONNECT パラメータ (S7-1200)
- MB_MODE および MB_DATA_ADDR パラメータ (S7-1200)
- MB_DATA_PTR パラメータ (S7-1200)
- パラメータステータス (S7-1200)

MB_CLIENT の説明

説明

「MB_CLIENT」命令は、S7-1200CPU の PROFINET 接続を経由した Modbus TCP クライアントとして通信します。 この命令を使用するためにハードウェアモジュールを追加する必要はありません。「MB_CLIENT」命令を使用し、クライアントとサーバー間の接続を確立し、リクエストの送信、応答の受信、および Modbus TCP サーバーの制御接続終了を行います。

パラメータ

以下の表に、「MB_CLIENT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	説明
REQ	Input	BOOL	Modbus TCP サーバーとの通信要求 REQ パラメータがレベル制御されます。 すなわち、この入力がセットされている限り(REQ=true)、この命令は通信要求を送信します。 • その他のクライアントのインスタンス DB は、通信要求でブロックされます。 • サーバーが応答するか、またはエラーメッセージが出力されるまで、入力パラメータへの変更は有効になりません。 • Modbus 要求の進行中にパラメータ REQ が再度設定された場合、その後は追加の送信は実行されません。
DISCONNECT	Input	BOOL	このパラメータを使用し、Modbus サーバーへの接続の確立および終了を制御します。 • 0: 指定された IP アドレスおよびポート番号への通信接続を確立します。 • 1: 通信接続を切断します。 接続終了中は、他のファンクションは実行されません。 正常な接続の終了の後に、値 7003 が STATUS パラメータに出力されます。 接続の確立中に REQ パラメータがセットされた場合、要求が直ちに送信されます。
CONNECT_ID	Input	UINT	接続を識別する一意の ID。 「MB_CLIENT」および「MB_SERVER」命令の各インスタンスは、一意の接続 ID に割り当てられる必要があります。
IP_OCTET_1	Input	USINT	1. Modbus TCP サーバーの IP アドレス*のオクテット。
IP_OCTET_2	Input	USINT	2. Modbus TCP サーバーの IP アドレス*のオクテット。
IP_OCTET_3	Input	USINT	3. Modbus TCP サーバーの IP アドレス*のオクテット。

IP_OCTET_4	Input	USINT	4. Modbus TCP サーバーの IP アドレス*のオクテット。
IP_PORT	Input	UINT	TCP/IP プロトコルを使用してクライアントが接続および通信を確立するサーバーの IP ポート番号(既定値: 502).
MB_MODE	Input	USINT	要求のモード(読み取り、書き込み、または診断)を選択します。
MB_DATA_ADDR	Input	UDINT	「MB_CLIENT」命令によってアクセスされるデータの開始アドレス。
DATA_LEN	Input	UINT	データ長: データアクセス用ビットまたはワードの数(「MB_MODE および MB_DATA_ADDR パラメータ」 - データ長を参照してください)。
MB_DATA_PTR	InOut	VARIANT	Modbus データレジスタへのポインタ。 このレジスタは、Modbus サーバーから受信する、または Modbus サーバーに送信するデータ用のバッファです。 ポインタは、グローバルデータブロックを標準アクセスで参照する必要があります。 アドレス指定されたビットの数は、8 で割り切れる必要があります。
DONE	Out	BOOL	最後のジョブが正常に完了すると、すぐに出力パラメータ DONE のビットが「1」にセットされます。
BUSY	Out	BOOL	• 0: 進行中の「MB_CLIENT 」ジョブはありません。 • 1: "「MB_ CLIENT 」ジョブが進行中です。
ERROR	Out	BOOL	• 0: エラーは発生していません。 • 1: エラーが発生しました。 エラーの原因は、STATUS パラメータによって識別されます。
STATUS	Out	WORD	命令のエラーコード。
* Mdbus TCP サーバーの 32 ビット IPv4 IP アドレスの 8 ビット長コンポーネント。			

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記

「MB_CLIENT」呼び出し中の一貫した入力データ。

Modbus クライアントが Modbus 命令を呼び出す際に、入力パラメータのステータスが内部的に保存され、次回の呼び出し時に比較されます。 この比較は、その特定の呼び出しが現在の要求を初期化したかどうかを判定するために使用されます。 共通のインスタンス DB を使用している場合、複数の「MB_CLIENT」呼び出しを実行できます。 「MB_CLIENT」インスタンスの実行中は、入力パラメータの値を変更してはなりません。 実行中に入力パラメータが変更されると、インスタンスを現在実行中か否かのチェックに「MB_CLIENT」を使用できなくなります。

複数クライアントの接続

Modbus TCP クライアントは複数の TCP 接続をサポートしています。接続の最大数は、使用している CPU によって異なります。 Modbus TCP クライアントおよびサーバーの接続を含む 1 つの CPU の接続合計数は、サポートされている接続の最大数を超えてはなりません。 Modbus TCP 接続は、クライアント、またはサーバー、またはその両方の接続で共有することが可能です。

個別のクライアント接続では、以下のルールに注意してください。

- 各「MB_CLIENT」接続は、固有のインスタンス DB を使用する必要があります。

- 各「MB_CLIENT」接続では、固有のサーバー IP アドレスが指定されている必要があります。
- 各「MB_CLIENT」接続は、固有の接続 ID を必要とします。

  命令の各インスタンス DB に対しては、関連する個別の接続 ID を使用する必要があります。 接続 ID およびインスタンス DB は対となり、各接続に対して固有である必要があります。
- 固有の IP ポート番号が必要かどうかは、サーバー設定によって異なります。

命令の静的タグ

以下の表では、「MB_CLIENT」命令のインスタンスデータブロックの編集可能な静的タグについて説明します。

タグ	データタイプ	開始値	説明
Blocked_Proc_Timeout	REAL	3.0	ブロックされた Modbus インスタンスが存在する場合、静的タグ ACTIVE がリセットされるまでの待機秒数。 これは、たとえばクライアントが要求を出力し、その要求が完全に実行される前にクライアントファンクションの実行が中止された場合などに発生します。 最大待機時間は 55 秒です。
MB_Transaction_ID	WORD	1	Modbus TCP プロトコルのトランザクション ID。 開始値「1」は、Modbus TCP サーバーが異なる値を要求する場合にのみ変更されます。
MB_Unit_ID	BYTE	255	Modbus デバイスの検出: Modbus TCP サーバーは、IP アドレスを使用してアドレス指定します。 このため、MB_UNIT_ID パラメータは Modbus TCP アドレス指定の場合には使用されません。 MB_UNIT_ID パラメータは、Modbus RTU プロトコルのスレーブアドレスのフィールドに対応します。 Modbus TCP サーバーを Modbus RTU プロトコルのゲートウェイとして使用する場合、MB_UNIT_ID を使用してシリアルネットワークのスレーブデバイスを識別できます。 この場合、MB_UNIT_ID パラメータは正しい Modbus RTU スレーブアドレスに要求を転送します。 Modbus TCP デバイスによっては、制限された値の範囲内で初期化を行うために MB_UNIT_ID パラメータを要求する場合があります。
RCV_TIMEOUT	REAL	2.0	「MB_CLIENT」命令がサーバーからの応答を待機する時間間隔の秒数。
Connected	BOOL	0	割り当てられたクライアントへの接続が確立されているかどうかを示します: 1 = 接続済み、0 = 未接続。

REQ および DISCONNECT パラメータ

説明

実行中の「MB_CLIENT」命令のインスタンスがなく、DISCONNECT パラメータの値が「0」であると、REQ = 1 の場合、新しいジョブが実行されます。まだ接続が存在しない場合は、実行中に接続が確立されます。

アクティブなジョブの実行前に「MB_CLIENT」命令の同一のインスタンスが再度実行されると(DISCONNECT = 0 および REQ = 1)、アクティブなジョブの完了時にはこのインスタンスは実行されません。 新しいジョブは、アクティブなジョブの完了時(REQ = 1)にのみ開始することが可能です。

実行のステータスは、DONE パラメータを使用してモニタすることができます。 「MB_CLIENT」命令が順番に実行される場合、このパラメータを使用して実行のステータスをモニタすることが可能です。

MB_MODE および MB_DATA_ADDR パラメータ

説明

「MB_CLIENT」命令は、ファンクションコードの代わりに MB_MODE パラメータを使用します。MB_DATA_ADDR パラメータを使用し、アクセスするデータの Modbus 開始アドレスを指定します。パラメータ MB_MODE および MB_DATA_ADDR を組み合わせ、現在の Modbus メッセージで使用するファンクションコードを定義します。

以下の表に、MB_MODE パラメータ、Modbus ファンクション、およびアドレス空間の関係を示します。

MB_MODE	Modbus ファンクション	データ長	ファンクションおよびデータタイプ	MB_DATA_ADDR
0	01	1 ~ 2000	出力ビットの読み取り: 1 回の呼び出しにつき 1 ~ 2000 ビット	1 ~ 9999
0	02	1 ~ 2000	入力ビットの読み取り: 1 回の呼び出しにつき 1 ~ 2000 ビット	10001 ~ 19999
0	03	1 ~ 125	保持レジスタの読み取り: 1 回の呼び出しにつき 1 ~ 125 WORD	40001 ~ 49999
0	04	1 ~ 125	入力ワードの読み取り: 1 回の呼び出しにつき 1 ~ 125 WORD	30001 ~ 39999
1	05	1	1 つの出力ビットの書き込み: 1 回の呼び出しにつき 1 ビット	1 ~ 9999
1	06	1	1 つの保持レジスタの書き込み: 1 1 回の呼び出しにつき WORD	40001 ~ 49999
1	15	2 ~ 1968	複数の出力ビットの読み取り: 1 回の呼び出しにつき 2 ~ 1968 ビット	1 ~ 9999
1	16	2 ~ 123	複数の保持レジスタの書き込み: 1 回の呼び出しにつき 2 ~ 123 WORD	40001 ~ 49999
2	15	1 ~ 1968	1 つ以上の出力ビットの書き込み: 1 回の呼び出しにつき 1 ~ 1968 ビット	1 ~ 9999
2	16	1 ~ 123	1 つ以上の保持レジスタの書き込み: 1 回の呼び出しにつき 1 ~ 123 WORD	40001 ~ 49999
11	11	0	サーバー通信のステータスワードおよびイベントカウンタの読み取り: • ステータスワードは処理ステータスを表します(0 - 処理中ではない、0xFFFF - 処理中)。	-

			• イベントカウンタは、メッセージが正常に送信されるたびにインクリメントされます。 このファンクションの実行中は、「MB_CLIENT」命令の MB_DATA_ADDR および MB_DATA_LEN パラメータは評価されません。	
80	08	1	エラーコード 0x0000 でサーバーステータスをチェックしてください(戻り値ループテスト - サーバーが要求を返信します): 1 1 回の呼び出しにつき WORD	-
81	08	1	エラーコード 0x000A でサーバーのイベントカウンタをリセット: 1 1 回の呼び出しにつき WORD	
3 ~ 10、12 ~ 79、82 ~ 255			予備	

MB_DATA_PTR パラメータ

説明

MB_DATA_PTR パラメータは、Modbus サーバーから読み取られた、または Modbus サーバーに書き込まれたデータの格納用データバッファに対するポインタです。 グローバルデータブロックまたはメモリ領域(M)をデータバッファとして使用することが可能です。

メモリ領域(M)内のバッファの場合、次の要領で ANY 形式のポインタを使用します。「P#ビットアドレス」「データタイプ」「長さ」(例: P#M1000.0 WORD 500).

以下の場合、MB_DATA_PTR パラメータは通信バッファを使用します。

- 「MB_CLIENT」命令の通信ファンクションの場合:
  - Modbus サーバーアドレス 00001 ~ 09999 および 10001 ~ 19999 のデータの 1 ビットの読み取りおよび書き込み。
  - Modbus サーバーアドレス 30001 ~ 39999 および 40001 ~ 49999 の 16 ビット WORD データの読み取り。
  - Modbus サーバーアドレス 40001 ~ 49999 の 16 ビット WORD データの書き込み。
- MB_DATA_PTR パラメータで割り当てたグローバル DB またはメモリ領域(M)とのデータ転送中(長さ:ビットまたは WORD)。

MB_DATA_PTR パラメータ内でバッファ用のデータブロックを使用する場合、DB エレメントにデータタイプを割り当てる必要があります。

- Modbus ビットアドレスには、1 ビットデータタイプ BOOL を使用します。
- ModbusWORD アドレスには、WORD、UINT、INT、または REAL などの 16 ビットデータタイプを使用します。
- 2 つの ModbusWORD アドレスには、DWORD、DINT、または REAL などの 32 ビットデータタイプ(ダブルワード)を使用します。
- MB_DATA_PTR では、以下のような複合 DB エレメントにもアクセス可能です。
  - 標準配列
  - エレメント名が一意の構造体
  - エレメント名が一意、かつデータタイプ長が 16 または 32 ビットの複合構造体。
- MB_DATA_PTR パラメータのデータ領域は、別のグローバルデータブロック(または別のメモリ領域)にも確保することができます。 例えば、読み取りジョブと書き込みジョブで異なるデータブロックを使用することや、各「MB_CLIENT」ステーションに対して個別のデータブロックを使用することが可能です。

パラメータステータス

パラメータ STATUS (全般的なステータス情報)

STA-TUS* (W#16#)	説明
0000	命令がエラーなしで実行されました。
0001	接続が確立されました。
0003	接続が終了しました。
7000	動作中の呼び出しはありません(REQ=0)。
7001	REQ=1 による最初の呼び出し: 処理が開始されます。BUSY の値は 1 です。
7002	中間呼び出し(REQ は無関係)。 処理が既に有効です。BUSY の値は 1 です。
7003	接続を終了中です。
7004	接続が確立され、モニタされています。ジョブ処理が行われていません。
7005	データが送信されました。
7006	データが受信されました。
80BB	ACTIVE_EST パラメータの無効な値(接続確立のタイプの識別子、T_CON_PARAM を参照): • パッシブ接続の確立のみがサーバーに対して許可されています(ACTIVE_EST = FALSE)。 • アクティブ接続の確立のみがクライアントに対して許可されています(ACTIVE_EST = TRUE)。
8380	受信された Modbus フレームの形式が不正であるか、または受信されたバイト数が少なすぎます。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

パラメータ STATUS(プロトコルエラー)

STA-TUS* (W#16#)	Modbus クライアントへの応答コード (B#16#)	説明
8381	01	ファンクションコードはサポートされていません。
8382	03	データ長のエラー。
8383	02	データアドレスのエラー、または MB_DATA_PTR のアドレス領域外へのアクセス。
8384	03	データ値のエラー。
8385	03	サポートされていない診断のエラーコード(ファンクションコード 08)。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

パラメータ STATUS (パラメータエラー)

以下の表に記載されたエラーの他に、「MB_CLIENT」では通信命令(「TCON」、「TDISCON」、「TSEND」、および「TRCV」)によって引き起こされるエラーが発生する可能性があります。

STATUS* (W#16#)	説明
80C8	定義された期間内にサーバーの応答がありませんでした。 Modbus サーバーへの接続をチェックしてください。 このエラーは、設定された反復試行が終了したときにのみレポートされます。 「MB_CLIENT」命令が、転送時のトランザクション ID(MB_TRANSACTION_ID タグ)の応答を定義された期間内に受信しない場合、このエラーコードが出力されます。
8188	MB_MODE パラメータの値が無効です。
8189	MB_DATA_ADDR パラメータのデータのアドレス指定が無効です。
818A	MB_DATA_LEN パラメータのデータ長が無効です。
818B	MB_DATA_PTR パラメータのポインタが無効です。 MB_DATA_ADDR および MB_DATA_LEN パラメータの値もチェックする必要があります。
818C	• MB_DATA_PTR ポインタは、最適化されたデータブロックを参照します。 データブロックを標準アクセスで使用するか、メモリ領域を使用してください。 • パラメータ BLOCKED_PROC_TIMEOUT でのタイムアウト(命令の静的タグを参照)。限界値の 55 秒を超えました。
818D	トランザクション ID (MB_TRANSACTION_ID タグ)が、はじめに送信された ID と一致しません(命令の静的タグを参照)。
8200	• 現在、このポート経由で別の Modbus 応答が処理されています。 • 同じ接続パラメータの MB_CLIENT の別のインスタンスが、既存の Modbus 要求を処理しています。
8380	受信された Modbus データのブロックの受信が不完全であるか、受信されたバイト数が少なすぎます。
8386	受信したファンクションコードが、はじめに送信されたファンクションコードと一致しません。
8387	• 割り当てられた接続 ID が、前回の要求で使用されたものと異なります。 「MB_CLIENT」命令の各インスタンス DB に対して使用できる接続 ID は 1 つのみです。 • サーバーによって受信される Modbus TCP プロトコルの ID が「0」でない場合も、このエラーコードが出力されます。
8388	Modbus サーバーが、要求されたデータ長とは異なるデータ長を送信しました。 このエラーは、Modbus ファンクション 15 または 16 の使用時にのみ発生します。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

注記

内部で使用される通信命令のエラーコード。

「MB_CLIENT」命令を使用すると、表にリストされたエラーに加えて、命令によって使用される通信命令(「TCON」、「TDISCON」、「TSEND」、および「TRCV」)が原因のエラーが発生する可能性があります。

エラーコードが、「MB_CLIENT」命令のインスタンスデータブロックを介して割り当てられます。それぞれの命令に対して、エラーコードが Static セクションの STATUS に表示されます。

エラーコードの意味は、対応する通信命令の文書で入手できます。

MB_SERVER: PROFINET を介した Modbus TCP サーバーとしての通信

この章には下記に関する情報が記載されています：

- MB_SERVER の説明 (S7-1200)
- MB_HOLD_REG パラメータ (S7-1200)
- パラメータステータス (S7-1200)

MB_SERVER の説明

説明

「MB_SERVER」命令は、S7-1200CPU の PROFINET 接続を経由した Modbus TCP サーバーとして通信します。 この命令を使用するためにハードウェアモジュールを追加する必要はありません。「MB_SERVER」命令は、Modbus TCP クライアントの接続要求の処理、Modbus ファンクションからの要求の受信、および応答の送信を行います。

通知
セキュリティ情報 ネットワークの各クライアントは、プロセスイメージ入出力や、Modbus 保持レジスタによって定義されるデータブロックまたはビットメモリ領域に対する読み出し/書き込みのアクセスが与えられていることに注意してください。 このオプションは、IP アドレスへのアクセスを制限して、未許可の読み出しおよび書き込み操作を防止するために使用できます。 ただし、未許可のアクセスの場合でも、共有アドレスは使用できることに注意してください。

パラメータ

以下の表に、「MB_SERVER」命令のパラメータを示します。

パラメータ	宣言	データタイプ	説明
DISCONNECT	Input	BOOL	命令「MB_SERVER」が、パートナーモジュールとのパッシブ接続を作成します。つまり、要求中の各 IP アドレスからの TCP 接続要求にサーバーが応答します。 このパラメータを使用して、接続要求の承認のタイミングを制御することが可能です。 • 0: パッシブ接続は、通信接続がない場合に確立されます。 • 1: 接続終了の初期化。 入力が設定された場合、その他の操作は実行されません。 正常な接続の終了の後に、値 7003 が STATUS パラメータに出力されます。
CONNECT_ID	Input	UINT	このパラメータは、CPU 内の接続を一意に識別します。「MB_CLIENT」および「MB_SERVER」命令の各インスタンスには、それぞれ一意となる CONNECT_ID パラメータがあります。
IP_PORT	Input	UINT	開始値は 502 です。IP ポートの番号は、Modbus クライアントの接続要求をどの IP ポートでモニタするかを定義します。 次の TCP ポート番号は、「MB_SERVER」命令のパッシブ接続で使用してはなりません。 20、21、25、80、102、123、5001、34962、34963、および 34964。
MB_HOLD_REG	InOut	VAR-IANT	「MB_SERVER」命令の Modbus 保持レジスタへのポインタ。 グローバルデータブロックを標準アクセスで保持レジスタとして使用します。 保持レジスタは、Modbus クライアントが Modbus ファンクション 3 (読み取り)、6 (書き

			込み)、および 16 (読み取り)を使用してアクセスする可能性のある値を含んでいます。
NDR	Output	BOOL	"New Data Ready": • 0: 新規データなし • 1: 新規データが Modbus クライアントによって書き込まれました。
DR	Output	BOOL	"Data Read": • 0: データ読み出しなし • 1: データが Modbus クライアントによって読み取られました。
ERROR	Output	BOOL	「MB_SERVER」命令の呼び出し中にエラーが発生した場合、ERROR パラメータの出力は TRUE にセットされます。 問題の原因に関する詳細情報は、STATUS パラメータによって示されます。
STATUS	Output	WORD	命令のエラーコード。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

Modbus アドレスのプロセスイメージへのマッピング

「MB_SERVER」命令によって、受信 Modbus ファンクション(1、2、4、5、および 15)が S7-1200 CPU (データタイプ BOOL および WORD の使用)のプロセスイメージ入力および出力に対し、直接読み取りおよび書き込みアクセスを実行できるようになります。

ファンクションコード 3、6、および 16 のデータ転送の場合、保持レジスタ(MB_HOLD_REG パラメータ)には 1 バイト以上が定義される必要があります。 次の表に、Modbus アドレスの CPU のプロセスイメージへのマッピングを示します。

Modbus ファンクション						S7-1200	
コード	ファンクション	データ領域	アドレス空間			データ領域	CPU アドレス
01	読み取り: ビット	Output	1	終了	8192	プロセスイメージ出力	Q0.0 ~ Q1023.7
02	読み取り: ビット	Input	10001	終了	18192	プロセスイメージ入力	I0.0 ~ I1023.7
04	読み取り: WORD	Input	30001	終了	30512	プロセスイメージ入力	IW0 ~ IW1022
05	書き込み: ビット	Output	1	終了	8192	プロセスイメージ出力	Q0.0 ~ Q1023.7
15	書き込み: ビット	Output	1	終了	8192	プロセスイメージ出力	Q0.0 ~ Q1023.7

ファンクションコード 3、6、および 16 の受信 Modubus アラームは、Modbus 保持レジスタ(保持レジスタは MB_HOLD_REG パラメータで指定)に書き込み、または Modbus 保持レジスタから読み出しを行います。

複数のサーバー接続

複数のサーバー接続を作成することが可能です。 これによって、単一の CPU が同時に 1 つ以上の Modbus TCP クライアントに対して接続を確立することが可能になります。

Modbus TCP サーバーは複数の TCP 接続をサポートしています。接続の最大数は、使用している CPU によって異なります。

Modbus TCP クライアントおよびサーバーの接続を含む 1 つの CPU の接続合計数は、サポートされている接続の最大数を超えてはなりません。

Modbus TCP 接続は、クライアント、またはサーバー、またはその両方の接続で共有することが可能です。

サーバー接続の場合は、以下のルールに注意してください。

- 各「MB_SERVER」接続は、固有のインスタンス DB を使用する必要があります。
- 各「MB_SERVER」接続は、固有の IP ポート番号で作成する必要があります。 各ポートに対して、1 つの接続のみサポートされます。
- 各「MB_SERVER」接続は、固有の接続 ID を使用する必要があります。

  命令の各インスタンス DB に対しては、関連する個別の接続 ID を使用する必要があります。 接続 ID およびインスタンス DB は対となり、各接続に対して固有である必要があります。
- 「MB_SERVER」命令は、各接続に対し個別に呼び出される必要があります。

Modbus 診断ファンクション

次の表に、Modbus 診断ファンクションの説明を記載しています。

コード	サブファンクション	説明
08	0x0000	エコーテスト:「MB_SERVER」命令は、データワードを受信し、これを変更せずに Modbus マスタに返します。
08	0x000A	イベントカウンタのリセット:「MB_SERVER」命令は、Modbus ファンクション 11 に使用される通信用のイベントカウンタをリセットします。
11	-	通信イベントカウンタのフェッチ:「MB_SERVER」命令は通信用の内部イベントカウンタを使用し、Modbus サーバーに送信された、正常に実行された読み取りおよび書き込み要求の回数を記録します。 このイベントカウンタは、ファンクション 8、11、またはブロードキャスト要求ではインクリメントされません。 通信エラー(たとえばパリティエラーや CRC エラー)となる要求の場合もインクリメントされません。 同時に存在可能なクライアント/サーバー通信は 1 つのみであるため、Modbus TCP ではブロードキャストファンクションは使用できません。

命令の静的タグ

以下の表では、プログラム内で使用される「MB_SERVER」命令のインスタンスデータブロックの静的タグについて説明します。 HR_Start_Offset タグを書き込むことができます。 他のタグを読み取り、Modbus ステータスをモニタすることができます。

タグ	データタイプ	開始値	説明
HR_Start_Offset	WORD	0	Modbus 保持レジスタの開始アドレスを割り当てます。
Request_Count	WORD	0	サーバーが受信した要求の合計数です。
Server_Message_Count	WORD	0	受信した関連するサーバーへのアラームの合計数です。

Xmt_Rcv_Count	WORD	0	エラーが発生した転送数を検知するためのカウンタです。このカウンタは、無効な Modbus アラームを受信した場合もインクリメントされます。
Exception_Count	WORD	0	Modbus が例外を発生させた場合の工ラー数を検知するためのカウンタです。
Success_Count	WORD	0	転送プロトコルにエラーを含まない要求数を検知するためのカウンタです。
Connected	BOOL	0	割り当てられたクライアントへの接続が確立されているかどうかを示します: 1 = 接続済み、0 = 未接続。

例: 静的タグ HR_Start_Offset でのアドレス指定。

Modbus 保持レジスタのアドレスは、40001 から開始します。これらのアドレスは、保持レジスタ用 CPU メモリ領域のアドレス空間に対応します。 Modbus 保持レジスタの開始アドレスが 40001 以外となるよう、HR_Start_Offset タグを定義することも可能です。

例: 保持レジスタを MW100 から開始し、長さを 100 WORD とします。 HR_Start_Offset パラメータのオフセット値は、保持レジスタの開始アドレスが 40001 から 40021 に移動したことを意味します。保持レジスタが 40021 より小さいアドレス、および 40119 よりも大きいアドレスにアドレス指定されると、必ずエラーが発生します。

HR_Start_Offset	アドレス	最小値	最大値
0	Modbus アドレス(WORD)	40001	40099
	S7-1200 アドレス	MW100	MW298
20	Modbus アドレス(WORD)	40021	40119
	S7-1200 アドレス	MW100	MW298

MB_HOLD_REG パラメータ

説明

MB_HOLD_REG パラメータは、Modbus サーバーから読み取られた、または Modbus サーバーに書き込まれたデータの格納用データバッファに対するポインタです。 グローバルデータブロックまたはメモリ領域(M)をデータバッファとして使用することが可能です。

メモリ領域(M)内のバッファに対するポインタとして、次の要領で ANY 形式を使用します。「P#ビットアドレス」「データタイプ」「長さ」(例: P#M1000.0 WORD 500).

以下の表に、Modbus ファンクション 3 (WORD の読み取り)、6 (WORD の書き込み)、および 16 (複数 WORD の書き込み)用の保持レジスタに対する Modbus アドレスのマッピング例を示します。 データブロックのアドレス数の上限値は、CPU の最大ワークメモリによって決まります。 メモリ領域を使用する場合は、アドレスの上限値は CPU のメモリ領域のサイズによって決まります。

Modbus アドレス	MB_HOLD_REG パラメータ - 例		
	P#M100.0 WORD 5	P#DB10.DBx0.0 WORD 5	"Recipe".ingredient
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

パラメータステータス

パラメータ STATUS(全般的なステータス情報)

STA-TUS* (W#16#)	説明
0000	命令がエラーなしで実行されました。
0001	接続が確立されました。
0003	接続が終了しました。
7000	動作中の呼び出しはありません(REQ=0)。
7001	REQ=1 による最初の呼び出し: 処理が開始されます。BUSY の値は 1 です。
7002	中間呼び出し(REQ は無関係)。 処理が既に有効です。BUSY の値は 1 です。
7003	接続を終了中です。
7004	接続が確立され、モニタされています。ジョブ処理が行われていません。
7005	データが送信されました。
7006	データが受信されました。
80BB	ACTIVE_EST パラメータの無効な値(接続確立のタイプの識別子、T_CON_PARAM を参照): • パッシブ接続の確立のみがサーバーに対して許可されています(ACTIVE_EST = FALSE)。 • アクティブ接続の確立のみがクライアントに対して許可されています(ACTIVE_EST = TRUE)。
8380	受信された Modbus フレームの形式が不正であるか、または受信されたバイト数が少なすぎます。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

パラメータ STATUS (パラメータエラー)

STATUS* (W#16#)	Modbus サーバーへの応答コード (B#16#)	説明
8187	応答なし	MB_HOLD_REG パラメータのポインタが無効です。 データ領域が小さすぎます。
818C	応答なし	• MB_HOLD_REG パラメータは、最適化されたデータブロックを参照します。 データブロックを標準アクセスで使用するか、メモリ領域を使用してください。 • 実行のタイムアウトによるエラー(55 秒以上)。
8381	01	ファンクションコードはサポートされていません。
8382	03	データ長のエラー

8383	02	データアドレスのエラー、または保持レジスタ(MB_HOLD_REG パラメータ)のアドレス領域外へのアクセス。
8384	03	データ値のエラー
8385	03	診断コードの値がサポートされていません(ファンクションコード 08 のみ)。
*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

注記

内部で使用される通信命令のエラーコード。

「MB_SERVER」命令を使用すると、表にリストされたエラーに加えて、命令によって使用される通信命令(「TCON」、「TDISCON」、「TSEND」、および「TRCV」)が原因のエラーが発生する可能性があります。

エラーコードが、「MB_SERVER」命令のインスタンスデータブロックを介して割り当てられます。 それぞれの命令に対して、エラーコードが Static セクションの STATUS に表示されます。

エラーコードの意味は、対応する通信命令の文書で入手できます。

例

この章には下記に関する情報が記載されています：

- MB_CLIENT 例 1: TCP 接続を介した複数要求の送信 (S7-1200)
- MB_CLIENT 例 2: 複数の TCP 接続を介した複数要求の送信 (S7-1200)
- MB_CLIENT 例 3: 複数要求の調整 (S7-1200)
- MB_SERVER 例: 複数の TCP 接続 (S7-1200)

MB_CLIENT 例 1: TCP 接続を介した複数要求の送信

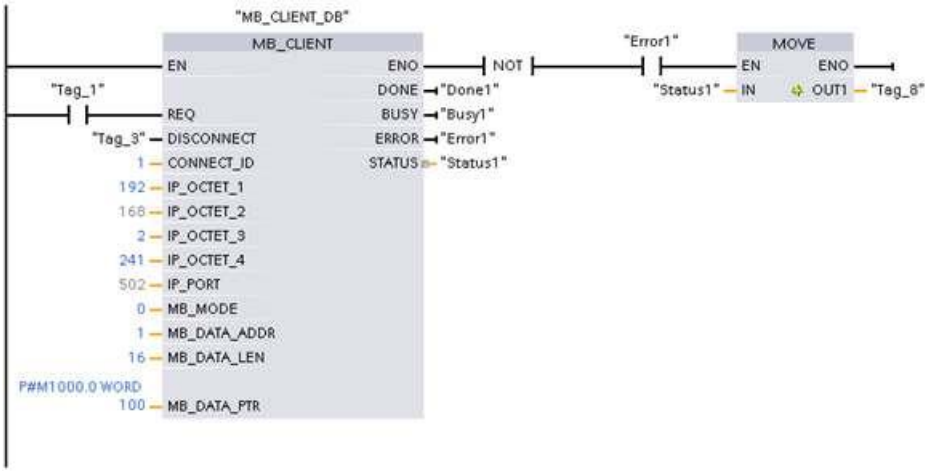
説明

Modbus クライアントの複数の要求を TCP 接続経由で送信することが可能です。 これを行うには、同一のインスタンス DB、同一の接続 ID、および同一のポート番号を使用します。

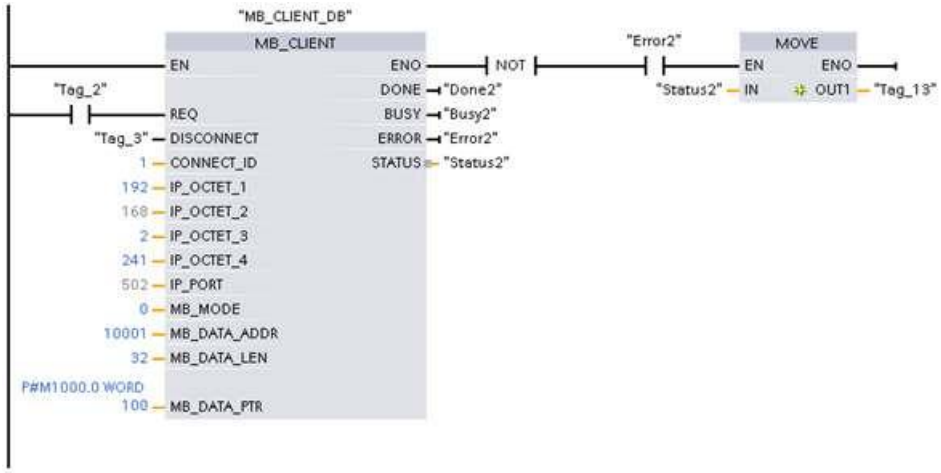
1 回に 1 つのクライアントのみを有効にすることができます。 クライアントの処理完了後、次のクライアントを処理します。 実行順序はプログラム内で定義する必要があります。

以下のサンプルプログラムでは、STATUS 出力パラメータの値もコピーされます。

ネットワーク 1: Modbus ファンクション 1 - 16 出力ビットの読み取り



ネットワーク 2: Modbus ファンクション 2 - 32 出力ビットの読み取り



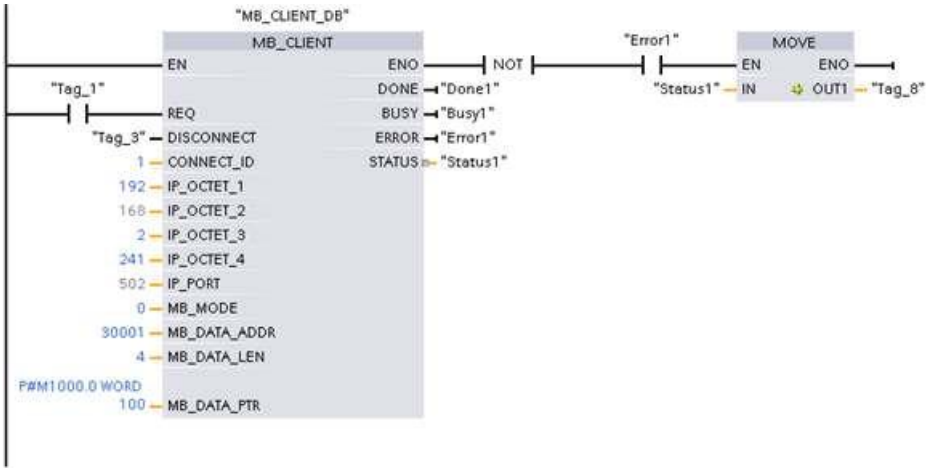
MB_CLIENT 例 2: 複数の TCP 接続を介した複数要求の送信

説明

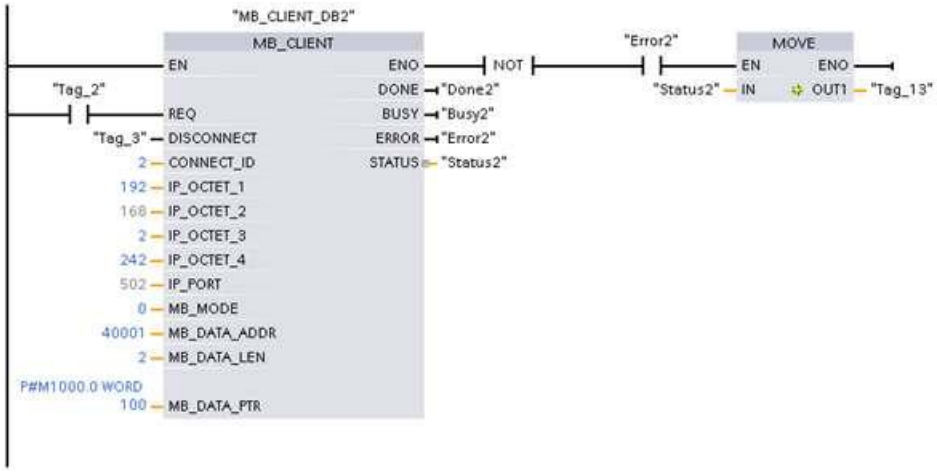
Modbus クライアントからの要求を異なる TCP 接続経由で送信することが可能です。 これを行うには、異なるインスタンス DB および異なる接続 ID を使用します。

同一の Modbus サーバーに対する接続である場合は、異なるポート番号を使用します。 異なる Modbus サーバーに対する接続である場合は、ポート番号を自由に割り当てることが可能です。

ネットワーク 1: Modbus ファンクション 4 - 入力の読み取り(WORD)



ネットワーク 2: Modbus ファンクション 3 - 保持レジスタの読み取り(WORD)

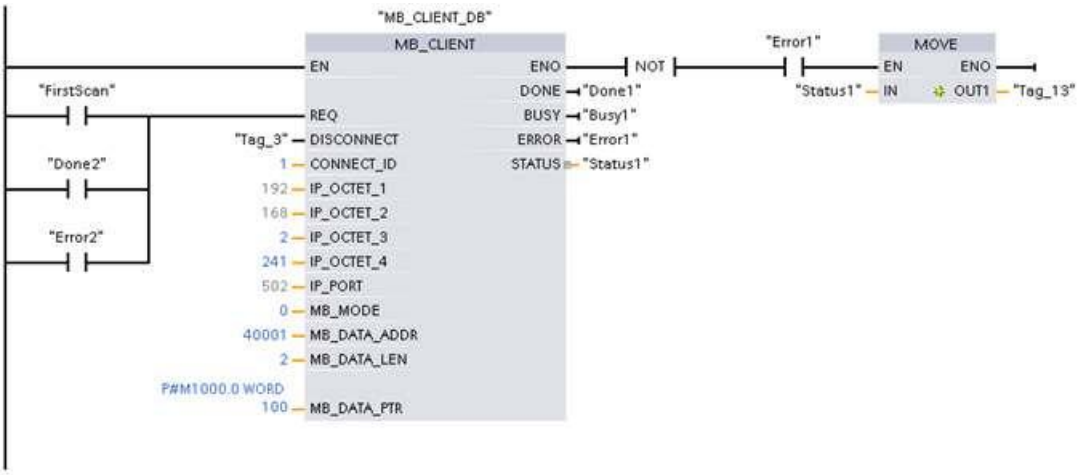


MB_CLIENT 例 3: 複数要求の調整

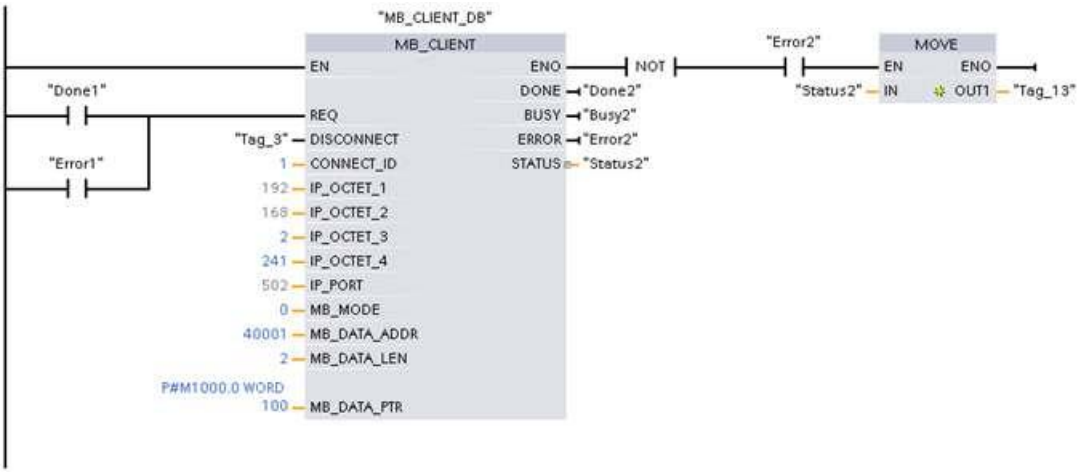
説明

個々の Modbus 要求が実行されていることを確認します。 要求の調整は、プログラムで制御します。以下の例は、1 番目および 2 番目のクライアント要求の出力パラメータが、命令の実行の調整にどのように使用できるかを説明しています。

ネットワーク 1: Modbus ファンクション 3 - 保持レジスタの読み取り(WORD)



ネットワーク 2: Modbus ファンクション 3 - 保持レジスタの読み取り(WORD)



MB_SERVER 例: 複数の TCP 接続

説明

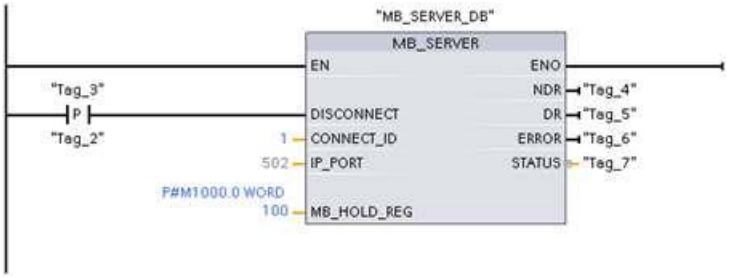
複数の Modbus サーバー接続を使用することが可能です。 これを行うには、「MB_SERVER」命令を各接続に対して個別に呼び出す必要があります。

各接続には、以下のことが要求されます。

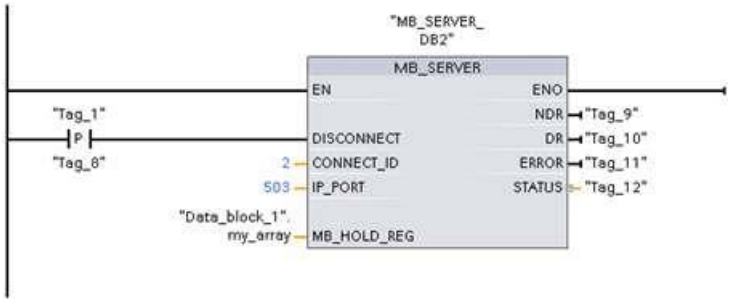
- 命令の独立したインスタンスデータブロック
- 固有の接続 ID
- 個別の IP ポート(S7-1200 では、1 つの IP ポートにつき 1 つの接続のみが許可されています)

パフォーマンスを最適化するために、「MB_SERVER」は各接続に対してプログラムサイクルごとに 1 回実行される必要があります。

ネットワーク 1: 関連する IP ポート接続 ID およびインスタンス DB での接続#1



ネットワーク 2: 関連する IP ポート接続 ID およびインスタンス DB での接続#1



MODBUS (TCP)

この章には下記に関する情報が記載されています：

- MB_CLIENT:PROFINET を介した Modbus TCP クライアントとしての通信 (S7-1200, S7-1500)
- MB_SERVER:PROFINET を介した Modbus TCP サーバーとしての通信 (S7-1200, S7-1500)

MB_CLIENT:PROFINET を介した Modbus TCP クライアントとしての通信

この章には下記に関する情報が記載されています：

MB_CLIENT の説明

説明

「MB_CLIENT」命令は、PROFINET 接続を経由して Modbus TCP クライアントとして通信します。「MB_CLIENT」命令を使用し、クライアントとサーバー間の接続を確立し、Modbus 要求の送信、応答の受信、および Modbus TCP クライアントの制御接続終了を行います。

「MB_CLIENT」命令 V3.0 以降は、S7-1200 バージョン 4.0 以降と同様に、S7-1500 でも使用できます。 この接続は、CPU または CM/CP のローカルインターフェースを経由して実行できます。

この命令を使用するためにハードウェアモジュールを追加する必要はありません。

複数クライアントの接続

Modbus TCP クライアントは複数の TCP 接続をサポートしています。接続の最大数は、使用している CPU によって異なります。 Modbus TCP クライアントおよびサーバーの接続を含む 1 つの CPU の接続合計数は、サポートされている接続の最大数を超えてはなりません。 Modbus TCP 接続は、「MB_CLIENT」および「MB_SERVER」インスタンスのいずれかまたは両方によって共有することもできます。

個別のクライアント接続では、以下のルールに注意してください。

- 各「MB_CLIENT」接続は、固有のインスタンス DB を使用する必要があります。
- 各「MB_CLIENT」接続では、固有のサーバー IP アドレスが指定されている必要があります。
- 各「MB_CLIENT」接続は、固有の接続 ID を必要とします。

  命令の各インスタンス DB に対しては、関連する個別の接続 ID を使用する必要があります。 接続 ID およびインスタンス DB は対となり、各接続に対して固有である必要があります。
- 固有の IP ポート番号が必要かどうかは、サーバー設定によって異なります。

パラメータ

以下の表に、「MB_CLIENT」命令のパラメータを示します。

パラメータ	宣言	データタイプ	説明
REQ	Input	BOOL	Modbus TCP サーバーとの通信要求 REQ パラメータがレベル制御されます。 すなわち、この入力がセットされている限り(REQ=true)、この命令は通信要求を送信します。 • その他のクライアントのインスタンス DB は、通信要求でブロックされます。 • サーバーが応答するか、またはエラーメッセージが出力されるまで、入力パラメータへの変更は有効になりません。 • Modbus 要求の進行中にパラメータ REQ が再度設定された場合、その後は追加の送信は実行されません。
DISCONNECT	Input	BOOL	このパラメータを使用し、Modbus サーバーへの接続の確立および終了を制御します。

			• 0: CONNECT パラメータで設定された接続パートナーへの通信接続を確立します(CONNECT パラメータを参照)。 • 1: 通信接続を切断します。 接続終了中は、他のファンクションは実行されません。 正常な接続の終了の後に、値 0003 が STATUS パラメータに出力されます。 接続の確立中に REQ パラメータがセットされた場合、Modbus 要求が直ちに送信されます。
MB_MODE	Input	USINT	Modbus 要求のモード(読み取り、書き込み、または診断)を選択します。
MB_DATA_ADDR	Input	UDINT	「MB_CLIENT」命令によってアクセスされるデータの開始アドレス。
MB_DATA_LEN	Input	UINT	データ長: データアクセス用ビットまたはワードの数(「MB_MODE および MB_DATA_ADDR パラメータ」 - データ長を参照してください)。
MB_DATA_PTR	InOut	VARIANT	Modbus データレジスタへのポインタ。 このレジスタは、Modbus サーバーから受信する、または Modbus サーバーに送信するデータ用のバッファです。 ポインタは、グローバルデータブロックを最適化されたアクセスで参照する必要があります。 アドレス指定されたビットの数は、8 で割り切れる必要があります。
CONNECT	InOut	VARIANT	接続記述子の構造へのポインタ 以下の構造体(システムデータタイプ)を使用できます。 • TCON_IP_v4: プログラミングされた接続の確立に必要なすべてのアドレスパラメータを含みます。 TCON_IP_v4 を使用している場合、命令「MB_SERVER」を呼び出しているときに接続が確立されます。 • TCON_Configured: 設定した接続のアドレスパラメータを含みます。 TCON_Configured を使用している場合、ハードウェアコンフィグレーションのダウンロード後に、CPU によって作成された既存の接続が使用されます。
DONE	Out	BOOL	最後のジョブが正常に完了すると、すぐに出力パラメータ DONE のビットが「1」にセットされます。
BUSY	Out	BOOL	• 0: 進行中の Modbus 要求はありません • 1: Modbus 要求を処理しています。 出力パラメータ BUSY は、接続の確立および終了中には設定されません。
ERROR	Out	BOOL	• 0: エラーは発生していません。 • 1: エラーが発生しました。 エラーの原因は、STATUS パラメータによって識別されます。
STATUS	Out	WORD	命令の詳細なステータス情報。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

注記

「MB_CLIENT」呼び出し中の一貫した入力データ。

Modbus クライアント命令が呼び出されると、入力パラメータのステータスが内部的に保存され、次回の呼び出し時に比較されます。 この比較は、その特定の呼び出しが現在の要求を初期化したかどうかを判定するために使用されます。 共通のインスタンス DB を使用している場合、複数の「MB_CLIENT」呼び出しを実行できます。 「MB_CLIENT」インスタンスの実行中は、入力パラメータの値を変更してはなりません。 実行中に入力パラメータが変更されると、インスタンスを現在実行中か否かのチェックに「MB_CLIENT」を使用できなくなります。

命令の静的タグ

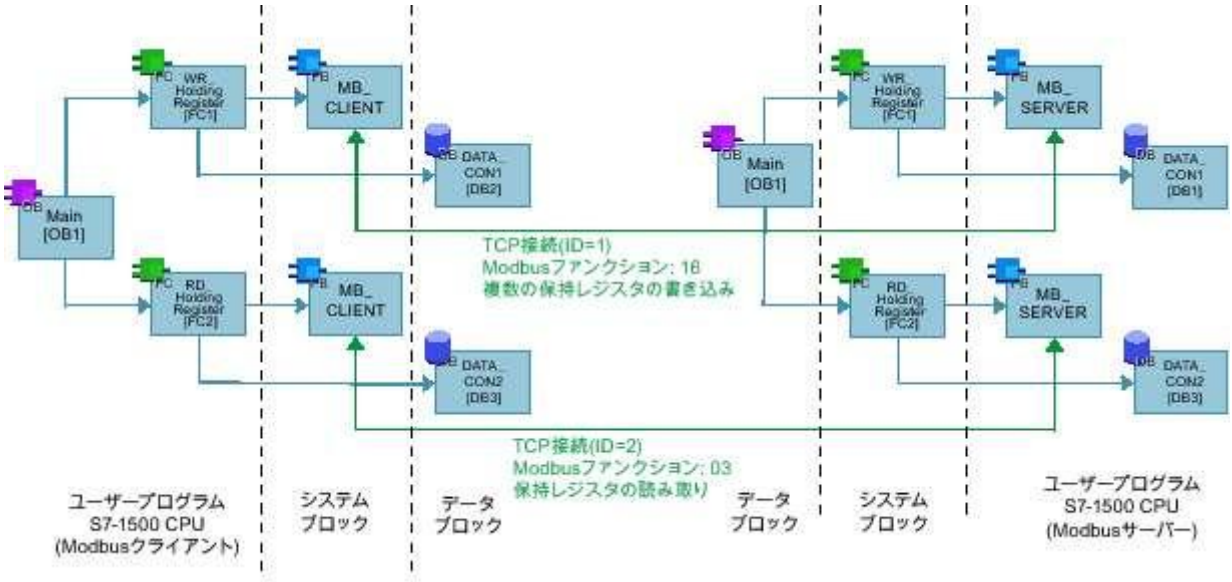
以下の表では、「MB_CLIENT」命令のインスタンスデータブロックの編集可能な静的タグについて説明します。

タグ	データタイプ	開始値	説明
Blocked_Proc_Timeout	REAL	3.0	ブロックされた Modbus インスタンスが存在する場合、静的タグ ACTIVE がリセットされるまでの待機秒数。 これは、たとえばクライアントが要求を出力し、その要求が完全に実行される前にクライアントファンクションの実行が中止された場合などに発生します。 最大待機時間は 55 秒です。
MB_Transaction_ID	WORD	1	Modbus TCP プロトコルのトランザクション ID。 開始値「1」は、Modbus TCP サーバーが異なる値を要求する場合にのみ変更されます。
MB_Unit_ID	BYTE	255	Modbus デバイスの検出: Modbus TCP サーバーは、IP アドレスを使用してアドレス指定します。 このため、MB_UNIT_ID パラメータは Modbus TCP アドレス指定の場合には使用されません。 MB_UNIT_ID パラメータは、Modbus RTU プロトコルのスレーブアドレスのフィールドに対応します。 Modbus TCP サーバーを Modbus RTU プロトコルのゲートウェイとして使用する場合、MB_UNIT_ID を使用してシリアルネットワークのスレーブデバイスを識別できます。 この場合、MB_UNIT_ID パラメータは正しい Modbus RTU スレーブアドレスに要求を転送します。 Modbus TCP デバイスによっては、制限された値の範囲内で初期化を行うために MB_UNIT_ID パラメータを要求する場合があります。
RCV_TIMEOUT	REAL	2.0	「MB_CLIENT」命令がサーバーからの応答を待機する時間間隔の秒数。
Connected	BOOL	0	割り当てられたサーバーへの接続が確立されているかどうかを示します。 1 = 接続済み、0 = 未接続。

例

2 つの S7-1500 CPU 間の Modbus TCP 通信のサンプルプロジェクトは、エントリ ID 94766380 のサービスおよびサポートポータルにあります。

このサンプルでは 2 つの Modbus ファンクションが使用されています。 各 Modbus ファンクションでは、Modbus ブロックペア(MB_CLIENT および MB_SERVER)を使用して Modbus TCP 接続が確立されます。



REQ および DISCONNECT パラメータ

説明

実行中の「MB_CLIENT」命令のインスタンスがなく、DISCONNECT パラメータの値が「0」であると、REQ=1 の場合、新しいジョブが実行されます。 接続が存在していない場合は、実行中に接続が確立されます。

アクティブなジョブの実行前に「MB_CLIENT」命令の同一のインスタンスが再度実行されると(DISCONNECT = 0 および REQ = 1)、アクティブなジョブの完了時にはこのインスタンスは実行されません。 新しいジョブは、アクティブなジョブの完了時(REQ = 1)にのみ開始することが可能です。

実行のステータスが、出力パラメータによって出力されます。 「MB_CLIENT」命令が順番に実行される場合、このパラメータを使用して実行のステータスをモニタすることが可能です。

MB_MODE および MB_DATA_ADDR パラメータ

説明

「MB_CLIENT」命令は、ファンクションコードの代わりに MB_MODE パラメータを使用します。MB_DATA_ADDR パラメータを使用し、アクセスするデータの Modbus 開始アドレスを指定します。

パラメータ MB_MODE、MB_DATA_ADDR、および MB_DATA_LEN を組み合わせ、現在の Modbus メッセージで使用するファンクションコードを定義します。 たとえば、

- ファンクションコード 5
  - MB_MODE=1
  - MB_DATA_ADDR=1
  - MB_DATA_LEN=1
- ファンクションコード 15
  - MB_MODE=1
  - MB_DATA_ADDR=1
  - MB_DATA_LEN=2

次の表に、「MB_CLIENT」命令の入力パラメータと Modbus ファンクションの関係を示します。

MB_MODE	MB_DATA_ADDR	MB_DATA_LEN	Modbus ファンクション	ファンクションおよびデータタイプ
0	開始アドレス: • 1～9999	1 回の呼び出しごとのデータ長(ビット): • 1～2000	01	出力ビットの読み出し: • 1～2000
0	開始アドレス: • 10001～19999	1 回の呼び出しごとのデータ長(ビット): • 1～2000	02	入力ビットの読み取り: • 1～2000
0	開始アドレス: • 40001～49999 • 400001～465535	1 回の呼び出しごとのデータ長(WORD): • 1～125 • 1～125	03	保持レジスタの読み取り: • 0～9998 • 0～65534
0	開始アドレス: • 30001～39999	1 回の呼び出しごとのデータ長(WORD): • 1～125	04	入力ワードの読み取り: • 0～9998

1	開始アドレス: • 1 ~ 9999	1 回の呼び出しごとのデータ長(ビット): • 1	05	出力ビットの書き込み: • 0 ~ 9998
1	開始アドレス: • 40001 ~ 49999 • 400001 ~ 465535	1 回の呼び出しごとのデータ長 (WORD): • 1 • 1	06	1 つの保持レジスタの書き込み: • 0 ~ 9998 • 0 ~ 65534
1	開始アドレス: • 1 ~ 9999	1 回の呼び出しごとのデータ長(ビット): • 2 ~ 1968	15	複数の出力ビットの読み取り: • 0 ~ 9998
1	開始アドレス: • 40001 ~ 49999 • 400001 ~ 465535	1 回の呼び出しごとのデータ長 (WORD): • 2 ~ 123 • 2 ~ 123	16	複数の保持レジスタの書き込み • 0 ~ 9998 • 0 ~ 65534
2	開始アドレス: • 1 ~ 9999	1 回の呼び出しごとのデータ長(ビット): • 1 ~ 1968	15	1 つ以上の出力ビットの書き込み: • 0 ~ 9998
2	開始アドレス: • 40001 ~ 49999 • 400001 ~ 465535	1 回の呼び出しごとのデータ長 (WORD): • 1 ~ 123 • 1 ~ 123	16	1 つ以上の保持レジスタの書き込み: • 0 ~ 9998 • 0 ~ 65534
11	このファンクションが実行される場合、MB_DATA_ADDR および MB_DATA_LEN パラメータは評価されません。		11	サーバーのステータスワードおよびイベントカウンタの読み取り: • ステータスワードは処理ステータスを表します(0 - 処理中ではない、0xFFFF - 処理中)。 • このイベントカウンタは、Modbus 要求が正常に実行された場合にインクリメントされます。 Modbus ファンクションの実行中にエラーが発生した場合、サーバーによってメッセージが送信されますが、イベントカウンタはインクリメントされません。
80	-	1 回の呼び出しごとのデータ	08	診断コード 0x0000 でサーバーステータスをチェック(戻り値ループテスト - サーバーが要求を返信します):

		長 (WORD): • 1		• 1 1 回の呼び出しにつき WORD
81	-	1 回の呼び出しごとのデータ長 (WORD): • 1	08	診断コード 0x000A でサーバーのイベントカウンタをリセット: • 1 1 回の呼び出しにつき WORD
3 ~ 10、12 ~ 79、82 ~ 255				予約済み

MB_DATA_PTR パラメータ

説明

MB_DATA_PTR パラメータは、Modbus サーバーから読み取られた、または Modbus サーバーに書き込まれたデータの格納用データバッファに対するポインタです。 グローバルデータブロックまたはメモリ領域(M)をデータバッファとして使用することが可能です。

メモリ領域(M)内のバッファの場合、次の要領で ANY 形式のポインタを使用します。「P#ビットアドレス」「データタイプ」「長さ」(例: P#M1000.0 WORD 500).

以下の場合、MB_DATA_PTR パラメータは通信バッファを使用します。

通信バッファを使用:

- 「MB_CLIENT」命令の通信ファンクションの場合:
  - Modbus サーバーアドレス 00001 ~ 09999 および 10001 ~ 19999 のデータの 1 ビットの読み取りおよび書き込み。
  - Modbus サーバーアドレス 30001 ~ 39999 および 40001 ~ 49999 の 16 ビット WORD データの読み出し。
  - Modbus サーバーアドレス 40001 ~ 49999 の 16 ビット WORD データの書き込み。
- データ伝送中(長さ: MB_DATA_PTR パラメータで割り当てたグローバル DB またはメモリ領域(M)とのデータ転送中(長さ:ビットまたは WORD)。

MB_DATA_PTR パラメータ内でバッファ用のデータブロックを使用する場合、DB エレメントにデータタイプを割り当てる必要があります。

- Modbus ビットアドレスには、1 ビットデータタイプ BOOL を使用します。
- ModbusWORD アドレスには、WORD、UINT、INT、または REAL などの 16 ビットデータタイプを使用します。
- 2 つの ModbusWORD アドレスには、DWORD、DINT、または REAL などの 32 ビットデータタイプ(ダブルワード)を使用します。
- MB_DATA_PTR では、以下のような複合 DB エレメントにもアクセス可能です。
  - 標準配列
  - エレメント名が一意の構造体
  - エレメント名が一意、かつデータタイプ長が 16 または 32 ビットの複合構造体。
- MB_DATA_PTR パラメータのデータ領域は、別のグローバルデータブロック(または別のメモリ領域)にも確保することができます。 例えば、読み取りジョブと書き込みジョブで異なるデータブロックを使用することや、各「MB_CLIENT」ステーションに対して個別のデータブロックを使用することが可能です。

CONNECT パラメータ

CONNECT パラメータの接続記述子

「MB_CLIENT」命令には、2 つの異なる接続記述子を使用できます。

- 構造 TCON_IP_v4 によるプログラミングされた接続

  接続パラメータは TCON_IP_v4 構造体に保存され、接続は命令「MB_CLIENT」の呼び出しによってセットアップされます。

- 構造 TCON_Configured による設定した接続

  設定した接続は、既に CPU によって確立されています。 構造 TCON_Configured を使用して、命令に対して使用する既存の接続を指定します。

命令「MB_CLIENT」の各インスタンスには、一意の接続が必要です。 命令の各インスタンスに対して個別の構造 TCON_IP_v4 または TCON_Configured を作成して、接続を記述します。

プログラミングされた接続の接続記述子

CONNECT パラメータでプログラミングされた接続の場合、接続の記述には TCON_IP_v4 に従って以下の構造を使用します。

- タイプ TCP の接続のみが TCON_IP_v4 構造に指定されていることを確認します。
- この接続は、以下の TCP ポート番号を使用できません。 20、21、25、80、102、123、5001、34962、34963、および 34964。

バイト	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceID	HW_ANY	-	ローカルインターフェースのハードウェア識別子(値の範囲:0 ~ 65535)。
2 … 3	ID	CONN_OUC	-	この接続への参照(値の範囲:1 ~ 4095)。 このパラメータは、CPU 内の接続を一意に識別します。 命令「MB_CLIENT」の個々のインスタンスは、固有の ID を使用する必要があります。
4	Connection-Type	BYTE	11	接続タイプ TCP の場合は、11 (10 進数)を選択します。 その他の接続タイプは、許可されていません。 別の接続タイプ(UDP など)が使用された場合、対応するエラーメッセージが命令の STATUS パラメータで出力されます。
5	ActiveEstab-lished	BOOL	TRUE	接続の確立方法の ID アクティブ接続の確立の場合は、TRUE を選択します。
6 … 9	RemoteAd-dress	ARRAY [1..4] of BYTE	-	接続パートナーの IP アドレス(Modbus サーバー)、たとえば、192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0

				• addr[4] = 1
10 … 11	RemotePort	UINT	502	リモート接続パートナーのポート番号(値の範囲: 1～49151)。 TCP/IP プロトコルを使用してクライアントが接続および通信を確立するサーバーの IP ポート番号を使用(既定値: 502).
12 … 13	LocalPort	UINT	0	ローカル接続パートナーのポート番号: • ポート番号:1～49151 • 任意のポート: "0"

注記

命令「MB_CLIENT」バージョン 2.1 の移行

パラメータ CONNECT_ID、IP_PORT、および IP_OCTET_x が、「MB_CLIENT」命令のバージョン 3.0 で TCON_IP_v4 構造体にマッピングされます。

- 「MB_CLIENT」V2.1 命令のパラメータ「CONNECT_ID」は、TCON_IP_v4 のパラメータ ID に相当します。
- 「MB_CLIENT」V2.1 命令のパラメータ「IP_PORT」は、TCON_IP_v4 のパラメータ RemotePort に相当します。
- 「MB_CLIENT」V2.1 命令の 4 つのパラメータ「 IP_OCTET_x 」は、TCON_IP_v4 のパラメータ RemoteAddress の配列に相当します。

設定した接続の接続記述子

CONNECT パラメータでプログラミングした接続の場合、接続の記述には TCON_Configured.に従って以下の構造を使用します。

- タイプ TCP の接続のみが TCON_Configured 構造に指定されていることを確認します。
- この接続は、以下の TCP ポート番号を使用できません。20、21、25、80、102、123、5001、34962、34963、および 34964。

バイト	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceId	HW_ANY	-	ローカルインターフェースのハードウェア識別子(値の範囲:0～65535)。
2 … 3	ID	CONN_OUC	-	この接続への参照(値の範囲:1～4095)。 既存の接続の接続 ID を入力します。
4	Connection-Type	BYTE	0	接続タイプ 設定した接続に対して 254 (10 進数)を選択します。

パラメータステータス

パラメータ STATUS (全般的なステータス情報)

STATUS* (W#16#)	説明
0000	命令がエラーなしで実行されました。
0001	接続が確立されました。
0003	接続が終了しました。
7000	動作中の呼び出しはなく、接続は確立されません(REQ=0)。
7001	接続の確立がトリガされました。
7002	中間呼び出し。 接続を確立中です。
7003	接続を終了中です。
7004	接続が確立され、モニタされています。ジョブ処理が行われていません。
7005	データを送信中です。
7006	データを受信中です。
80BB	ActiveEstablished パラメータの無効な値(接続確立のタイプの識別子、CONNECT パラメータを参照): • パッシブ接続の確立のみがサーバーに対して許可されています(ActiveEstablished = FALSE)。 • アクティブ接続の確立のみがクライアントに対して許可されています(ActiveEstablished = TRUE)。
8380	受信された Modbus フレームの形式が不正であるか、または受信されたバイト数が少なすぎます。
*ステータスコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

パラメータ STATUS(プロトコルエラー)

STATUS* (W#16#)	MB_SERVER からのエラーメッセージのエラーコード(B#16#)	説明
8381	01	ファンクションコードはサポートされていません。
8382	03	データ長のエラー: • 受信した Modbus フレームでの不正な長さ指定。 • フレームヘッダーの Modbus フレームの長さが、受信したバイト数と一致しません。 • バイト数が、実際に送信されたバイト数と一致しません(ファンクション 1-4 のみ)。 • 受信した開始アドレスが、はじめに送信された開始アドレスと一致しません(ファンクション 5、6、15、16)。

		• ワード数が、実際に送信されたワード数と一致しません(ファンクション 15 および 16 のみ)。
8383	02	データの読み出しまたは書き込みのエラー、または MB_DATA_PTR のアドレス領域外へのアクセス。 命令 MB_SERVER を使用する場合と同様に、ローカルにエラーが発生する可能性があります。
8384	03	データ値のエラー: • ファンクション 5 のデータ値のエラー(サーバーのエラー)。 • クライアントによってはじめに送信されたデータ値とは異なるデータ値が受信されました(ファンクション 5 および 6) (ローカルエラー)。 • 無効な例外コードが受信されました。
8385	03	診断コードはサポートされていません(ファンクションコード 08)。 サーバー上と同様に、ローカルにエラーが発生する可能性があります。
8386	-	受信したファンクションコードが、はじめに送信されたファンクションコードと一致しません。
8387	-	• 割り当てられた接続 ID が、前回の要求で使用されたものと異なります。「MB_CLIENT」命令の各インスタンス DB に対して使用できる接続 ID は 1 つのみです。 • サーバーによって受信される Modbus TCP フレームのプロトコル ID が「0」でない場合も、このエラーコードが出力されます。
8388	15 または 16	Modbus サーバーが、要求されたデータ長とは異なるデータ長を送信しました。 このエラーは、Modbus ファンクション 15 または 16 の使用時にのみ発生します。
*ステータスコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

パラメータ STATUS (パラメータエラー)

STATUS* (W#16#)	説明
80B6	無効な接続タイプ。TCP 接続のみがサポートされています。
80C8	定義された期間内にサーバーの応答がありませんでした。 Modbus サーバーへの接続をチェックしてください。 このエラーは、設定された反復試行が終了したときにのみレポートされます。 「MB_CLIENT」命令が、転送時のトランザクション ID(MB_TRANSACTION_ID タグを参照)の応答を定義された期間内に受信しない場合、このエラーコードが出力されます。
8188	MB_MODE パラメータの値が無効です。
8189	MB_DATA_ADDR パラメータのデータのアドレス指定が無効です。
818A	MB_DATA_LEN パラメータのデータ長が無効です。
818B	MB_DATA_PTR パラメータのポインタが無効です。 MB_DATA_ADDR および MB_DATA_LEN パラメータの値もチェックする必要があります。
818C	パラメータ BLOCKED_PROC_TIMEOUT または RCV_TIMEOUT でのタイムアウト(命令の静的タグを参照)。 限界値の 55 秒を超えました。

818D	トランザクション ID (MB_TRANSACTION_ID タグ)が、はじめに送信された ID と一致しません(命令の静的タグを参照)。
8200	• 現在、このポート経由で別の Modbus 応答が処理されています。 • 同じ接続パラメータの MB_CLIENT の別のインスタンスが、既存の Modbus 要求を処理しています。
*ステータスコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

注記

内部で使用される通信命令のエラーコード

「MB_CLIENT」命令を使用すると、表にリストされたエラーに加えて、命令によって使用される通信命令(「TCON」、「TDISCON」、「TSEND」、「TRCV」、「T_DIAG」、および「TRESET」)が原因のエラーが発生する可能性があります。

エラーコードが、「MB_CLIENT」命令のインスタンスデータブロックを介して割り当てられます。それぞれの命令に対して、エラーコードが"Static"セクションの STATUS に表示されます。

エラーコードの意味は、対応する通信命令の文書で入手できます。

MB_SERVER:PROFINET を介した Modbus TCP サーバーとしての通信

この章には下記に関する情報が記載されています：

- MB_SERVER の説明 (S7-1200, S7-1500)
- MB_HOLD_REG パラメータ (S7-1200, S7-1500)
- CONNECT パラメータ (S7-1200, S7-1500)
- パラメータステータス (S7-1200, S7-1500)

MB_SERVER の説明

説明

「MB_SERVER」命令は、PROFINET 接続を経由して Modbus TCP サーバーとして通信します。「MB_SERVER」命令は、Modbus TCP クライアントの接続要求の処理、Modbus の要求の受信および処理、および応答の送信を行います。

「MB_SERVER」命令 V3.0 以降は、S7-1200 バージョン 4.0 以降と同様に、S7-1500 でも使用できます。 この接続は、CPU または CM/CP のローカルインターフェースを経由して実行できます。

この命令を使用するためにハードウェアモジュールを追加する必要はありません。

通知
セキュリティ情報 ネットワークの各クライアントは、プロセスイメージ入出力や、Modbus 保持レジスタによって定義されるデータブロックまたはビットメモリ領域に対する読み出し/書き込みのアクセスが与えられていることに注意してください。 このオプションは、IP アドレスへのアクセスを制限して、未許可の読み出しおよび書き込み操作を防止するために使用できます。 ただし、未許可のアクセスの場合でも、共有アドレスは使用できることに注意してください。

複数のサーバー接続

複数のサーバー接続を作成することが可能です。 これによって、単一の CPU が同時に複数の Modbus TCP クライアントからの接続を受け入れることができます。

Modbus TCP サーバーは複数の TCP 接続をサポートしています。接続の最大数は、使用している CPU によって異なります。

Modbus TCP クライアントおよびサーバーの接続を含む 1 つの CPU の接続合計数は、サポートされている接続の最大数を超えてはなりません。

Modbus TCP 接続は、「MB_CLIENT」および「MB_SERVER」インスタンスのいずれかまたは両方によって共有することもできます。

サーバー接続の場合は、以下のルールに注意してください。

- 各「MB_SERVER」接続は、固有のインスタンス DB を使用する必要があります。
- 各「MB_SERVER」接続は、固有の接続 ID を使用する必要があります。

  命令の各インスタンス DB に対しては、関連する個別の接続 ID を使用する必要があります。 接続 ID およびインスタンス DB は対となり、各接続に対して固有である必要があります。
- 「MB_SERVER」命令は、各接続に対し個別に呼び出される必要があります。

パラメータ

以下の表に、「MB_SERVER」命令のパラメータを示します。

パラメータ	宣言	データタイプ	説明
DISCONNECT	Input	BOOL	「MB_SERVER」命令を使用し、パートナーモジュールとのパッシブ接続を開始します。 サーバーが、CONNECT パ

			ラメータで SDT 「TCON_IP_v4」に入力される IP アドレスからの接続要求に応答します。 このパラメータを使用して、接続要求の承認のタイミングを制御することが可能です。 • 0: パッシブ接続は、通信接続がない場合に確立されます。 • 1: 接続終了の初期化。 入力が設定された場合、その他の操作は実行されません。 正常な接続の終了の後に、値 0003 が STATUS パラメータに出力されます。
MB_HOLD_REG	InOut	VARIANT	「MB_SERVER」命令の Modbus 保持レジスタへのポインタ 保持レジスタは、Modbus クライアントが Modbus ファンクション 3 (読み出し)、6 (書き込み)、および 16 (複数の読み出し)を使用してアクセスする可能性のある値を含んでいます。 保持レジスタとして、最適化されたアクセスのあるグローバルデータブロックまたはビットメモリのメモリ領域を使用します。
CONNECT	InOut	VARIANT	接続記述子の構造へのポインタ 以下の構造(SDT)を使用できます。 • TCON_IP_v4: プログラミングされた接続の確立に必要なすべてのアドレスパラメータを含みます。 既定のアドレスは 0.0.0.0 (任意の IP アドレス)ですが、サーバーがこのアドレスからの要求のみに応答するように、特定の IP アドレスを入力できます。 TCON_IP_v4 を使用している場合、命令「MB_SERVER」を呼び出しているときに接続が確立されます。 • TCON_Configured: 設定した接続のアドレスパラメータを含みます。 TCON_Configured を使用している場合、ハードウェアコンフィグレーションのダウンロード後に、CPU によって接続が確立されます。
NDR	Output	BOOL	"New Data Ready": • 0: 新規データなし • 1: 新規データが Modbus クライアントによって書き込まれました。
DR	Output	BOOL	"Data Read": • 0: データ読み出しなし • 1: データが Modbus クライアントによって読み取られました。
ERROR	Output	BOOL	「MB_SERVER」命令の呼び出し中にエラーが発生した場合、ERROR パラメータの出力が「1」にセットされます。問題の原因に関する詳細情報は、STATUS パラメータによって示されます。
STATUS	Output	WORD	命令の詳細なステータス情報。

有効なデータタイプに関する追加情報については、「有効なデータタイプの概要」を参照してください。

命令の静的タグ

以下の表では、プログラム内で使用される「MB_SERVER」命令のインスタンスデータブロックの静的タグについて説明します。 HR_Start_Offset タグを書き込むことができます。 他のタグを読み取り、Modbus ステータスをモニタすることができます。

タグ	データタイプ	開始値	説明
HR_Start_Offset	WORD	0	Modbus 保持レジスタの開始アドレスを割り当てます。
Request_Count	WORD	0	サーバーが受信した要求の合計数です。
Server_Message_Count	WORD	0	受信した関連するサーバーへのアラームの合計数です。
Xmt_Rcv_Count	WORD	0	エラーが発生した転送数を検知するためのカウンタです。このカウンタは、無効な Modbus 要求を受信する場合のみインクリメントされます。
Exception_Count	WORD	0	特に「MB_CLIENT」へのエラーメッセージの原因になる Modbus に対してエラー数を検知するためのカウンタです。
Success_Count	WORD	0	サーバーによって正常に実行された要求の数を検知するためのイベントカウンタです。
Connected	BOOL	0	割り当てられたクライアントへの接続が確立されているかどうかを示します: 1 = 接続済み、0 = 未接続。

Modbus アドレスのプロセスイメージへのマッピング

「MB_SERVER」命令によって、受信 Modbus ファンクション(1、2、4、5、および 15)が CPU (データタイプ BOOL および WORD の使用)のプロセスイメージ入力および出力に対し、直接読み取りおよび書き込みアクセスを実行できるようになります。

ファンクションコード 3、6、および 16 のデータ転送の場合、保持レジスタ(MB_HOLD_REG パラメータ)には 1 バイト以上が定義される必要があります。 次の表に、Modbus アドレスの CPU のプロセスイメージへのマッピングを示します。

Modbus ファンクション						S7-1500、S7-1200 V4.0	
ファンクションコード	ファンクション	データ領域	アドレス空間			データ領域	CPU アドレス
01	読み取り: ビット	Output	1	終了	9999	プロセスイメージ出力	Q0.0 ~ Q1249.6
02	読み取り: ビット	Input	1	終了	9999	プロセスイメージ入力	I0.0 ~ I1249.6
04	読み取り: WORD	Input	1	終了	9999	プロセスイメージ入力	IW0 ~ IW19996
05	書き込み: ビット	Output	1	終了	9999	プロセスイメージ出力	Q0.0 ~ Q1249.6
15	書き込み: ビット	Output	1	終了	9999	プロセスイメージ出力	Q0.0 ~ Q1249.6

ファンクションコード 3、6、16、および 23 の受信 Modbus 要求は、Modbus 保持レジスタ(保持レジスタは MB_HOLD_REG パラメータで指定)への書き込み、または Modbus 保持レジスタから読み取りを行います。

例: 静的タグ HR_Start_Offset でのアドレス指定。

Modbus 保持レジスタのアドレスは、40001 から開始します。これらのアドレスは、保持レジスタ用 CPU メモリ領域のアドレス空間に対応します。 Modbus 保持レジスタの開始アドレスが 40001 以外となるよう、HR_Start_Offset タグを定義することも可能です。

例: 保持レジスタを MW100 から開始し、長さを 100 WORD とします。 HR_Start_Offset パラメータのオフセット値は、保持レジスタの開始アドレスが 40001 から 40021 に移動したことを意味します。保持レジスタが 40021 より小さいアドレス、および 40120 よりも大きいアドレスにアドレス指定されると、必ずエラーが発生します。

HR_Start_Offset	アドレス	最小値	最大値
0	Modbus アドレス(WORD)	40001	40100
	CPU アドレス	MW100	MW298
20	Modbus アドレス(WORD)	40021	40120
	CPU アドレス	MW100	MW298

Modbus 診断ファンクション

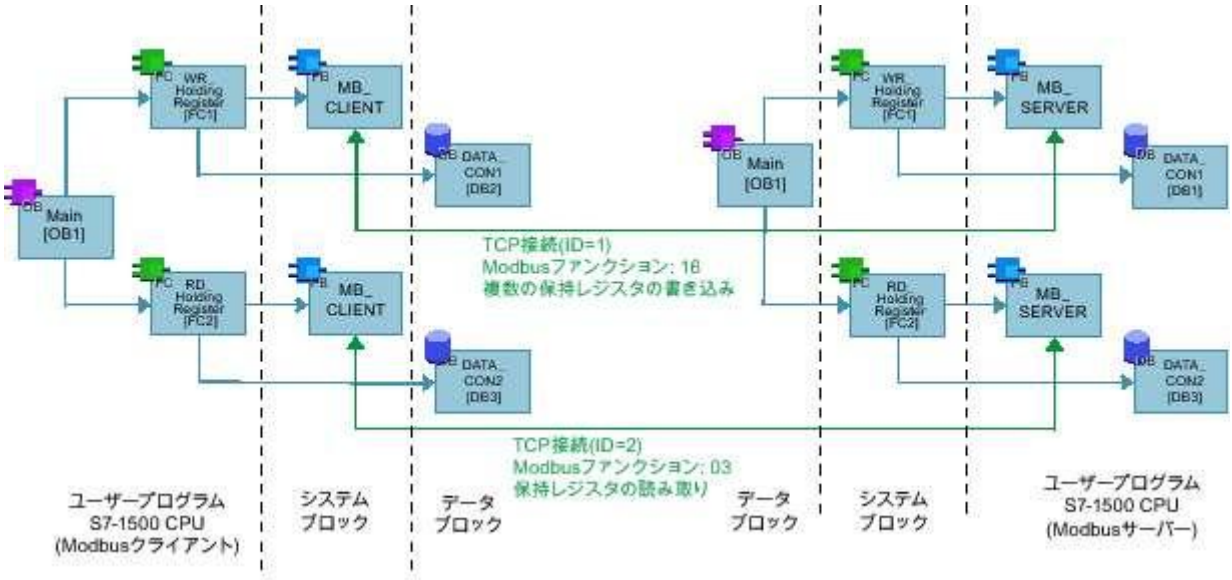
次の表に、Modbus 診断ファンクションの説明を記載しています。

ファンクションコード	診断コード	説明
08	0x0000	エコーテスト:「MB_SERVER」命令は、データワードを受信し、これを変更せずに Modbus クライアントに返します。
08	0x000A	イベントカウンタのリセット:「MB_SERVER」命令は、以下のイベントカウンタをリセットします。 "Success_Count」、「Xmt_Rcv_Count」、「Exception_Count」、「Server_Message_Co」、および「Request_Count」。
11	-	通信イベントカウンタのフェッチ:「MB_SERVER」命令は通信用の内部イベントカウンタを使用し、Modbus サーバーに送信された、正常に実行された読み取りおよび書き込み要求の回数を記録します。 イベントカウンタは、ファンクション 8 または 11 ではインクリメントされません。同じことが、通信エラーの原因になる要求についても言えます。たとえば、プロトコルエラーが発生した場合です(受信した Modbus 要求のファンクションコードがサポートされていないなど)。

例

2 つの S7-1500 CPU 間の Modbus TCP 通信のサンプルプロジェクトは、エントリ ID 94766380 のサービスおよびサポートポータルにあります。

このサンプルでは 2 つの Modbus ファンクションが使用されています。 各 Modbus ファンクションでは、Modbus ブロックペア(MB_CLIENT および MB_SERVER)を使用して Modbus TCP 接続が確立されます。



MB_HOLD_REG パラメータ

説明

MB_HOLD_REG パラメータは、Modbus サーバーから読み取られた、または Modbus サーバーに書き込まれたデータの格納用データバッファに対するポインタです。 グローバルデータブロックまたはビットメモリ(M)をメモリ領域として使用することができます。

- データブロック(D)のアドレス数の上限値は、CPU の最大ワークメモリによって決まります。
- ビットメモリ(M)の数の上限値は、CPU のメモリ領域のサイズによって決まります。

次の表に、Modbus ファンクション 3 (WORD の読み取り)、6 (WORD の書き込み)、16 (複数の WORD の書き込み)、および 23 (複数のワードの読み取りおよび書き込み)用の保持レジスタに対する Modbus アドレスのマッピング例を示します。

Modbus アドレス	MB_HOLD_REG パラメータ - 例		
40001	MW100	DB10.DBW0	"Recipe".ingredient[1]
40002	MW102	DB10.DBW2	"Recipe".ingredient[2]
40003	MW104	DB10.DBW4	"Recipe".ingredient[3]
40004	MW106	DB10.DBW6	"Recipe".ingredient[4]
40005	MW108	DB10.DBW8	"Recipe".ingredient[5]

CONNECT パラメータ

CONNECT パラメータの接続記述子

「MB_SERVER」命令には、2 つの異なる接続記述子を使用できます。

- 構造 TCON_IP_v4 によるプログラミングされた接続

  接続パラメータは TCON_IP_v4 構造体に保存され、接続は命令「MB_SERVER」の呼び出しによってセットアップされます。

- 構造 TCON_Configured による設定した接続

  設定した接続は、既に CPU によって確立されています。 構造 TCON_Configured を使用して、命令に対して使用する既存の接続を指定します。

命令「MB_SERVER」の各インスタンスには、一意の接続が必要です。 命令の各インスタンスに対して個別の構造 TCON_IP_v4 または TCON_Configured を作成して、接続を記述します。

プログラミングされた接続の接続記述子

CONNECT パラメータでプログラミングした接続の場合、接続の記述には TCON_IP_v4.に従って以下の構造を使用します。

- タイプ TCP の接続のみが TCON_IP_v4 構造に指定されていることを確認します。
- この接続は、以下の TCP ポート番号を使用できません。20、21、25、80、102、123、5001、34962、34963、および 34964。

バイト	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceID	HW_ANY	-	ローカルインターフェースのハードウェア識別子(値の範囲:0～65535)。
2 … 3	ID	CONN_OUC	-	この接続への参照(値の範囲:1～4095)。 このパラメータは、CPU 内の接続を一意に識別します。 命令「MB_SERVER」の個々のインスタンスは、固有の ID を使用する必要があります。
4	Connection-Type	BYTE	11	接続タイプ TCP の場合は、11 (10 進数)を選択します。 その他の接続タイプは、許可されていません。 別の接続タイプ(UDP など)が使用された場合、対応するエラーメッセージが命令の STATUS パラメータで出力されます。
5	ActiveEstab-lished	BOOL	FALSE	接続の確立方法の ID パッシブ接続の確立の場合は、FALSE を選択します。
6 … 9	RemoteAd-dress	ARRAY [1..4] of BYTE	0.0.0.0	接続パートナーの IP アドレス、たとえば、192.168.0.1: • addr[1] = 192 • addr[2] = 168 • addr[3] = 0

				• addr[4] = 1 命令「MB_SERVER」が任意の接続パートナーからの接続要求を受け入れる必要がある場合、IP アドレスとして「0.0.0.0」を使用します。
10 … 11	RemotePort	UINT	0	リモート接続パートナーのポート番号(値の範囲: 1～49151)。 命令「MB_SERVER」が任意のリモートパートナーからの接続要求を受け入れる必要がある場合、ポート番号として「0」を使用します。
12 … 13	LocalPort	UINT	502	ローカル接続パートナーのポート番号(値の範囲: 1～49151)。 IP ポートの番号は、Modbus クライアントの接続要求に対してモニタされる IP ポートを定義します。 次の TCP ポート番号は、「MB_SERVER」命令のパッシブ接続で使用してはなりません。20、21、25、80、102、123、5001、34962、34963、および 34964。

注記

命令「MB_SERVER」バージョン 2.1 の移行

パラメータ CONNECT_ID および IP_PORT は、「MB_SERVER」命令のバージョン 3.0 で TCON_IP_v4 構造にマッピングされます。

- 「MB_SERVER」V2.1 命令のパラメータ「CONNECT_ID」は、TCON_IP_v4 のパラメータ ID に相当します。
- 「MB_SERVER」V2.1 命令のパラメータ「IP_PORT」は、TCON_IP_v4 のパラメータ LocalPort に相当します。

設定した接続の接続記述子

CONNECT パラメータで設定した接続の場合、接続の記述には TCON_Configured に従って以下の構造を使用します。

- タイプ TCP の接続のみが TCON_Configured 構造に指定されていることを確認します。
- この接続は、以下の TCP ポート番号を使用できません。20、21、25、80、102、123、5001、34962、34963、および 34964。

バイト	パラメータ	データタイプ	開始値	説明
0 … 1	InterfaceID	HW_ANY	-	ローカルインターフェースのハードウェア識別子(値の範囲:0～65535)。
2 … 3	ID	CONN_OUC	-	この接続への参照(値の範囲:1～4095)。 既存の接続の接続 ID を入力します。
4	Connection-Type	BYTE	-	接続タイプ 設定した接続に対して 254 (10 進数)を選択します。

パラメータステータス

パラメータ STATUS (全般的なステータス情報)

STA-TUS* (W#16#)	説明
0000	命令がエラーなしで実行されました。
0001	接続が確立されました。
0003	接続が終了しました。
7000	動作中の呼び出しはありません(REQ=0)。
7001	最初の呼び出し。 接続の確立がトリガされました。
7002	中間呼び出し。 接続を確立中です。
7003	接続を終了中です。
7005	データを送信中です。
7006	データを受信中です。
80BB	ActiveEstablished パラメータの無効な値(接続確立のタイプの識別子、CONNECT パラメータを参照): • パッシブ接続の確立のみがサーバーに対して許可されています(active_established = FALSE)。 • アクティブ接続の確立のみがクライアントに対して許可されています(active_established = TRUE)。
8380	受信された Modbus フレームの形式が不正であるか、または受信されたバイト数が少なすぎます。
*ステータスコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。	

パラメータ STATUS (パラメータエラー)

STATUS* (W#16#)	「MB_SERVER」からのエラーメッセージのエラーコード (B#16#)	説明
8187	応答なし	MB_HOLD_REG パラメータのポインタが無効です。 データ領域が小さすぎます。
8381	01	ファンクションコードはサポートされていません。
8382	03	データ長のエラー: • 受信した Modbus フレームでの不正な長さ指定 • Modbus フレームのヘッダーに入力されたフレーム長が、実際に受信したバイト数と一致しません。 • Modbus フレームのヘッダーに入力されたバイト数が、実際に受信したバイト数と一致しません(ファンクション 15 および 16)。

8383	02	データアドレスのエラー、または保持レジスタ(MB_HOLD_REG パラメータ)のアドレス領域外へのアクセス。
8384	03	無効なデータ値(ファンクション 5)。
8385	03	診断コードはサポートされていません(ファンクションコード 08 の場合のみ)。
*ステータスコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。		

注記

内部で使用される通信命令のエラーコード

「MB_SERVER」命令を使用すると、表にリストされたエラーに加えて、命令によって使用される通信命令(「TCON」、「TDISCON」、「TSEND」、「TRCV」、「T_DIAG」、および「T_RESET」)が原因のエラーが発生する可能性があります。

エラーコードが、「MB_SERVER」命令のインスタンスデータブロックを介して割り当てられます。 それぞれの命令に対して、エラーコードが"Static"セクションの STATUS に表示されます。

エラーコードの意味は、対応する通信命令の文書で入手できます。

TeleService

この章には下記に関する情報が記載されています：

- TM_MAIL: 電子メールの転送 (S7-1200)

TM_MAIL: 電子メールの転送

この章には下記に関する情報が記載されています：

- TM_MAIL の説明 (S7-1200)
- パラメータ TO_S、CC、および FROM (S7-1200)
- STATUS および SFB_STATUS パラメータ (S7-1200)

TM_MAIL の説明

説明

「TM_MAIL」命令は、非同期で動作します。つまり、複数の呼び出しにわたって実行されます。「TM_MAIL」命令を呼び出すとき、インスタンスを指定する必要があります。 インスタンスには、属性「保持」を設定できません。 この属性は、CPU が STOP から RUN に切り替わった場合にインスタンスが初期化され、その後、新規の電子メール送信ジョブがトリガされるようにします。

電子メールの送信は、REQ パラメータの「0」から「1」へのエッジの切り替えで開始します。ジョブステータスは、出力パラメータ BUSY、「DONE」、「ERROR」、「STATUS」、および「SFC_STATUS」で示されます。 "この場合、「SFC_STATUS」は呼び出された通信ブロックの「STATUS」出力パラメータに対応します。

出力パラメータ DONE、ERROR、STATUS、および SFC_STATUS は、それぞれ BUSY 出力パラメータのステータスが「1」から「0」に切り替わった場合に、1 サイクルのみ表示されます。 次の表に、BUSY、DONE、および ERROR の関係を示します。 この表を使って、「TM_MAIL」命令の現在のステータスと、電子メールの送信がいつ完了するか決定できます。

DONE	BUSY	ER-ROR	説明
0	1	0	ジョブが処理中です。
1	0	0	ジョブが正常に完了しました。
0	0	1	ジョブがエラーありで終了しました。 エラーの原因は、STATUS および SFC_STATUS パラメータで参照できます。
0	0	0	「TM_MAIL」命令に(新規)ジョブが割り当てられていません。

「TM_MAIL」が有効なときに CPU が STOP モードに移行した場合、メールサーバーへの通信接続が中止されます。 メールサーバーへの通信接続は、産業用イーサネットバスで通信上の問題が発生した場合にも失われます。この場合、メールの転送が割り込まれ、受け手に届きません。

通知

ユーザープログラムの変更

使用しているユーザープログラムでは、「TM_MAIL」の呼び出しに直接影響を与える部分のみを変更できます。

- CPU が「STOP」モードの場合、または
- メールが送信されない場合(REQ = 0 および BUSY = 0)。

これは、特に、「TM_MAIL」呼び出しまたは「TM_MAIL」インスタンスの呼び出しを含むプログラムブロックの削除および置き換えに関連します。

この制限を無視すると、接続リソースが占有される場合があります。 産業用イーサネット経由で、オートメーションシステムの TPC/IP 通信ファンクションが未定義ステータスに変更されることがあります。

この切り替えが転送された後、CPU のウォームリスタートまたはコールドリスタートが必要になります。

データの整合性

電子メールの送信がトリガされるたびに、この命令の ADDR_MAIL_SERVER 入力パラメータが再度「TM_MAIL」命令から取得されます。 操作中に変更が加えられた場合、「新しい」値は電子メールが再度トリガされた場合にのみ有効になります。

一方、WATCH_DOG_TIME、TO_S、CC、FROM、SUBJECT、TEXT、ATTACHMENT、および該当する場合は USERNAME および PASSWORD パラメータも実行中に「TM_MAIL」命令から取得されます。つまり、これらはジョブの完了後(BUSY = 0)のみに変更可能です。

TS Adapter IE のパラメータの設定

TS Adapter IE では、TS Adapter IE が使用しているサービスプロバイダのダイヤルアップサーバーへの接続を確立できるように、発信呼び出しのパラメータを指定する必要があります。

接続の確立を「必要に応じて」に設定すると、電子メールを送信する必要がある場合にのみ接続が確立されます。

アナログモデム接続の場合、接続の確立にかかる時間が長くなる場合があります(約 1 分)。WATCH_DOG_TIME パラメータを指定した場合、接続を確立するために必要な時間を考慮してください。

パラメータ

次の表に、「TM_MAIL」命令のパラメータを示します。

パラメータ	宣言	データタイプ	メモリ領域	説明
REQ	Input	BOOL	I、Q、M、D、L または定数	制御パラメータ REQUEST: 信号立ち上がりエッジで電子メールの送信を有効にします。
ID	Input	CONN_OUC (Word)	D、L または定数	確立する接続の参照。 TCON、TDISCON、TSEND、および TRCV 命令の ID パラメータを参照してください。 ここでは、ユーザープログラムでこの命令の追加のインスタンスに使用されていない番号を入力する必要があります。
TO_S	Input	STRING	D	受信元アドレスの入力パラメータ: 最大長 240 文字の STRING (呼び出しの例を参照)。
CC	Input	STRING	D	CC 受信者アドレスの入力パラメータ(オプション): 最大長 240 文字の STRING (呼び出しの例を参照)。 ここに空の文字列が割り当てられている場合、電子メールは CC 受信者には送信されません。
SUBJECT	Input	STRING	D	電子メールの件名の入力パラメータ。 最大長 240 文字の STRING。
TEXT	Input	STRING	D	電子メールのテキストの入力パラメータ(オプション): 最大長 240 文字のデータ文字列の参照。 このパラメータに空の文字列が割り当てられている場合、電子メールはテキストなしで送信されます。

ATTACH-MENT	Input	VAR-IANT	I、Q、M、D、L	電子メール添付ファイルの入力パラメータ (オプション): 最大長 65534 バイトのバイト/ワード/ダブルワードフィールドの参照。 値が割り当てられていない場合、電子メールは添付ファイルなしで送信されます。
DONE	Output	BOOL	I、Q、M、D、L	• DONE = 0: ジョブがまだ開始されていないか、または実行中です。 • DONE = 1: ジョブがエラーなしで実行されました。
BUSY	Output	BOOL	I、Q、M、D、L	• BUSY = 1: 電子メールの送信が未完了です。 • BUSY=0: 「TM_MAIL」の処理が停止されました。
ERROR	Output	BOOL	I、Q、M、D、L	ERROR=1: 処理中にエラーが発生し、STATUS および SFC_STATUS がエラーの種類に関する詳細情報を提供します。
STATUS	Output	WORD	I、Q、M、D、L	出力/ステータスパラメータ STATUS: 「TM_MAIL」命令の戻り値またはエラー情報。
ADDR_MAIL_SERVER	Static*	DWORD	I、Q、M、D、L	メールサーバーの IP アドレス入力パラメータ: たとえば、16 進数フォーマットのデータワードとしての指定: IP アドレス= 192.168.0.200. ADDR_MAIL_SERVER = DW#16#C0A800C8、ただし: • 192 = 16#C0, • 168 =16#A8 • 0 = 16#00 および • 200 = 16#C8.
WATCH_DOG_TIME	Static*	TIME	I、Q、M、D、L	最大時間の入力パラメータ: 「TM_MAIL」命令は、WATCH_DOG_TIME で指定された時間内に接続を確立するはずです。 この時間が経過すると、ブロックがエラーで終了されます。 接続の切断には特定の時間がかかるため、ブロックが終了されエラーが出力されるまでの時間が WATCH_DOG_TIME を超える場合があります。 まず、2 分の設定から始めてください。 ISDN 電話接続を使用する場合、大幅に短い時間を選択できます。
USERNAME	Static*	STRING	D	ユーザー名の入力パラメータ: 最大長 180 文字の STRING。 ユーザー名は認証の必須条件です。
PASSWORD	Static*	STRING	D	パスワードの入力パラメータ: 最大長 180 文字の STRING。 パスワードは認証の必須条件です。
FROM	Static*	STRING	D	送信元アドレスの入力パラメータ:

				最大長 240 文字の STRING (呼び出しの例を参照)。
SFC_STATUS	Static*	WORD	I、Q、M、D、L	出力/ステータスパラメータ「SFC_STATUS」: 呼び出された通信ブロックのエラー情報。
*パラメータの値は、「TM_MAIL」命令の呼び出しごとには変更されません。 値はインスタンスの静的パラメータ内にあり、命令が最初に呼び出されたときに 1 回のみ書き込まれます。				

有効なデータタイプの詳細については、「有効なデータタイプの概要」を参照してください。

注記

オプションのパラメータ

オプションのパラメータ CC、TEXT、および ATTACHMENT は、対応するパラメータが長さ> 0 の文字列を含む場合のみ電子メールと一緒に送信されます。

SMTP 認証

認証は、ID の確認手順、たとえばパスワードの照会などに、これを参照します。

「TM_MAIL」命令は、ほとんどのメールサーバーが要求する SMTP 認証プロシージャ AUTH-LOGIN をサポートします。 使用しているメールサーバーの認証手続きについての情報は、メールサーバーのマニュアルまたはインターネットサービスプロバイダの Web サイトを参照してください。

AUTH-LOGIN 認証プロシージャを使用するには、「TM_MAIL」命令がメールサーバーにログオンするためのユーザー名を要求します。 このユーザー名は、メールサーバーでメールアカウントを設定するために使ったユーザー名に対応します。 これは、USERNAME パラメータで「TM_MAIL」命令が使用できるようになります。

ログオンするために、「TM_MAIL」命令は、関連するパスワードを要求します。 このパスワードは、お使いのメールアカウントを設定したときに指定したパスワードに対応します。 これは、PASSWORD パラメータで「TM_MAIL」命令が使用できるようになります。

ユーザー名およびパスワードが、それぞれ暗号化されていない状態で(BASE64 コード)、メールサーバーに転送されます。

DB でユーザー名が指定されていない場合、AUTH-LOGIN 認証プロシージャは使用されません。 電子メールは、認証なしで送信されます。

パラメータ TO_S、CC、および FROM

説明

TO_S、CC、および FROM パラメータは、たとえば次の内容を持つ文字列です。

- TO: <wenna@mydomain.com>, <ruby@mydomain.com>,
- CC: <admin@mydomain.com>, <judy@mydomain.com>,
- FROM: <admin@mydomain.com>

パラメータの入力時には、次のルールに注意してください。

- 「TO:」、「CC:」、および「FROM:」文字を入力する必要があります。
- それぞれのアドレスの前に空白文字および右開きの括弧「<」を入力する必要があります。
- それぞれのアドレスの後に左開きの括弧「>」を入力する必要があります。
- TO および CC では、それぞれのアドレスの後にコンマを入力する必要があります。
- FROM には 1 通の電子メールアドレスのみを入力でき、このアドレスの後にはコンマを入力できません。

ランタイムおよびメモリ領域の理由で、「TM_MAIL」命令はパラメータ TO_S、CC および FROM の構文チェックを行いません。

STATUS および SFB_STATUS パラメータ

説明

「TM_MAIL」命令の戻り値は、次のように分類されます。

- W#16#0000:「TM_MAIL」が正常に完了しました。
- W#16#7xxx:「TM_MAIL」のステータス
- W#16#8xxx:通信ブロックの内部呼び出し中、またはメールサーバーからエラーが報告されました。

次の表に、内部呼び出しの通信ブロックのエラーコードを除く「TM_MAIL」の戻り値を示します。

戻り値 STATUS* (W#16#...):	戻り値 SFB_STATUS (W#16#...):	説明	注記
0000	-	「TM_MAIL」の処理がエラーなしで完了されました。	「TM_MAIL」がエラーなしで完了しても、送信した電子メールが届いたことを意味しません (以下参照 - ポイント 1 の注)
7001		"TM_MAIL」が有効(BUSY = 1)です。	最初の呼び出し、ジョブが開始されました。
7002	7002	"TM_MAIL」が有効(BUSY = 1)です。	中間呼び出し、ジョブが既に有効です。
8xxx	xxxx	「TM_MAIL」の処理が、内部呼び出しの通信命令のエラーコードで完了しました。	SFB_STATUS パラメータの評価の詳細情報については、通信命令の STATUS パラメータの説明を参照してください。
8010	xxxx	接続の確立中のエラー。	SFB_STATUS パラメータの評価の詳細情報については、「TCON」命令の STATUS パラメータの説明を参照してください。
8011	xxxx	データ送信のエラー。	SFB_STATUS パラメータの評価の詳細情報については、「TSEND」命令の STATUS パラメータの説明を参照してください。
8012	xxxx	データ受信のエラー。	SFB_STATUS の評価の詳細情報については、「TRCV」命令の STATUS パラメータの説明を参照してください。
8013	xxxx	接続の確立中のエラー。	SFB_STATUS の評価の詳細情報については、「TCON」および「TDISCON」命令の STATUS パラメータの説明を参照してください。

8014	-	接続の確立が不可能です。	不正なメールサーバーの IP アドレス(ADDR_MAIL_SERV-ER)、または接続を確立するためには短すぎる時間(WATCH_DOG_TIME)を入力した可能性があります。 また、CPU がネットワークに接続していない、または CPU の設定が不正である可能性もあります。
8016	-	添付ファイルのコピー中のエラー	-
82xx, 84xx, または 85xx	-	エラーメッセージがメールサーバーから発信され、SMTP プロトコルのエラー番号に対応します(「8」を除く)。 次の列に、発生する可能性のあるエラーコードを示します。	注のポイント 2 を参照してください。
8450	-	操作が実行されませんでした: メールボックスが使用不可/到達不可。	後で再度実行してみてください。
8451	-	操作が中止されました: ローカルの処理エラー	後で再度実行してみてください。
8500	-	シンタックスエラー: エラーが認識されませんでした. これには、コマンド文字列が長すぎる場合のエラーも含まれます。 これは、電子メールが LOGIN 認証プロシージャをサポートしない場合にも発生します。	「TM_MAIL」のパラメータをチェックします。 電子メールを認証なしで送信してみてください。 これを行うには、USER-NAME パラメータを空の文字列で置き換えます。
8501	-	シンタックスエラー: パラメータまたは引数が不正	TO_S または CC に不正なアドレスを入力した可能性があります。
8502	-	コマンドが不明または実装されていません。	エントリ、特に FROM パラメータをチェックします。 これが不完全であったり、「@」または「.」が抜けている場合があります。
8535	-	SMTP 認証が不完全。	不正なユーザー名または不正なパスワードを入力した可能性があります。
8550	-	メールサーバーに接続できない、アクセス権がありません。	不正なユーザー名または不正なパスワードを入力した可能性があります。または、メールサーバーが使用している LOGIN をサポートしていません。 また、別のエラーの原因は、TO_S または CC で「@」の後に不正なドメイン名を入力した可能性もあります。
8552	-	操作が中止されました: 割り当てられたメモリサイズを超えました。	後で再度実行してみてください。
8554	-	送信に失敗しました。	後で再度実行してみてください。

*エラーコードはプログラムエディタ内に整数、または 16 進数の値として表示できます。 表示形式の切り替えに関する追加情報は、「関連項目」を参照してください。

注記

ステータスエラー

1. 受信者のアドレスの不正な入力では、「TM_MAIL」命令のステータスエラーは生成されません。 この場合、これらが正しく入力されていても、電子メールの他の受信者への送信は保証されません。
2. SMTP エラーコードおよびその他の SMTP プロトコルのエラーコードの詳細な情報は、インターネットまたはメールサーバーのエラー文書で参照してください。 メールサーバーからの最新のメッセージは、BUFFER1 パラメータで使用しているインスタンス DB でも参照できます。 「データ」以下で、「TM_MAIL」命令で最後に送信したデータを参照できます。

アドオンパッケージ

この章には下記に関する情報が記載されています：