Allen-Bradley Micro800 Serial ドライバー

目次

Allen-Bradley Micro800 Serial ドライバー

ヘルプバージョン 1.035

目次

概要

Allen-Bradley Micro800 Serial ドライバー は Allen-Bradley Micro800 コントローラが HMI、SCADA、Historian、MES、ERP や多数のカスタムアプリケーションを含む OPC クライアントアプリケーションにシリアル接続するための信頼性の高い手段を提供します。

設定

サポートされるデバイス

Micro830
Micro850

注記: 接続は内蔵シリアルポートまたはプラグインシリアルモジュール経由で行われます。

通信プロトコル

Rockwell Automation フラグメントプロトコル (CIP over DF1)。

DH-485 と DH+ のサポート

このドライバーを DH-485 ネットワークに接続するためには Allen Bradley KF3 またはこれと互換性のあるデバイスが必要です。Allen-Bradley Micro800 Serial ドライバー を使用して DH+ 上のデバイスと通信するには次の 4 つのオプションがあります。

- Allen Bradley KF2 またはこれと互換性のあるデバイス。
- 1784-U2DHP USB コンバータ。このコンバータはシステムには新規のシリアルポートとして表示されます。
- DataLink DL インタフェースカード (PCI/ISA/PC104)。これらのカードによって仮想シリアルポートが追加され、シームレスな構成が可能になります。
- DataLink DL4500 イーサネット-DH+ コンバータ。イーサネットカプセル化用にデバイスを設定します。NIC が必要です。

イーサネットカプセル化

このドライバーでは**イーサネットカプセル化**がサポートされているため、ドライバーはターミナルサーバーを使用してイーサネットネットワークに接続されているシリアルデバイスと通信できます。イーサネットカプセル化モードは「チャネルのプロパティ」の**「物理メディア」**から呼び出すことができます。

チャネルとデバイスの構成
サポートされているチャネルの最大数は 256 です。サポートされているデバイスの最大数は 1 チャネルにつき 1024 です。

チャネルのプロパティ - 一般
このサーバーは、複数の通信ドライバーの同時使用をサポートしています。サーバープロジェクトで使用される各プロトコルおよびドライバーをチャネルと呼びます。サーバープロジェクトは、同じ通信ドライバーまたは一意の通信ドライバーを使用する多数のチャネルから成ります。チャネルは、OPC リンクの基本的な構成要素として機能します。このグループは、識別属性や動作モードなどの一般的なチャネルプロパティを指定するときに使用します。

識別

「名前」: このチャネルのユーザー定義の識別情報。各サーバープロジェクトで、それぞれのチャネル名が一意でなければなりません。名前は最大 256 文字ですが、一部のクライアントアプリケーションでは OPC サーバーのタグ空間をブラウズする際の表示ウィンドウが制限されています。チャネル名は OPC ブラウザ情報の一部です。チャネルの作成にはこのプロパティが必要です。
予約済み文字の詳細については、サーバーのヘルプで「チャネル、デバイス、タグ、およびタググループに適切な名前を付ける方法」を参照してください。

「説明」: このチャネルに関するユーザー定義の情報。
「説明」などのこれらのプロパティの多くには、システムタグが関連付けられています。

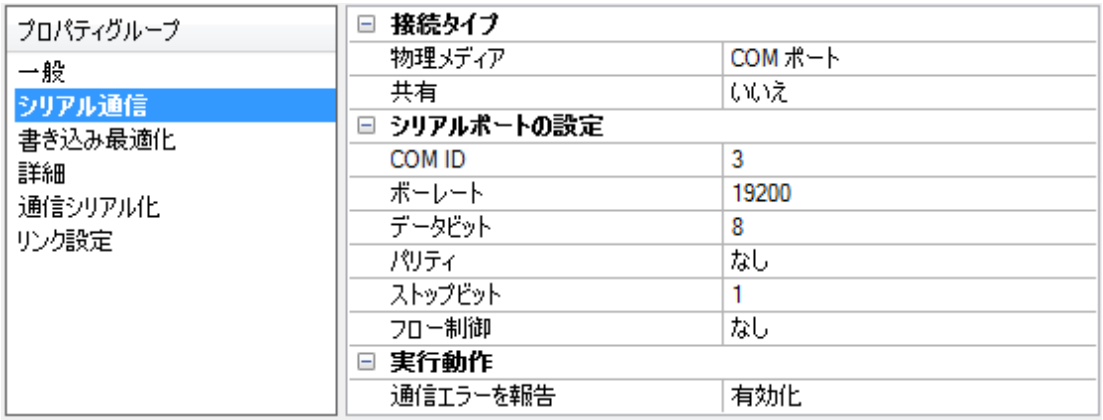
「ドライバー」: このチャネルに選択されているプロトコル/ドライバー。このプロパティでは、チャネル作成時に選択されたデバイスドライバーが示されます。チャネルのプロパティではこの設定を変更することはできません。チャネルの作成にはこのプロパティが必要です。
注記: サーバーがオンラインで常時稼働している場合、これらのプロパティをいつでも変更できます。これには、クライアントがデータをサーバーに登録できないようにチャネル名を変更することも含まれます。チャネル名を変更する前にクライアントがサーバーからアイテムをすでに取得している場合、それらのアイテムは影響を受けません。チャネル名が変更された後で、クライアントアプリケーションがそのアイテムを解放し、古いチャネル名を使用して再び取得しようとしても、そのアイテムは取得されません。このことを念頭において、大規模なクライアントアプリケーションを開発した後はプロパティに対する変更を行わないようにします。サーバー機能へのアクセス権を制限してオペレータがプロパティを変更できないようにするには、ユーザーマネージャを使用します。

診断

「診断取り込み」: このオプションが有効な場合、チャネルの診断情報が OPC アプリケーションに取り込まれ、。サーバーの診断機能は最小限のオーバーヘッド処理を必要とするので、必要なときにだけ利用し、必要がないときには無効にしておくことをお勧めします。デフォルトでは無効になっています。
注記: ドライバーで診断機能がサポートされていない場合、このプロパティは使用できません。
詳細については、サーバーのヘルプの「通信診断」および「統計タグ」を参照してください。

チャネルのプロパティ - シリアル通信
シリアル通信のプロパティはシリアルドライバーで設定でき、選択されているドライバー、接続タイプ、オプションによって異なります。使用可能なプロパティのスーパーセットを以下に示します。
クリックして**接続タイプ**、**シリアルポートの設定**、**イーサネット設定**、**実行動作**のいずれかのセクションにジャンプします。

注記: サーバーがオンラインで常時稼働している場合、これらのプロパティをいつでも変更できます。これらのプロパティに対する変更によって通信が一時的に不通になることがあるので、サーバー機能へのアクセス権を制限するには、ユーザーマネージャを使用します。



接続タイプ

「物理メディア」: データ通信に使用するハードウェアデバイスのタイプを選択します。オプションには「COM ポート」、「なし」、「モデム」、「イーサネットカプセル化」があります。デフォルトは「COM ポート」です。

- **「なし」**: 物理的な接続がないことを示すには「なし」を選択します。これによって通信なしの動作セクションが表示されます。
- **「COM ポート」**: シリアルポートの設定セクションを表示して設定するには、「COM ポート」を選択します。
- **「モデム」**: 通信に電話回線を使用する場合 (モデム設定セクションで設定)、「モデム」を選択します。
- **「イーサネットカプセル化」**: 通信にイーサネットカプセル化を使用する場合に選択します。これによってイーサネット設定セクションが表示されます。
- **「共有」**: 現在の構成を別のチャネルと共有するよう接続が正しく識別されていることを確認します。これは読み取り専用プロパティです。

シリアルポートの設定

「COM ID」: チャネルに割り当てられているデバイスと通信するときに使用する通信 ID を指定します。有効な範囲は 1 から 9991 から 16 です。デフォルトは 1 です。

「ボーレート」: 選択した通信ポートを設定するときに使用するボーレートを指定します。

「データビット」: データワードあたりのデータビット数を指定します。オプションは 5、6、7、8 です。

「パリティ」: データのパリティのタイプを指定します。オプションには「奇数」、「偶数」、「なし」があります。

「ストップビット」: データワードあたりのストップビット数を指定します。オプションは 1 または 2 です。

「フロー制御」: RTS および DTR 制御回線の利用方法を指定します。一部のシリアルデバイスと通信する際にはフロー制御が必要です。以下のオプションがあります。

- **「なし」**: このオプションでは、制御回線はトグル (アサート) されません。
- **「DTR」**: このオプションでは、通信ポートが開いてオンのままになっている場合に DTR 回線がアサートされます。
- **「RTS」**: このオプションでは、バイトを転送可能な場合に RTS 回線がハイになります。バッファ内のすべてのバイトが送信されると、RTS 回線はローになります。これは通常、RS232/RS485 コンバータハードウェアで使用されます。
- **「RTS、DTR」**: このオプションは DTR と RTS を組み合わせたものです。
- **「RTS 常時」**: このオプションでは、通信ポートが開いてオンのままになっている場合に、RTS 回線がアサートされます。
- **「RTS 手動」**: このオプションでは、「RTS 回線制御」で入力したタイミングプロパティに基づいて RTS 回線がアサートされます。これは、ドライバーが手動による RTS 回線制御をサポートしている場合 (またはプロパティが共

有され、このサポートを提供するドライバーに 1 つ以上のチャネルが属している場合) にのみ使用できます。
「RTS 手動」を選択した場合、次のオプションから成る**「RTS 回線制御」**プロパティが追加されます。

- **「事前オン」**: このプロパティでは、データ転送のどれだけ前に RTS 回線を事前にオンにするかを指定します。有効な範囲は 0 から 9999 ミリ秒です。デフォルトは 10 ミリ秒です。
- **「遅延オフ」**: このプロパティでは、データ転送後に RTS 回線をハイのままにする時間を指定します。有効な範囲は 0 から 9999 ミリ秒です。デフォルトは 10 ミリ秒です。
- **「ポーリング遅延」**: このプロパティでは、通信のポーリングが遅延する時間を指定します。有効な範囲は 0 から 9999 です。デフォルトは 10 ミリ秒です。

ヒント: 2 回線 RS 485 を使用している場合、通信回線上で "エコー" が発生することがあります。この通信はエコー除去をサポートしていないので、エコーを無効にするか、RS-485 コンバータを使用することをお勧めします。

実行動作

- **「通信エラーを報告」**: 低レベル通信エラーに関するレポートを有効または無効にします。オンにした場合、低レベルのエラーが発生するとイベントログに書き込まれます。オフにした場合、通常の要求の失敗は書き込まれますが、これと同じエラーは書き込まれません。デフォルトは「有効化」です。
- **「アイドル接続を閉じる」**: チャネル上のクライアントによっていずれのタグも参照されなくなった場合、接続を閉じます。デフォルトは「有効化」です。
- **「クローズするまでのアイドル時間」**: すべてのタグが除去されてから COM ポートを閉じるまでサーバーが待機する時間を指定します。デフォルトは 15 秒です。

イーサネット設定

注記: すべてのシリアルドライバーがイーサネットカプセル化をサポートするわけではありません。このグループが表示されない場合、機能はサポートされていません。

イーサネットカプセル化は、イーサネットネットワーク上のターミナルサーバーに接続しているシリアルデバイスとの通信を可能にします。ターミナルサーバーは基本的には仮想のシリアルポートであり、イーサネットネットワーク上の TCP/IP メッセージをシリアルデータに変換します。メッセージが変換されると、ユーザーはシリアル通信をサポートする標準デバイスをターミナルサーバーに接続可能になります。ターミナルサーバーのシリアルポートが接続先のシリアルデバイスの要件に合うように適切に設定されている必要があります。詳細については、サーバーのヘルプで「イーサネットカプセル化の使用方法」を参照してください。

- **「ネットワークアダプタ」**: このチャネルのイーサネットデバイスがバインドするネットワークアダプタを指定します。バインド先のネットワークアダプタを選択するか、OS がデフォルトを選択可能にします。
  一部のドライバーでは追加のイーサネットカプセル化プロパティが表示されることがあります。詳細については、「チャネルのプロパティ - イーサネットカプセル化」を参照してください。

モデム設定

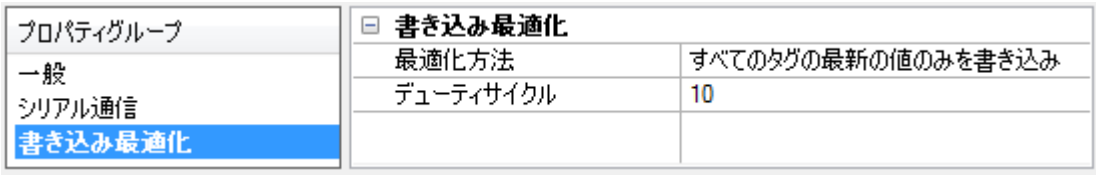
- **「モデム」**: 通信に使用するインストール済みモデムを指定します。
- **「接続タイムアウト」**: 接続が確立される際に待機する時間を指定します。この時間を超えると読み取りまたは書き込みが失敗します。デフォルトは 60 秒です。
- **「モデムのプロパティ」**: モデムハードウェアを設定します。クリックした場合、ベンダー固有のモデムプロパティが開きます。
- **「自動ダイヤル」**: 電話帳内のエントリに自動ダイヤルできます。デフォルトは「無効化」です。詳細については、サーバーのヘルプで「モデム自動ダイヤル」を参照してください。
- **「通信エラーを報告」**: 低レベル通信エラーに関するレポートを有効または無効にします。オンにした場合、低レベルのエラーが発生するとイベントログに書き込まれます。オフにした場合、通常の要求の失敗は書き込まれますが、これと同じエラーは書き込まれません。デフォルトは「有効化」です。
- **「アイドル接続を閉じる」**: チャネル上のクライアントによっていずれのタグも参照されなくなった場合、モデム接続を閉じます。デフォルトは「有効化」です。
- **「クローズするまでのアイドル時間」**: すべてのタグが除去されてからモデム接続を閉じるまでサーバーが待機する時間を指定します。デフォルトは 15 秒です。

通信なしの動作

- **「読み取り処理」**: 明示的なデバイス読み取りが要求された場合の処理を選択します。オプションには「無視」と「失敗」があります。「無視」を選択した場合には何も行われません。「失敗」を選択した場合、失敗したことがクライアントに通知されます。デフォルト設定は「無視」です。

チャネルのプロパティ - 書き込み最適化

サーバーと同様に、デバイスへのデータの書き込みはアプリケーションの最も重要な要素です。サーバーは、クライアントアプリケーションから書き込まれたデータがデバイスに遅延なく届くようにします。このため、サーバーに用意されている最適化プロパティを使用して、特定のニーズを満たしたり、アプリケーションの応答性を高めたりできます。



書き込み最適化

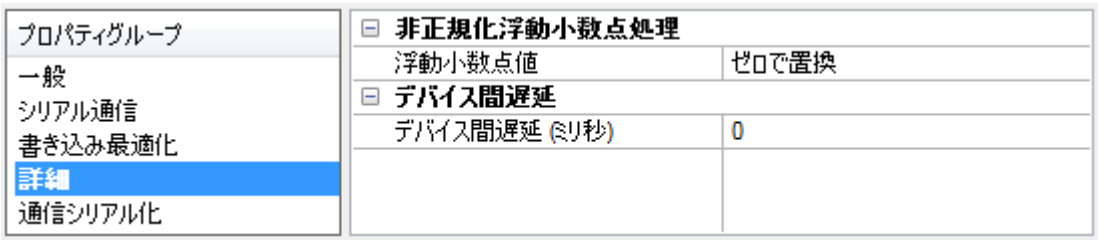
「最適化方法」: 基礎となる通信ドライバーに書き込みデータをどのように渡すかを制御します。以下のオプションがあります。

- **「すべてのタグのすべての値を書き込み」**: このオプションを選択した場合、サーバーはすべての値をコントローラに書き込もうとします。このモードでは、サーバーは書き込み要求を絶えず収集し、サーバーの内部書き込みキューにこれらの要求を追加します。サーバーは書き込みキューを処理し、デバイスにできるだけ早くデータを書き込むことによって、このキューを空にしようとします。このモードでは、クライアントアプリケーションから書き込まれたすべてのデータがターゲットデバイスに送信されます。ターゲットデバイスで書き込み操作の順序または書き込みアイテムのコンテンツが一意に表示される必要がある場合、このモードを選択します。
- **「非 Boolean タグの最新の値のみを書き込み」**: デバイスにデータを実際に送信するのに時間がかかっているために、同じ値への多数の連続書き込みが書き込みキューに累積することがあります。書き込みキューにすでに置かれている書き込み値をサーバーが更新した場合、同じ最終出力値に達するまでに必要な書き込み回数ははるかに少なくなります。このようにして、サーバーのキューに余分な書き込みが累積することがなくなります。ユーザーがスライドスイッチを動かすのをやめると、ほぼ同時にデバイス内の値が正確な値になります。モード名からもわかるように、Boolean 値でない値はサーバーの内部書き込みキュー内で更新され、次の機会にデバイスに送信されます。これによってアプリケーションのパフォーマンスが大幅に向上します。
  注記: このオプションを選択した場合、Boolean 値への書き込みは最適化されません。モーメンタリプッシュボタンなどの Boolean 操作で問題が発生することなく、HMI データの操作を最適化できます。
- **「すべてのタグの最新の値のみを書き込み」**: このオプションを選択した場合、2 つ目の最適化モードの理論がすべてのタグに適用されます。これはアプリケーションが最新の値だけをデバイスに送信する必要がある場合に特に役立ちます。このモードでは、現在書き込みキューに入っているタグを送信する前に更新することによって、すべての書き込みが最適化されます。これがデフォルトのモードです。

「デューティサイクル」: 読み取り操作に対する書き込み操作の比率を制御するときに使用します。この比率は必ず、読み取り 1 回につき書き込みが 1 から 10 回の間であることが基になっています。デューティサイクルはデフォルトで 10 に設定されており、1 回の読み取り操作につき 10 回の書き込みが行われます。アプリケーションが多数の連続書き込みを行っている場合でも、読み取りデータを処理する時間が確実に残っている必要があります。これを設定すると、書き込み操作が 1 回行われるたびに読み取り操作が 1 回行われるようになります。実行する書き込み操作がない場合、読み取りが連続処理されます。これにより、連続書き込みを行うアプリケーションが最適化され、データの送受信フローがよりバランスのとれたものとなります。
注記: 本番環境で使用する前に、強化された書き込み最適化機能との互換性が維持されるようにアプリケーションのプロパティを設定することをお勧めします。

チャネルのプロパティ - 詳細

このグループは、チャネルの詳細プロパティを指定するときに使用します。すべてのドライバーがすべてのプロトコルをサポートしているわけではないので、サポートしていないデバイスには詳細グループが表示されません。



「非正規化浮動小数点処理」: 非正規化値は無限、非数 (NaN)、または非正規化数として定義されます。デフォルトは「ゼロで置換」です。ネイティブの浮動小数点処理が指定されているドライバーはデフォルトで「未修正」になります。「非正規化浮動小数点処理」では、ドライバーによる非正規化 IEEE-754 浮動小数点データの処理方法を指定できます。オプションの説明は次のとおりです。

- **「ゼロで置換」**: このオプションを選択した場合、ドライバーが非正規化 IEEE-754 浮動小数点値をクライアントに転送する前にゼロで置き換えることができます。
- **「未修正」**: このオプションを選択した場合、ドライバーは IEEE-754 非正規化、正規化、非数、および無限の値を変換または変更せずにクライアントに転送できます。

注記: ドライバーが浮動小数点値をサポートしていない場合や、表示されているオプションだけをサポートする場合、このプロパティは使用できません。チャネルの浮動小数点正規化の設定に従って、リアルタイムのドライバータグ (値や配列など) が浮動小数点正規化の対象となります。たとえば、EFM データはこの設定の影響を受けません。

浮動小数点値の詳細については、サーバーのヘルプで「非正規化浮動小数点値を使用する方法」を参照してください。

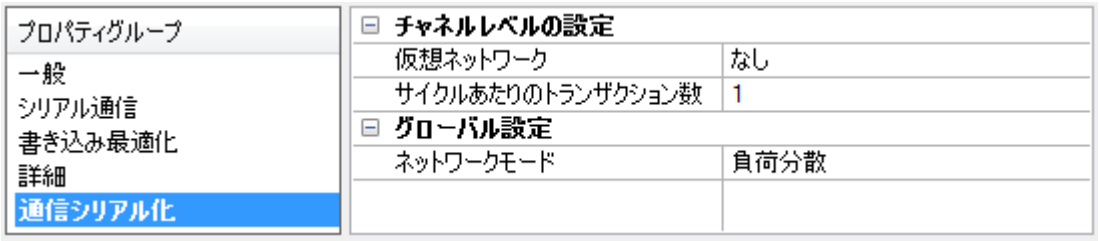
「デバイス間遅延」: 通信チャネルが同じチャネルの現在のデバイスからデータを受信した後、次のデバイスに新しい要求を送信するまで待機する時間を指定します。ゼロ (0) を指定すると遅延は無効になります。

注記: このプロパティは、一部のドライバー、モデル、および依存する設定では使用できません。

チャネルのプロパティ - 通信シリアル化

サーバーのマルチスレッドアーキテクチャにより、チャネルはデバイスとの並列通信が可能になります。これは効率的ですが、物理ネットワークに制約がある (無線イーサネットなど) 場合には通信をシリアル化できます。通信シリアル化によって、仮想ネットワーク内で同時に通信可能なチャネルは 1 つに制限されます。

"仮想ネットワーク" という用語は、通信に同じパイプラインを使用するチャネルと関連デバイスの集合を表します。たとえば、無線イーサネットのパイプラインはマスター無線です。同じマスター無線を使用しているチャネルはすべて同じ仮想ネットワークに関連付けられています。チャネルは "ラウンドロビン" 方式で 1 つずつ順番に通信できます。デフォルトでは、チャネルが 1 つのトランザクションを処理した後で、通信を別のチャネルに渡します。トランザクションには 1 つ以上のタグが含まれることがあります。要求に応答しないデバイスが制御チャネルに含まれている場合、そのトランザクションがタイムアウトになるまでチャネルは制御を解放できません。これによって、仮想ネットワーク内のその他のチャネルでデータ更新の遅延が生じます。



チャネルレベルの設定

「仮想ネットワーク」: このプロパティでは、チャネルの通信シリアル化モードを指定します。オプションには「なし」、「ネットワーク 1」 - 「ネットワーク 500」があります。デフォルトは「なし」です。オプションの説明は次のとおりです。

- **「なし」**: このオプションを選択した場合、チャネルの通信シリアル化は無効になります。
- **「ネットワーク 1」 - 「ネットワーク 500」**: このオプションでは、チャネルを割り当てる仮想ネットワークを指定します。

「サイクルあたりのトランザクション数」: このプロパティでは、チャネルで実行可能な単一ブロック/非ブロック読み取り/書き込みトランザクションの数を指定します。あるチャネルが通信する機会を得ると、この数だけトランザクションが試みられます。有効な範囲は 1 から 99 です。デフォルトは 1 です。

グローバル設定

- **「ネットワークモード」**: このプロパティでは、チャネル通信を委譲する方法を制御します。**「負荷分散」**モードでは、各チャネルが 1 つずつ順番に通信する機会を得ます。**「優先順位」**モードでは、チャネルは次の規則 (最も高い優先順位から最も低い優先順位の順) に従って通信する機会を得ます。
  - 書き込みが保留中になっているチャネルの優先順位が最も高くなります。
  - (内部のプラグインまたは外部のクライアントインタフェースによって) 明示的な読み取りが保留中になっているチャネルは、その読み取りの優先順位に基づいて優先順位が決まります。
  - スキャン読み取りおよびその他の定期的イベント (ドライバー固有)。

  デフォルトは「負荷分散」であり、すべての仮想ネットワークとチャネルに影響します。

非送信請求応答に依存するデバイスを仮想ネットワーク内に配置してはなりません。通信をシリアル化する必要がある場合、「自動格下げ」を有効にすることをお勧めします。

データを読み書きする方法はドライバーによって異なるので (単一ブロック/非ブロックトランザクションなど)、アプリケーションの「サイクルあたりのトランザクション数」プロパティを調整する必要があります。その場合、次の要因について検討します。

- 各チャネルから読み取る必要があるタグの数
- 各チャネルにデータを書き込む頻度
- チャネルが使用しているのはシリアルドライバーかイーサネットドライバーか?
- ドライバーは複数の要求に分けてタグを読み取るか、複数のタグをまとめて読み取るか?
- デバイスのタイミングプロパティ (「要求のタイムアウト」や「連続した *x* 回のタイムアウト後の失敗」など) が仮想ネットワークの通信メディアに最適化されているか?

チャネルのプロパティ - イーサネットカプセル化

イーサネットカプセル化はワイヤレスネットワーク接続 (802.11b ネットワークや CDPD パケットネットワークなど) で使用でき、これは広範なシリアルデバイスをサポートすることも目的として開発されました。ターミナルサーバーデバイスを使用することで、RS-232 または RS-485 デバイスをプラント全体に配置すると同時に、それらのリモートマウントのデバイスに 1 台のローカライズされた PC からアクセスできます。イーサネットカプセル化では、必要に応じて各デバイスに個別のネットワーク IP アドレスを割り当てることもできます。複数のターミナルサーバーを介して、1 台の PC から数百のシリアルデバイスにアクセスできます。1 つのチャネルはローカル PC のシリアルポートを使用するよう定義し、別のチャネルはイーサネットカプセル化を使用するよう定義できます。

注記: これらのプロパティはシリアルドライバーでのみ使用できます。表示されるプロパティは、選択した通信ドライバーおよびサポートされている機能によって異なります。

「ネットワークアダプタ」: このプロパティでは、ネットワークアダプタを指定します。

「デバイスアドレス」: このプロパティでは、このデバイスが接続しているターミナルサーバーの 4 つのフィールドから成る IP アドレスを指定します。IP は *YYY.YYY.YYY.YYY* として指定します。*YYY* は IP アドレスを示します。各 *YYY* バイトが 0 から 255 の範囲でなければなりません。チャネルごとに独自の IP アドレスがあります。

「ポート」: このプロパティでは、リモートターミナルサーバーに接続する際に使用するイーサネットポートを設定します。有効な範囲は 1 から 65535 であり、一部の番号は予約済みです。デフォルトは 2101 です。

「プロトコル」: このプロパティでは、使用されているターミナルサーバーの特性に応じて、TCP/IP または UDP 通信を指定します。デフォルトは TCP/IP です。使用可能なプロトコルの詳細については、ターミナルサーバーのヘルプドキュメントを参照してください。

重要: イーサネットカプセル化モードは実際のシリアル通信ドライバーに対して完全に透過的です。ユーザーは残りのデバイスプロパティを、これらがあたかもローカル PC のシリアルポート上で直接デバイスに接続しているかのように構成する必要があります。

「接続タイムアウト」: このプロパティでは、調整するリモートデバイスのソケット接続を確立するために必要な時間を指定します。多くの場合、デバイスとの接続にかかる時間は、そのデバイスに対する通常の通信要求にかかる時間よりも長くなります。有効な範囲は 1 から 999 秒です。デフォルトは 3 秒です。

注記: サーバーがオンラインで常時稼働している場合、これらのプロパティをいつでも変更できます。サーバー機能へのアクセス権を制限してオペレータがプロパティを変更できないようにするには、ユーザーマネージャを使用します。

チャネルのプロパティ - リンク設定

リンク設定

「ステーション ID」: このプロパティではローカルマシンの一意のネットワーク ID を指定します。これは通信先のデバイス (ラジオモデムを除く) に基づいて設定する必要があります。このフォーマットは 10 進です。デフォルトは 0 です。

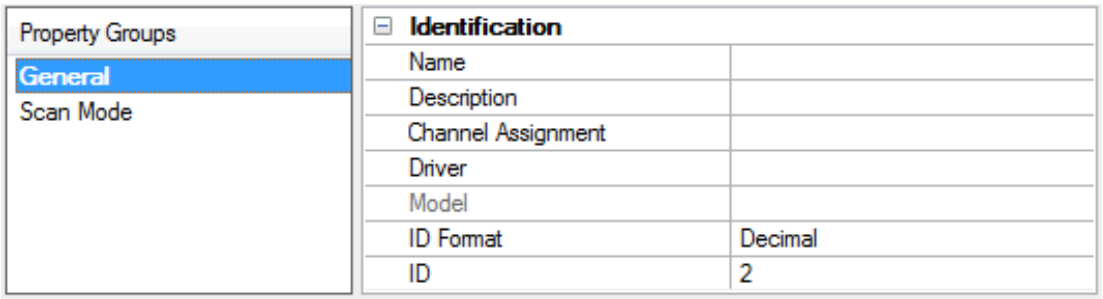
リンクプロトコル: Allen-Bradley Micro800 Serial ドライバーは「全二重」をサポートします。これはポイントツーポイントリンクで使用され、高いパフォーマンスのピアツーピア双方向通信が可能です。

「別のステーションの応答を無視」: このプロパティを有効にした場合、「ステーション ID」に示されているステーション宛の応答の受け付けが制限されます。このプロパティは全二重のみに適用されます。デフォルトでは無効になっています。

注記: 宛先デバイスが DH+ または DH-485 ネットワーク上にある場合、通信はシリアルと -DH+/DH-485 間のコンバータ (つまり KF2/KF3 モジュール) を経由する必要があります。この場合、通信先のデバイスは、宛先デバイス自体ではなくコンバータになります。この構成では、ステーション ID にはコンバータのノードアドレスを設定する必要があります。DH-4850 の範囲は 1 から 63 です。宛先デバイスが DH+ または DH-485 ネットワーク上にない場合、通信先のデバイスは Micro800 となります。この構成でのステーション ID には任意の一意のアドレスを設定できます。範囲は 0 から 255 です。

デバイスのプロパティ - 一般

デバイスは、通信チャネル上の 1 つのターゲットを表します。ドライバーが複数のコントローラをサポートしている場合、ユーザーは各コントローラのデバイス ID を入力する必要があります。



識別

「名前」: このプロパティでは、デバイスの名前を指定します。これは最大 256 文字のユーザー定義の論理名であり、複数のチャネルで使用できます。

注記: わかりやすい名前にすることを一般的にはお勧めしますが、一部の OPC クライアントアプリケーションでは OPC サーバーのタグ空間をブラウズする際の表示ウィンドウが制限されています。デバイス名とチャネル名はブラウズツリー情報の一部にもなります。OPC クライアント内では、チャネル名とデバイス名の組み合わせが "<チャネル名>.<デバイス名>" として表示されます。
詳細については、サーバーのヘルプで「チャネル、デバイス、タグ、およびタググループに適切な名前を付ける方法」を参照してください。

「説明」: このデバイスに関するユーザー定義の情報。
「説明」などのこれらのプロパティの多くには、システムタグが関連付けられています。

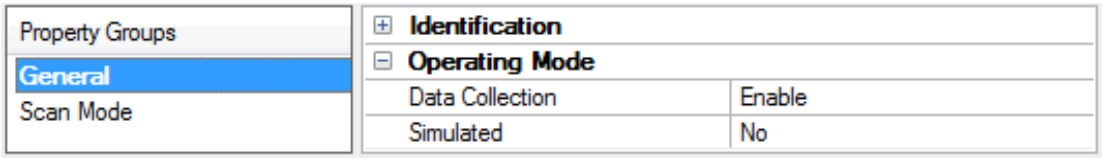
「チャネル割り当て」: このデバイスが現在属しているチャネルのユーザー定義の名前。

「ドライバー」: このデバイスに設定されているプロトコルドライバー。

「モデル」: このプロパティでは、この ID に関連付けられるデバイスのタイプを指定します。このドロップダウンメニューの内容は、使用されている通信ドライバーのタイプによって異なります。ドライバーによってサポートされていないモデルは無効になります。通信ドライバーが複数のデバイスモデルをサポートしている場合、デバイスにクライアントアプリケーションが 1 つも接続していない場合にのみモデル選択を変更できます。
注記: 通信ドライバーが複数のモデルをサポートしている場合、ユーザーは物理デバイスに合わせてモデルを選択する必要があります。このドロップダウンメニューにデバイスが表示されない場合、ターゲットデバイスに最も近いモデルを選択します。一部のドライバーは "オープン" と呼ばれるモデル選択をサポートしており、ユーザーはターゲットデバイスの詳細を知らなくても通信できます。詳細については、ドライバーのヘルプドキュメントを参照してください。

「ID」: このプロパティでは、デバイスのドライバー固有のステーションまたはノードを指定します。入力する ID のタイプは、使用されている通信ドライバーによって異なります。多くの通信ドライバーでは、ID は数値です。数値 ID をサポートするドライバーでは、ユーザーは数値を入力でき、そのフォーマットはアプリケーションのニーズまたは選択した通信ドライバーの特性に合わせて変更できます。フォーマットはデフォルトではドライバーによって設定されます。オプションには「10 進数」、「8 進数」、「16 進数」があります。
注記: ドライバーがイーサネットベースであるか、通常とは異なるステーションまたはノード名をサポートしている場合、デバイスの TCP/IP アドレスをデバイス ID として使用できます。TCP/IP アドレスはピリオドで区切った 4 つの値から成り、各値の範囲は 0 から 255 です。一部のデバイス ID は文字列ベースです。ドライバーによっては、ID フィールドで追加のプロパティを設定する必要があります。詳細については、ドライバーのヘルプドキュメントを参照してください。

動作モード



「データコレクション」: このプロパティでは、デバイスのアクティブな状態を制御します。デバイスの通信はデフォルトで有効になっていますが、このプロパティを使用して物理デバイスを無効にできます。デバイスが無効になっている場合、通信は試みられません。クライアントから見た場合、そのデータは無効としてマークされ、書き込み操作は許可されません。このプロパティは、このプロパティまたはデバイスのシステムタグを使用していつでも変更できます。

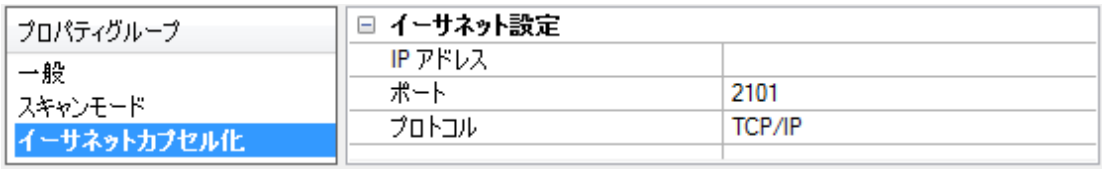
「シミュレーション」: このオプションは、デバイスをシミュレーションモードにします。このモードでは、ドライバーは物理デバイスとの通信を試みませんが、サーバーは引き続き有効な OPC データを返します。シミュレーションモードではデバイスとの物理的な通信は停止しますが、OPC データは有効なデータとして OPC クライアントに返されます。シミュレーションモードでは、サーバーはすべてのデバイスデータを自己反映的データとして扱います。つまり、シミュレーションモードのデバイスに書き込まれたデータはすべて再び読み取られ、各 OPC アイテムは個別に処理されます。アイテムのメモリマップはグループ更新レートに基づきます。(サーバーが再初期化された場合などに) サーバーがアイテムを除去した場合、そのデータは保存されません。デフォルトは「いいえ」です。
注記:

1. システムタグ (_Simulated) は読み取り専用であり、ランタイム保護のため、書き込みは禁止されています。このシステムタグを使用することで、このプロパティをクライアントからモニターできます。
2. シミュレーションモードでは、アイテムのメモリマップはクライアントの更新レート (OPC クライアントではグループ更新レート、ネイティブおよび DDE インタフェースではスキャン速度) に基づきます。つまり、異なる更新レートで同じアイテムを参照する 2 つのクライアントは異なるデータを返します。

シミュレーションモードはテストとシミュレーションのみを目的としています。本番環境では決して使用しないでください。

デバイスのプロパティ - イーサネットカプセル化

イーサネットカプセル化は、イーサネットネットワーク上のターミナルサーバーに接続しているシリアルデバイスとの通信用に設計されています。ターミナルサーバーは基本的には仮想のシリアルポートです。ターミナルサーバーはイーサネットネットワーク上の TCP/IP メッセージをシリアルデータに変換します。メッセージがシリアル形式に変換されると、ユーザーはシリアル通信をサポートする標準デバイスをターミナルサーバーに接続可能になります。
詳細については、サーバーのヘルプで「イーサネットカプセル化の使用方法」を参照してください。
イーサネットカプセル化はドライバーに対して透過的なので、残りのプロパティを、これらがあたかもローカルシリアルポート上で直接デバイスに接続しているかのように設定します。



「IP アドレス」: このプロパティには、デバイスが接続しているターミナルサーバーの 4 つのフィールドから成る IP アドレスを入力します。IP は YYY.YYY.YYY.YYY として指定します。YYY は IP アドレスを示します。各 YYY バイトが 0 から 255 の範囲でなければなりません。各シリアルデバイスは独自の IP アドレスを持つことができますが、単一のターミナルサーバーからマルチドロップされた複数のデバイスがある場合、複数のデバイスが同じ IP アドレスを持つことがあります。

「ポート」: このプロパティでは、リモートターミナルサーバーに接続する際に使用するイーサネットポートを設定します。

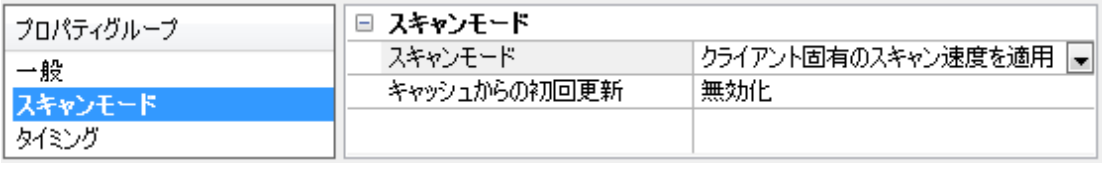
「プロトコル」: このプロパティでは、TCP/IP 通信または UDP 通信を選択します。この選択は使用されているターミナルサーバーの特性によります。デフォルトのプロトコル選択は TCP/IP です。使用可能なプロトコルの詳細については、ターミナルサーバーのヘルプドキュメントを参照してください。

注記

1. サーバーがオンラインで常時稼働している場合、これらのプロパティをいつでも変更できます。サーバー機能へのアクセス権を制限してオペレータがプロパティを変更できないようにするには、ユーザーマネージャを使用します。
2. IP アドレスの有効な範囲は 0.0.0.0 から 255.255.255.255 です (0.0.0.0 と 255.255.255.255 は含まれません)。

デバイスのプロパティ - スキャンモード

「スキャンモード」では、デバイスとの通信を必要とする、サブスクリプション済みクライアントが要求したタグのスキャン速度を指定します。同期および非同期デバイスの読み取りと書き込みは可能なかぎりただちに処理され、「スキャンモード」のプロパティの影響を受けません。



「スキャンモード」: 購読しているクライアントに送信される更新についてデバイス内のタグをどのようにスキャンするかを指定します。オプションの説明は次のとおりです。

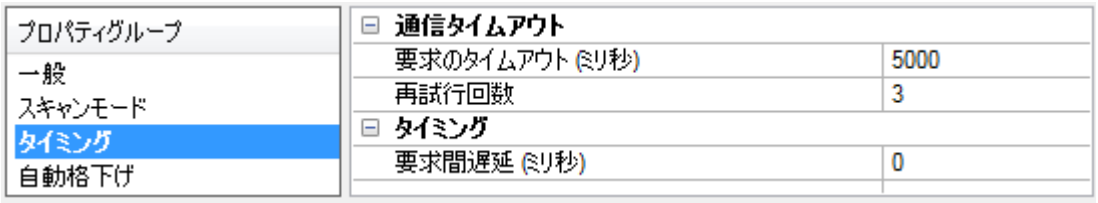
- **「クライアント固有のスキャン速度を適用」**: このモードでは、クライアントによって要求されたスキャン速度を使用します。

- **「指定したスキャン速度以下でデータを要求」**: このモードでは、最大スキャン速度として設定されている値を指定します。有効な範囲は 10 から 99999990 ミリ秒です。デフォルトは 1000 ミリ秒です。
  注記: サーバーにアクティブなクライアントがあり、デバイスのアイテム数とスキャン速度の値が増加している場合、変更はただちに有効になります。スキャン速度の値が減少している場合、すべてのクライアントアプリケーションが切断されるまで変更は有効になりません。
- **「すべてのデータを指定したスキャン速度で要求」**: このモードでは、指定した速度で購読済みクライアント用にタグがスキャンされます。有効な範囲は 10 から 99999990 ミリ秒です。デフォルトは 1000 ミリ秒です。
- **「スキャンしない、要求ポールのみ」**: このモードでは、デバイスに属するタグは定期的にポーリングされず、アクティブになった後はアイテムの初期値の読み取りは実行されません。更新のポーリングは、_DemandPoll タグに書き込むか、個々のアイテムについて明示的なデバイス読み取りを実行することによって、クライアントが行います。詳細については、サーバーのヘルプで「デバイス要求ポール」を参照してください。
- **「タグに指定のスキャン速度を適用」**: このモードでは、静的構成のタグプロパティで指定されている速度で静的タグがスキャンされます。動的タグはクライアントが指定したスキャン速度でスキャンされます。

「キャッシュからの初回更新」: このオプションを有効にした場合、サーバーは保存 (キャッシュ) されているデータから、新たにアクティブ化されたタグ参照の初回更新を行います。キャッシュからの更新は、新しいアイテム参照が同じアドレス、スキャン速度、データ型、クライアントアクセス、スケール設定のプロパティを共有している場合にのみ実行できます。1 つ目のクライアント参照についてのみ、初回更新にデバイス読み取りが使用されます。デフォルトでは無効になっており、クライアントがタグ参照をアクティブ化したときにはいつでも、サーバーがデバイスから初期値の読み取りを試みます。

デバイスのプロパティ - タイミング

デバイスのタイミングのプロパティでは、エラー状態に対するデバイスの応答をアプリケーションのニーズに合わせて調整できます。多くの場合、最適なパフォーマンスを得るためにはこれらのプロパティを変更する必要があります。電気的に発生するノイズ、モデムの遅延、物理的な接続不良などの要因が、通信ドライバーで発生するエラーやタイムアウトの数に影響します。タイミングのプロパティは、設定されているデバイスごとに異なります。



通信タイムアウト

「接続タイムアウト」: このプロパティ (イーサネットベースのドライバーで主に使用) は、リモートデバイスとのソケット接続を確立するために必要な時間を制御します。デバイスの接続時間は、同じデバイスへの通常の通信要求よりも長くかかることがよくあります。有効な範囲は 1 から 30 秒です。デフォルトは通常は 3 秒ですが、各ドライバーの特性によって異なる場合があります。この設定がドライバーでサポートされていない場合、無効になります。
注記: UDP 接続の特性により、UDP を介して通信する場合には接続タイムアウトの設定は適用されません。

「要求のタイムアウト」: このプロパティでは、ターゲットデバイスからの応答を待つのをいつやめるかを判断する際にすべてのドライバーが使用する間隔を指定します。有効な範囲は 50 から 9,999,999 ミリ秒 (167.6667 分) です。デフォルトは通常は 1000 ミリ秒ですが、ドライバーによって異なる場合があります。ほとんどのシリアルドライバーのデフォルトのタイムアウトは 9600 ボー以上のボーレートに基づきます。低いボーレートでドライバーを使用している場合、データの取得に必要な時間が増えることを補うため、タイムアウト時間を増やします。

「タイムアウト前の試行回数」: このプロパティでは、ドライバーが通信要求を発行する回数を指定します。この回数を超えると、要求が失敗してデバイスがエラー状態にあると見なされます。有効な範囲は 1 から 10 です。デフォルトは通常は 3 ですが、各ドライバーの特性によって異なる場合があります。アプリケーションに設定される試行回数は、通信環境に大きく依存します。このプロパティは、接続の試行と要求の試行の両方に適用されます。

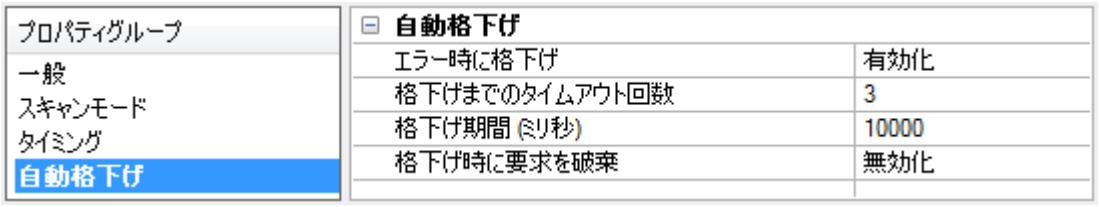
タイミング

「要求間遅延」: このプロパティでは、ドライバーがターゲットデバイスに次の要求を送信するまでの待ち時間を指定します。デバイスに関連付けられているタグおよび 1 回の読み取りと書き込みの標準のポーリング間隔がこれによってオーバーライドされます。この遅延は、応答時間が長いデバイスを扱う際や、ネットワークの負荷が問題である場合に役立ちます。デバイスの遅延を設定すると、そのチャネル上のその他すべてのデバイスとの通信に影響が生じます。可能な場合、要求間遅延を必要とするデバイスは別々のチャネルに分けて配置することをお勧めします。その他の通信プロパティ (通信シリアル化など) によってこの遅延が延長されることがあります。有効な範囲は 0 から 300,000 ミリ秒ですが、一部のド

ライバーでは独自の設計の目的を果たすために最大値が制限されている場合があります。デフォルトは 0 であり、ターゲットデバイスへの要求間に遅延はありません。
注記: すべてのドライバーで「要求間遅延」がサポートされているわけではありません。使用できない場合にはこの設定は表示されません。

デバイスのプロパティ - 自動格下げ

自動格下げのプロパティを使用することで、デバイスが応答していない場合にそのデバイスを一時的にスキャン停止にできます。応答していないデバイスを一定期間オフラインにすることで、ドライバーは同じチャネル上のほかのデバイスとの通信を引き続き最適化できます。停止期間が経過すると、ドライバーは応答していないデバイスとの通信を再試行します。デバイスが応答した場合はスキャンが開始され、応答しない場合はスキャン停止期間が再開します。



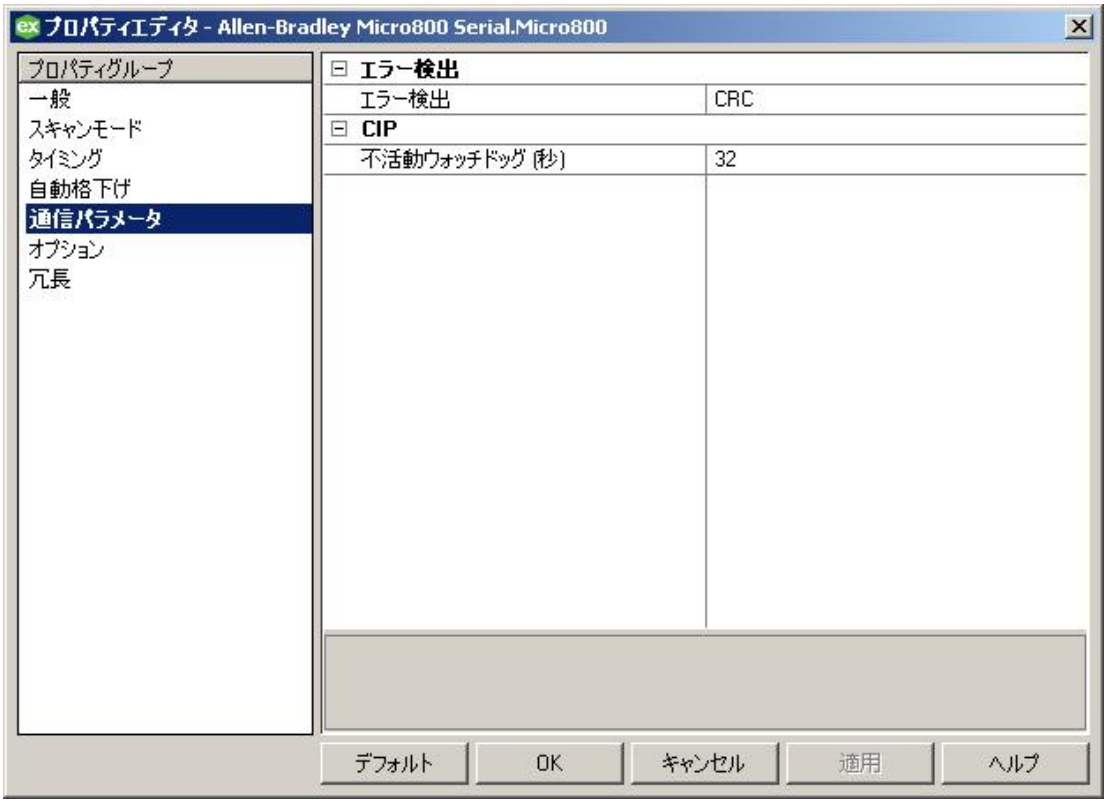
「エラー時に格下げ」: 有効にした場合、デバイスは再び応答するまで自動的にスキャン停止になります。
ヒント: システムタグ _AutoDemoted を使用して格下げ状態をモニターすることで、デバイスがいつスキャン停止になったかを把握できます。

「格下げまでのタイムアウト回数」: デバイスをスキャン停止にするまでに要求のタイムアウトと再試行のサイクルを何回繰り返すかを指定します。有効な範囲は 1 から 30 回の連続エラーです。デフォルトは 3 です。

「格下げ期間」: タイムアウト値に達したときにデバイスをスキャン停止にする期間を指定します。この期間中、そのデバイスには読み取り要求が送信されず、その読み取り要求に関連するすべてのデータの品質は不良に設定されます。この期間が経過すると、ドライバーはそのデバイスのスキャンを開始し、通信での再試行が可能になります。有効な範囲は 100 から 3600000 ミリ秒です。デフォルトは 10000 ミリ秒です。

「格下げ時に要求を破棄」: スキャン停止期間中に書き込み要求を試行するかどうかを選択します。格下げ期間中も書き込み要求を必ず送信するには、無効にします。書き込みを破棄するには有効にします。サーバーはクライアントから受信した書き込み要求をすべて自動的に破棄し、イベントログにメッセージを書き込みません。

デバイスのプロパティ - 通信パラメータ



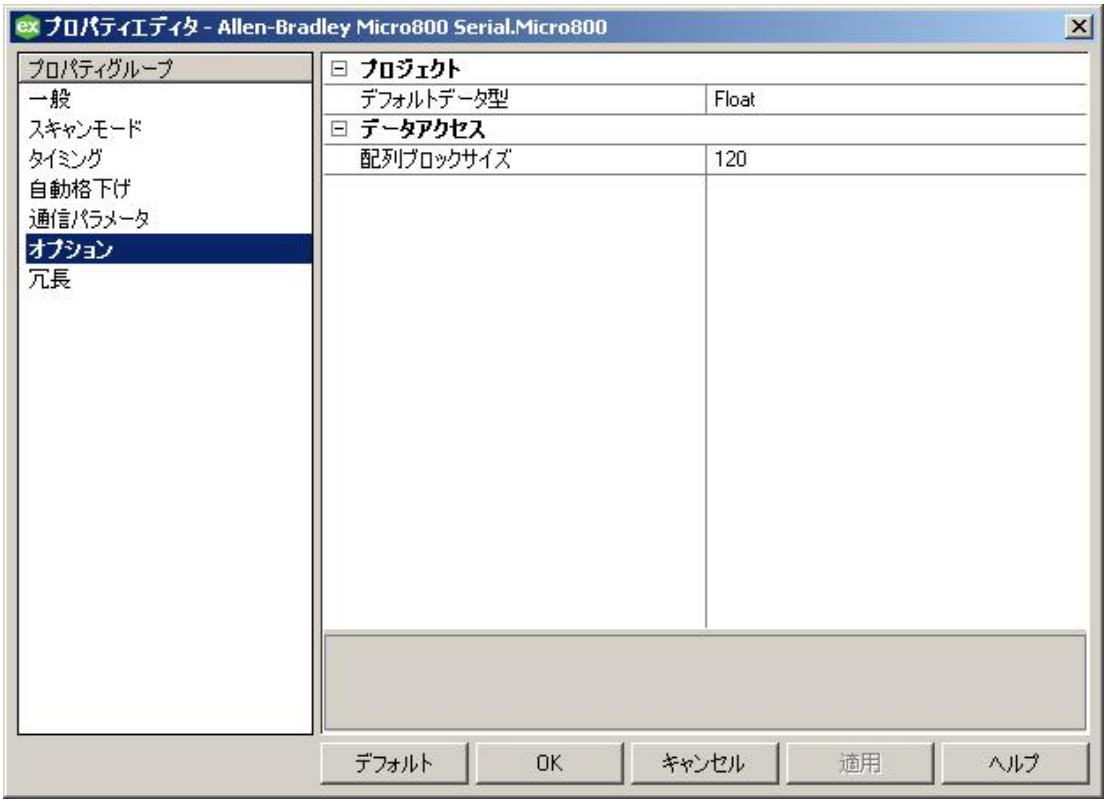
「エラー検出」

「エラー検出」: デバイスで定められているエラー検出/チェックサム方法をブロックチェック文字 (BCC) または巡回冗長検査 (CRC) から選択します。デフォルトは「CRC」です。

「CIP」

「不活動ウォッチドッグ」: 接続が (読み取り/書き込み可能なトランザクションがない) アイドル状態となりうる時間 (秒数) を指定します。この時間を経過するとコントローラによって接続はクローズします。一般的に、この値が大きいほど、接続のリソースがコントローラによって解放されるまでの時間が長くなります (値が小さいほど時間が短くなります)。デフォルトは 32 秒です。

デバイスのプロパティ - オプション



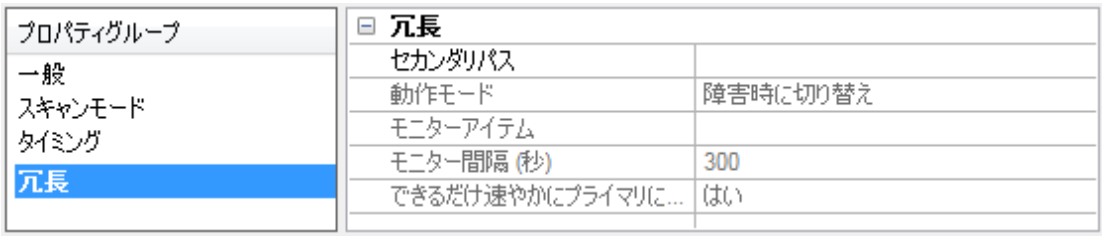
プロジェクト

「デフォルトデータ型」: タグの追加/修正/インポート時にデフォルトのデータ型を選択した場合にクライアント/サーバータグに割り当てられるデータ型を選択します。デフォルトは「Float」です。クライアントでネイティブのデータ型が割り当てられた動的タグが作成された場合や、サーバーでデフォルトのデータ型が割り当てられた静的タグが作成された場合、タグにはデフォルトデータ型が割り当てられます。

「データアクセス」

「配列ブロックサイズ」: 1 回のトランザクションで読み取られるアトミック配列要素の最大数を指定します。範囲は 30 から 3840 要素です。デフォルトは 120 要素です。
Boolean 配列では、1 要素は 32 要素のビット配列と見なされます。ブロックサイズを 30 要素に設定すると 960 ビット要素として解釈され、3840 要素は 122880 ビット要素として解釈されます。

デバイスのプロパティ - 冗長



冗長設定はメディアレベルの冗長プラグインで使用できます。
詳細については、Web サイトまたはユーザーマニュアルを参照するか、営業担当者までお問い合わせください。

パフォーマンスの最適化

最大のパフォーマンスを得るために Allen-Bradley Micro800 Serial ドライバーに適用可能ないくつかのガイドラインがあります。通信レベルとアプリケーションレベルでの最適化の詳細については、以下のリストからリンクを選択してください。

通信の最適化

どのようなプログラマブルコントローラにも、パフォーマンス全体とシステム通信を向上させるさまざまな手段が備わっています。

ネイティブタグ名を短くする

ネイティブタグは通信要求でそのシンボリック名を指定することでデバイスとの間で読み書きを行います。このため、タグ名が長いほど、要求は大きくなります。

配列要素のブロック化

アトミック配列要素の読み取りを最適化するには、配列を個別に読み取るのではなく1回の要求で配列のブロックを読み取ります。1つのブロックで読み取る要素の数が多いほど、パフォーマンスが向上します。ほとんどの時間はトランザクションのオーバーヘッドと処理に割かれるので、できるだけ少ないトランザクションでできるだけ多くのタグをスキャンするようにします。これが配列要素ブロック化の要点です。

ブロックサイズは要素数として指定します。ブロックサイズを120要素として指定した場合、1回の要求で最大120個の配列要素が読み取られます。ブロックの最大サイズは3840要素です。Boolean配列は処理が異なり、プロトコルではBoolean配列は32ビット配列です。したがって、要素0を要求するということは、ビット0から31を要求することになります。説明が一貫したものになるように、Boolean配列要素は1ビットと見なされます。つまり、(ブロックサイズを3840とした場合の)要求可能な配列要素の最大数は122880 BOOL、3840 SINT、3840 INT、3840 DINT、3840 LINT、および3840 REALです。

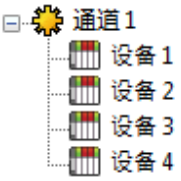
ブロックサイズは調整可能であり、使用中のプロジェクトに基づいて選択する必要があります。たとえば、配列要素0-26と要素3839が読み取り対象のタグである場合、ブロックサイズとして3840を使用するのは大きすぎであるだけでなく、ドライバーのパフォーマンスが損なわれます。これは、0から3839の要素のうち、重要であるのはその28個だけであるにもかかわらず、0から3839のすべての要素が要求のたびに読み取られるためです。この場合、ブロックサイズを30にするのが妥当です。要素0-26は1回の要求で読み取られ、要素3839は次の要求で読み取られます。

関連項目: *オプション*

アプリケーションの最適化

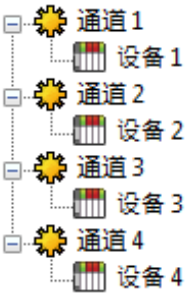
Allen-Bradley Micro800 Serialドライバーは、システム全体のパフォーマンスへの影響を最小限に抑えながら最大のパフォーマンスが得られるように設計されています。このドライバーは高速ですが、最大のパフォーマンスを得るために参考となるいくつかのガイドラインがあります。

このサーバーでは、Allen-Bradley Micro800 Serialなどの通信プロトコルのことをチャネルと呼びます。アプリケーションで定義されている各チャネルは、サーバーでの個々の実行パスを表します。チャネルが定義された後、そのチャネルの下に一連のデバイスを定義する必要があります。これらのデバイスそれぞれが、データの収集元となる単一のMicro800 CPUを表します。このアプローチに従ってアプリケーションを定義することで高いパフォーマンスが得られますが、Allen-Bradley Micro800 Serialドライバーやネットワークがフルに利用されるわけではありません。単一のチャネルを使用して構成されているアプリケーションの表示例を次に示します。



デバイスそれぞれが"Micro800"という名前の単一のチャネルの下に表示されます。この構成では、ドライバーは効果的な速度で情報を収集するために、できるだけ速やかにあるデバイスから次のデバイスに移動する必要があります。さらにデバイスが追加されたり、1つのデバイスからより多くの情報が要求されたりするにしたがい、全体的な更新レートが低下していきます。

Allen-Bradley Micro800 Serialドライバーがチャネルを1つだけ定義可能な場合、上に示した例が唯一可能なオプションとなりますが、このドライバーは最大256チャネルまで定義できます。複数のチャネルを使用して複数の要求をネットワークに同時に発行することで、データ収集のワークロードが分散されます。パフォーマンスを改善するために同じアプリケーションを複数のチャネルを使用して構成した場合の例を次に示します。



ここではそれぞれのデバイスが各自のチャネルの下に定義されています。この新しい構成では、各デバイスからのデータ収集タスクごとに 1 つの実行パスが割り当てられます。アプリケーションのデバイスの数が 256 以下である場合、まさにここで示したように最適化できます。

アプリケーションのデバイスの数が 256 を超える場合でもパフォーマンスは改善されます。デバイスの数は 256 以下であるのが理想的ですが、アプリケーションは追加のチャネルから恩恵を受けます。デバイスの負荷をすべてのチャネルに分散してもサーバーはデバイスを切り替えますが、単一のチャネルで処理するデバイスの数ははるかに少なくなります。

データ型の説明

データ型	説明
Boolean	1 ビット
Byte	符号なし 8 ビット値
Char	符号付き 8 ビット値
Word	符号なし 16 ビット値
Short	符号付き 16 ビット値
DWord	符号なし 32 ビット値
Long	符号付き 32 ビット値
BCD	2 バイトパックされた BCD、4 桁の 10 進数
LBCD	4 バイトパックされた BCD、8 桁の 10 進数
Float	32 ビット IEEE 浮動小数点
Double	64 ビット IEEE 浮動小数点
Date	64 ビットの日付/時刻
String	Null 終端文字配列

アドレスの説明

Micro800 では、シンボルベースのアドレス指定構造体であるタグが使用され、これはネイティブタグと呼ばれます。これらのタグは従来の PLC データアイテムとは異なり、ファイル番号やレジスタ番号ではなくタグ名自体がアドレスになります。

Allen-Bradley Micro800 Serial ドライバーでは、コントローラのアトミックデータ型 BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT、DWORD、LINT、ULINT、LWORD、REAL、LREAL、および SHORT_STRING にアクセスできます。定義済みの一部のデータ型は構造体ですが、これらは最終的にはそのアトミックデータ型に基づきます。このため、構造体のすべての非構造体 (アトミック) メンバーにアクセスできます。たとえば、TIMER はサーバータグに割り当てることができませんが、TIMER のアトミックメンバー (TIMER.EN、TIMER.ACC など) はタグに割り当てることができます。構造体メンバーが構造体自体である場合、サブ構造体のアトミックメンバーにアクセスするには両方の構造体を展開する必要があります。これはユーザー定義とモジュール定義の型でより一般的であり、定義済みの型では必要はありません。

アトミックデータ型	説明	クライアントタイプ	範囲
BOOL	1 ビット値	VT_BOOL	0, 1
SINT	符号付き 8 ビット値	VT_U1	-128 から 127
USINT	符号なし 8 ビット値	VT_UI1	0 から 255
BYTE	ビット文字列 (8 ビット)	VT_UI1	0 から 255
INT	符号付き 16 ビット値	VT_I2	-32,768 から 32,767
UINT	符号なし 16 ビット値	VT_UI2	0 から 65535
WORD	ビット文字列 (16 ビット)	VT_UI2	0 から 65535
DINT	符号付き 32 ビット値	VT_I4	-2,147,483,648 から 2,147,483,647

アトミックデータ型	説明	クライアントタイプ	範囲
UDINT	符号なし 32 ビット値	VR_UI4	0 から 4294967296
DWORD	ビット文字列 (32 ビット)	VR_UI4	0 から 4294967296
LINT	符号付き 64 ビット値	VT_R8	-1.798E+308 から -2.225E-308、0、2.225E-308 から 1.798E+308
ULINT	符号なし 64 ビット値	VT_R8	-1.798E+308 から -2.225E-308、0、2.225E-308 から 1.798E+308
LWORD	ビット文字列 (64 ビット)	VT_R8	-1.798E+308 から -2.225E-308、0、2.225E-308 から 1.798E+308
REAL	32 ビット IEEE 浮動小数点 POINT	VT_R4	1.1755 E-38 から 3.403E38、0、-3.403E-38 から -1.1755
LREAL	64 ビット IEEE 浮動小数点	VT_R8	-1.798E+308 から -2.225E-308、0、2.225E-308 から 1.798E+308
SHORT_STRING	文字列。使用可能な最大文字数は 80 です。	VT_BSTR	

関連項目: *高度な使用事例*

クライアント/サーバータグアドレスの規則
ネイティブタグ名はクライアント/サーバータグのアドレスに相当します。ネイティブタグ名 (Connected Components Workbench から入力) とクライアント/サーバータグアドレスの両方が IEC 1131-3 の識別子の規則に従います。規則の説明は次のとおりです。

- 先頭は英字またはアンダースコアでなければなりません
- 英数文字とアンダースコアのみを含むことができます
- 最大 40 文字まで使用できます
- アンダースコアが連続していてはなりません
- 大文字と小文字は区別されません

最適なパフォーマンスを得るためには、ネイティブタグ名のサイズを最小限に抑えてください。名前が小さいほど、1 つのトランザクションに含めることができる要求の数が増えます。

クライアント/サーバータグ名の規則
サーバーでのタグ名の割り当てはアドレスの割り当てとは異なり、名前の先頭がアンダースコアであってはなりません。

関連項目: *パフォーマンスの最適化*

アドレスのフォーマット

ネイティブタグはサーバーで静的にアドレス指定することも、いくつかの方法によってクライアントから動的にアドレス指定することもできます。タグのフォーマットはそのタイプと使用目的によって異なります。たとえば、SINT 型タグ内のビットにアクセスする場合にはビットフォーマットを使用します。アドレスのフォーマットと構文については、以下の表を参照してください。

注記: 配列フォーマット以外のフォーマットは Connected Components Workbench (CCW) ネイティブです。したがって、アトミックデータ型を参照する場合、CCW のタグ名をコピーしてサーバーのタグアドレスフィールドに貼り付けることでそのタグ名が有効になります。

関連項目: *高度な使用事例*

フォーマット	構文
配列要素	<ネイティブタグ名> [次元 1, 次元 2, 次元 3]
オフセットがある配列*	<ネイティブタグ名> {列数} <ネイティブタグ名> {行数}{列数}
オフセットがない配列*	<ネイティブタグ名> {列数} <ネイティブタグ名> {行数}{列数}
ビット	<ネイティブタグ名>.ビット

フォーマット	構文
	<ネイティブタグ名>.[ビット]
標準	<ネイティブタグ名>
文字列	<ネイティブタグ名>

*これらのフォーマットでは複数の要素が要求されることがあるため、配列データが渡される順序は配列タグの次元によって異なります。たとえば、行数 x 列数 = 4 でネイティブタグが 3X3 要素の配列である場合、array_tag [0,0]、array_tag [0,1]、array_tag [0,2]、array_tag [1,0] の順序で要素が参照されます。ネイティブタグが 2X10 要素の配列であった場合には結果が異なります。詳細については、**配列データの順序**を参照してください。

拡張アドレスフォーマット

配列要素

1 つ以上 (ただし 3 つ以下) の次元が指定されている必要があります。

構文	例	注記
<ネイティブタグ名> [次元 1]	tag_1 [5]	該当なし
<ネイティブタグ名> [次元 1, 次元 2]	tag_1 [2, 3]	該当なし
<ネイティブタグ名> [次元 1, 次元 2, 次元 3]	tag_1 [2, 58, 547]	該当なし

オフセットがある配列

このクラスでは複数の要素が要求されることがあるため、配列データが渡される順序は配列タグの次元によって異なります。

構文	例	注記
<ネイティブタグ名> [オフセット] {列数}	tag_1 [5] {8}	読み書きされる要素の数は、行数と列数を掛けた数と等しくなります。行数が指定されていない場合、行数はデフォルトで 1 になります。配列の 1 つ以上の要素がアドレス指定されている必要があります。 配列はゼロオフセットで開始します (すべての次元で配列のインデックスが 0)。
<ネイティブタグ名> [オフセット] {行数}{列数}	tag_1[5] {2} {4}	

注記: 行数 x 列数 = 4 でネイティブタグが 3X3 要素の配列である場合、array_tag [0,0]、array_tag [0,1]、array_tag [0,2]、array_tag [1,0] の順序で要素が参照されます。ネイティブタグが 2X10 要素の配列であった場合には結果が異なります。

オフセットがない配列

このクラスでは複数の要素が要求されることがあるため、配列データが渡される順序は配列タグの次元によって異なります。

構文	例	注記
<ネイティブタグ名> {列数}	tag_1 {8}	読み書きされる要素の数は、行数と列数を掛けた数と等しくなります。行数が指定されていない場合、行数はデフォルトで 1 になります。配列の 1 つ以上の要素がアドレス指定されている必要があります。 配列はゼロオフセットで開始します (すべての次元で配列のインデックスが 0)。
<ネイティブタグ名> {行数} {列数}	tag_1 {2} {4}	

注記: たとえば、行数 x 列数 = 4 でネイティブタグが 3X3 要素の配列である場合、array_tag [0,0]、array_tag [0,1]、array_tag [0,2]、array_tag [1,0] の順序で要素が参照されます。ネイティブタグが 2X10 要素の配列であった場合には結果が異なります。

ビット

構文	例	注記
<ネイティブタグ名>.ビット	tag_1.0	該当なし
<ネイティブタグ名>.[ビット]	tag_1.[0]	該当なし

標準

構文	例	注記
<ネイティブタグ名>	tag_1	該当なし

String

構文	例	注記
<ネイティブタグ名>	tag_1	読み書きされる文字の数は文字列長と等しくなり、1 以上でなければなりません。

*1 次元、2 次元、および 3 次元配列で要素がどのように参照されるかについては、**配列データの順序**を参照してください。*

タグの有効範囲

変数の有効範囲はプログラムにローカルであるかコントローラにグローバルです。

- ローカル変数はそのプロジェクト内の特定のプログラムに割り当てられ、そのプログラムでのみ使用できます。
- グローバル変数はそのプロジェクト内のコントローラに属し、そのプロジェクト内のすべてのプログラムで使用できます。

ローカル変数

ローカル変数 (特定のプログラムにだけ有効なタグ) にコントローラの通信ポートから直接アクセスすることはできないので、このドライバー内で直接にはサポートされていません。アクセスが必要な場合、ローカル変数テーブルからグローバル変数テーブルにタグを切り取って貼り付けます。

グローバル変数

グローバル変数 (コントローラ全体で有効なタグ) は、コントローラでグローバルに有効なネイティブタグです。どのようなプログラムまたはタスクでもグローバルタグにアクセスできますが、グローバルタグを参照可能な手段の数は、そのネイティブデータ型および使用されているアドレスフォーマットによって異なります。

ユーザー定義のデータ型

ユーザーは 80 文字ではなく 12 文字の STRING 型など、固有のデータ型を作成できます。このようなユーザー定義のデータ型をローカルまたはグローバル変数として使用できます。

構造体型変数

Micro800 コントローラには構造体変数はありません。ユーザーは固有のデータ型を作成できますが、各メンバーに一意の名前が必要です。

アトミックデータ型のアドレス指定

アドレスフォーマットが使用可能な各ネイティブデータ型に推奨される使用法とアドレス指定の方法を以下の表に示します。各データ型の詳細なアドレス指定の方法については、**詳細**をクリックしてください。

注記: 空のセルは必ずしもサポートしていないことを意味するものではありません。

BOOL

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	Boolean	Boolean (BOOL 型 1 次元配列)	Boolean 配列 (BOOL 型 1 次元配列)		
例	BOOLTAG	BOOLARR[0]	BOOLARR[0]{32}		

SINT、USINT、BYTE

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	Byte、Char	Byte、Char	Byte 配列、Char 配列 (SINT 型 1/2/3 次元配列)	Boolean (SINT 内のビット)	
例	SINTTAG	SINTARR[0]	SINTARR[0]{4}	SINTTAG.0	j

INT、UINT、WORD

タグ	標準	配列要素	オフセットがない配列	ビット	文字列
データ型 詳細	Word、Short	Word、Short	Word 配列、Short 配列 (INT 型 1/2/3 次元配列)	Boolean (INT 内のビット)	
例	INTTAG	INTARR[0]	INTARR[0]{4}	INTTAG.0	

DINT、UDINT、DWORD

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	DWord、Long	DWord、Long	DWord 配列、Long 配列	Boolean (DINT 内のビット)	詳細
例	DINTTAG	DINTARR[0]	DINTARR[0]{4}	DINTTAG.0	

LINT、ULINT、LWORD

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	Double、Date	Double、Date	Double 配列		
例	LINTTAG	LINTARR[0]	LINTARR[0]{4}		

REAL

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	Float	Float	Float 配列		
例	REALTAG	REALARR[0]	REALARR[0]{4}		

LREAL

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	Double	Double	Double 配列		
例	LREALTAG	LREALARR[0]	LREALARR[0]{4}		

SHORT_STRING

タグ	標準	配列要素	オフセットがある/ない配列	ビット	文字列
データ型 詳細	String	String			
例	STRINGTAG	STRINGARR[0]			

関連項目: アドレスのフォーマット

アドレス指定 構造的データ型

構造体は構造体レベルでは参照できません。アトミック構造体メンバーのみをアドレス指定できます。詳細については、以下の例を参照してください。

ネイティブタグ
MyTimer @ TIMER

有効なクライアント/サーバータグ
アドレス = MyTimer.ACC
データ型 = DWord

無効なクライアント/サーバータグ
アドレス = MyTimer
データ型 = ??

STRING データ型のアドレス指定

STRING は定義済みのネイティブデータ型であり、その構造体には DATA と LEN の 2 つのメンバーが含まれています。DATA は SINT の配列であり、STRING の文字が格納されます。LEN は DINT であり、クライアントに表示される DATA 内の文字数を表します。

LEN と DATA はアトミックメンバーなので、これらはクライアント/サーバーから別個に参照される必要があります。

説明	構文	例
STRING の値	DATA/<STRING の最大長>	MYSTRING.DATA/82
STRING の実際の長さ	LEN	MYSTRING.LEN

読み取り

DATA から読み取られた STRING は以下で打ち切られます。

a. 出現した 1 つ目の Null 終端。

b. a) が先に出現しない場合、LEN 内の値。

c. a) または b) が先に出現しない場合、<STRING の最大長>。

例
PLC で MYSTRING.DATA には "Hello World" が格納されていますが、LEN が手動で 5 に設定されています。MYSTRING.DATA/82 を読み取ると "Hello" と表示されます。LEN を 20 に設定した場合、MYSTRING.DATA/82 では "Hello World" と表示されます。

書き込み

STRING の値が DATA に書き込まれると、書き込まれた DATA の長さも LEN に書き込まれます。なんらかの理由によって LEN への書き込みが失敗した場合、(コントローラへの DATA の書き込みが成功したにもかかわらず) DATA への書き込み操作も失敗したものと見なされます。

注記: この動作は STRING 型のネイティブタグ (および STRING から派生したカスタムタグ) 専用に設計されました。独自の STRING を UDT に実装するユーザーは以下の点に注意してください。

- STRING として参照される DATA メンバーと DINT として参照される LEN メンバーを持つ UDT が存在する場合、その UDT で LEN が DATA の長さであるかどうかに関係なく LEN への書き込みは成功します。LEN が DATA の長さでない場合、この可能性を排除するため、UDT を設計する際には注意が必要です。
- STRING として参照される DATA メンバーを持つが LEN メンバーがない UDT が存在する場合、LEN への書き込みは警告なしで失敗し、DATA への影響はありません。

例
MYSTRING.DATA/82 に値 "Hello World" が格納されています。MYSTRING.LEN に 11 が格納されています。値 "Alarm Triggered" が MYSTRING.DATA/82 に書き込まれた場合、MYSTRING.LEN には 15 が書き込まれます。MYSTRING.LEN への書き込みに失敗した場合、MYSTRING.LEN には以前の値 11 が格納され、MYSTRING.DATA/82 では最初の 11 文字 ("Alarm Trigg") が表示されます。MYSTRING.DATA/82 への書き込みに失敗した場合、いずれのタグも影響を受けません。

配列データの順序

1 次元配列 - array [dim1]

1 次元配列データはコントローラとの間で昇順でやり取りされます。

for (dim1 = 0; dim1 < dim1_max; dim1++)

例: 3 要素の配列
array [0]
array [1]
array [2]

2 次元配列 - array [dim1, dim2]

2 次元配列データはコントローラとの間で昇順でやり取りされます。

for (dim1 = 0; dim1 < dim1_max; dim1++)
for (dim2 = 0; dim2 < dim2_max; dim2++)

例: 3x3 要素の配列
array [0, 0]
array [0, 1]
array [0, 2]
array [1, 0]
array [1, 1]
array [1, 2]
array [2, 0]
array [2, 1]
array [2, 2]

3 次元配列 - array [dim1, dim2, dim3]

3 次元配列データはコントローラとの間で昇順でやり取りされます。

for (dim1 = 0; dim1 < dim1_max; dim1++)
for (dim2 = 0; dim2 < dim2_max; dim2++)
for (dim3 = 0; dim3 < dim3_max; dim3++)

例: 3x3x3 要素の配列
array [0, 0, 0]
array [0, 0, 1]
array [0, 0, 2]
array [0, 1, 0]
array [0, 1, 1]
array [0, 1, 2]
array [0, 2, 0]
array [0, 2, 1]
array [0, 2, 2]
array [1, 0, 0]
array [1, 0, 1]
array [1, 0, 2]
array [1, 1, 0]
array [1, 1, 1]
array [1, 1, 2]
array [1, 2, 0]
array [1, 2, 1]
array [1, 2, 2]
array [2, 0, 0]
array [2, 0, 1]
array [2, 0, 2]
array [2, 1, 0]
array [2, 1, 1]
array [2, 1, 2]
array [2, 2, 0]
array [2, 2, 1]
array [2, 2, 2]

高度な使用事例

特定のアトミックデータ型の詳細な使用事例については、以下のリンクからリストを選択してください。

BOOL

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Boolean	ネイティブタグは 1 次元配列でなければなりません。
オフセットがある配列	Boolean 配列	1. ネイティブタグは 1 次元配列でなければなりません。 2. オフセットは 32 ビット境界内に収める必要があります。 3. 要素の数は 32 の因数でなければなりません。
オフセットがない配列	Boolean 配列	1. ネイティブタグは 1 次元配列でなければなりません。 2. 要素の数は 32 の因数でなければなりません。
ビット	Boolean	1. ネイティブタグは 1 次元配列でなければなりません。 2. 範囲は 0 から 31 に制限されます。
標準	Boolean、Byte、Char、Word、Short、BCD、DWord、Long、LBCD、Float*	なし
文字列	サポートされていません。	

*Float 型の値は Float 形式のネイティブタグの額面と等しくなります (非 IEEE 浮動小数点数)。

例

ハイライトされている例は一般的な使用事例を示しています。

BOOL 型アトミックタグ - booltag = True

サーバータグアドレス	フォーマット	データ型	注記
booltag	標準	Boolean	値 = True
booltag	標準	Byte	値 = 1
booltag	標準	Word	値 = 1
booltag	標準	DWord	値 = 1
booltag	標準	Float	値 = 1.0
booltag [3]	配列要素	Boolean	無効: タグが配列ではありません。

サーバータグアドレス	フォーマット	データ型	注記
booltag [3]	配列要素	Word	無効: タグが配列ではありません。
booltag {1}	オフセットがない配列	Word	無効: サポートされていません。
booltag {1}	オフセットがない配列	Boolean	無効: サポートされていません。
booltag [3] {32}	オフセットがある配列	Boolean	無効: タグが配列ではありません。
booltag . 3	ビット	Boolean	無効: タグが配列ではありません。
booltag / 1	String	文字列	無効: サポートされていません。
booltag / 4	String	文字列	無効: サポートされていません。

BOOL 配列タグ - bitarraytag = [0,1,0,1]

サーバータグアドレス	フォーマット	データ型	注記
bitarraytag	標準	Boolean	無効: タグが配列であってはなりません。
bitarraytag	標準	Byte	無効: タグが配列であってはなりません。
bitarraytag	標準	Word	無効: タグが配列であってはなりません。
bitarraytag	標準	DWord	無効: タグが配列であってはなりません。
bitarraytag	標準	Float	無効: タグが配列であってはなりません。
bitarraytag [3]	配列要素	Boolean	値 = True
bitarraytag [3]	配列要素	Word	無効: データ型が不適切です。
bitarraytag {3}	オフセットがない配列	Word	無効: タグが配列であってはなりません。
bitarraytag {1}	オフセットがない配列	Word	無効: タグが配列であってはなりません。
bitarraytag {1}	オフセットがない配列	Boolean	無効: 配列のサイズは 32 の因数でなければなりません。
bitarraytag {32}	オフセットがない配列	Boolean	値 = [0,1,0,1,...]
bitarraytag [3] {32}	オフセットがある配列	Boolean	オフセットは 32 ビット境界で開始する必要があります。
bitarraytag[0]{32}	オフセットがある配列	Boolean	値 = [0,1,0,1,...]
bitarraytag[32]{64}	オフセットがある配列	Boolean	構文は有効です。要素が範囲外です。
bitarraytag . 3	ビット	Boolean	値 = True
bitarraytag / 1	String	文字列	無効: サポートされていません。
bitarraytag / 4	String	文字列	無効: サポートされていません。

SINT、USINT、BYTE

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Byte、Char、Word、Short、BCD、DWord、Long、LBCD、Float***	ネイティブタグは配列でなければなりません。
オフセットがある配列	Byte 配列、Char 配列、Word 配列、Short 配列、BCD 配列**、DWord 配列、Long 配列、LBCD 配列**、Float 配列**、***	ネイティブタグは配列でなければなりません。
オフセット	Boolean 配列	1. SINT 内のビットを配列形式にするにはこの事例に従います。これは Boolean 表記の SINT の配

フォーマット	サポートされるデータ型	注記
がない配列	Byte 配列、Char 配列、Word 配列、Short 配列、BCD 配列**、DWord 配列、Long 配列、LBCD 配列**、Float 配列**、***	列ではありません。 2. SINT 内のビットのみに適用されます。例: tag_1.0{8}。 3. ビットオフセットと配列サイズの和が 8 ビットを超えてはなりません。例: tag_1.1{8} は SINT を超えていますが、tag_1.0{8} は超えていません。 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。
ビット	Boolean	1. 範囲は 0 から 7 に制限されます。 2. ネイティブタグが配列である場合、ビットクラス参照の先頭に配列要素クラス参照を付ける必要があります。例: tag_1 [2,2,3].0。
標準	Boolean*、Byte、Char、Word、Short、BCD、DWord、Long、LBCD、Float***	なし
文字列	String	1. 1 つの要素にアクセスする場合、ネイティブタグが配列である必要はありません。 **注記:** 文字列の値は SINT 値に相当する ASCII 文字です。例: SINT = 65dec = "A"。 2. 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。文字列の値は文字列内のすべての SINT に相当する Null 終端 ASCII 文字です。 文字列内の 1 文字 = 1 SINT。

*ゼロ以外の値は True にクランプされます。
**配列の各要素は SINT 配列内の要素に対応しています。配列はパックされません。
***Float 型の値は Float 形式のネイティブタグの額面と等しくなります (非 IEEE 浮動小数点数)。

例

ハイライトされている例は SINT、USINT、BYTE での一般的な使用事例を示しています。

SINT、USINT、BYTE アトミックタグ - sinttag = 122 (10 進)

サーバータグアドレス	フォーマット	データ型	注記
sinttag	標準	Boolean	値 = True
sinttag	標準	Byte	値 = 122
sinttag	標準	Word	値 = 122
sinttag	標準	DWord	値 = 122
sinttag	標準	Float	値 = 122.0
sinttag [3]	配列要素	Boolean	無効: タグが配列ではありません。さらに、Boolean は無効です。
sinttag [3]	配列要素	Byte	無効: タグが配列ではありません。
sinttag {3}	オフセットがない配列	Byte	無効: タグが配列ではありません。
sinttag {1}	オフセットがない配列	Byte	値 = [122]

サーバータグアドレス	フォーマット	データ型	注記
sinttag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
sinttag [3] {1}	オフセットがある配列	Byte	無効: タグが配列ではありません。
sinttag . 3	ビット	Boolean	値 = True
sinttag . 0 {8}	オフセットがない配列	Boolean	値 = [0,1,0,1,1,1,1,0] 122 のビット値
sinttag / 1	文字列	String	無効: サポートされていない構文/データ型です。
sinttag / 4	文字列	String	無効: サポートされていない構文/データ型です。

SINT、USINT、BYTE 配列タグ - sintarraytag [4,4] = [[83,73,78,84],[5,6,7,8],[9,10,11,12],[13,14,15,16]]

サーバータグアドレス	フォーマット	データ型	注記
sintarraytag	標準	Boolean	無効: タグが配列であってはなりません。
sintarraytag	標準	Byte	無効: タグが配列であってはなりません。
sintarraytag	標準	Word	無効: タグが配列であってはなりません。
sintarraytag	標準	DWord	無効: タグが配列であってはなりません。
sintarraytag	標準	Float	無効: タグが配列であってはなりません。
sintarraytag [3]	配列要素	Byte	無効: サーバータグで次元 2 のアドレスが欠落しています。
sintarraytag [1,3]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
sintarraytag [1,3]	配列要素	Byte	値 = 8
sintarraytag {10}	オフセットがない配列	Byte	値 = [83,73,78,84,5,6,7,8,9,10]
sintarraytag {2} {5}	オフセットがない配列	Word	値 = [83,73,78,84,5] [6,7,8,9,10]
sintarraytag {1}	オフセットがない配列	Byte	値 = 83
sintarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
sintarraytag [1,3] {4}	オフセットがある配列	Byte	値 = [8,9,10,11]
sintarraytag . 3	ビット	Boolean	無効: タグはアトミックの場所を参照する必要があります。
sintarraytag [1,3] . 3	ビット	Boolean	値 = 1
sintarraytag [1,3] . 0 {8}	オフセットがない配列	Boolean	値 = [0,0,0,1,0,0,0,0]
sintarraytag / 1	文字列	String	無効: サポートされていない構文/データ型です。
sintarraytag / 4	文字列	String	無効: サポートされていない構文/データ型です。

INT、UINT、WORD

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Byte、Char**、Word、Short、BCD、DWord、Long、LBCD、Float****	ネイティブタグは配列でなければなりません。
オフセットがある配列	Byte 配列、Char 配列**、Word 配列、Short 配列、BCD 配列、DWord 配列、Long 配列、LBCD 配列***、Float 配列***、****	ネイティブタグは配列でなければなりません。

フォーマット	サポートされるデータ型	注記
オフセットがない配列	Boolean 配列 Byte 配列、Char 配列**、Word 配列、Short 配列、BCD 配列、DWord 配列、Long 配列、LBCD 配列***、Float 配列***、****	1. INT 内のビットを配列形式にするにはこの事例に従います。これは Boolean 表記の INT の配列ではありません。 2. INT 内のビットのみに適用されます。例: tag_1.0{16}。 3. ビットオフセットと配列サイズの和が 16 ビットを超えてはなりません。例: tag_1.1{16} は INT を超えていますが、tag_1.0{16} は超えていません。 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。
ビット	Boolean	1. 範囲は 0 から 15 に制限されます。 2. ネイティブタグが配列である場合、ビットクラス参照の先頭に配列要素クラス参照を付ける必要があります。例: tag_1 [2,2,3].0。
標準	Boolean*、Byte、Char**、Word、Short、BCD、DWord、Long、LBCD、Float****	なし
文字列	String	1. 1 つの要素にアクセスする場合、コントローラタグが配列である必要はありません。 **注記:** 文字列の値は INT 値 (255 にクランプ) に相当する ASCII 文字です。例: INT = 65dec = "A"。 2. 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。文字列の値は文字列内のすべての INT (255 にクランプ) に相当する Null 終端 ASCII 文字です。 文字列内の 1 文字 = 1 INT、255 にクランプ。 INT 文字列はパックされません。効率を上げるには、代わりに SINT 型文字列または STRING 型構造体を使用します。

*ゼロ以外の値は True にクランプされます。
**255 を超える値は 255 にクランプされます。
*** 配列の各要素は INT 配列内の要素に対応しています。配列はパックされません。
****Float 型の値は Float 形式のネイティブタグの額面と等しくなります (非 IEEE 浮動小数点数)。

例

ハイライトされている例は INT、UINT、WORD での一般的な使用事例を示しています。

INT、UINT、WORD アトミックタグ - inttag = 65534 (10 進)

サーバータグアドレス	クラス	データ型	注記
inttag	標準	Boolean	値 = True
inttag	標準	Byte	値 = 255
inttag	標準	Word	値 = 65534
inttag	標準	DWord	値 = 65534

サーバータグアドレス	クラス	データ型	注記
inttag	標準	Float	値 = 65534.0
inttag [3]	配列要素	Boolean	無効: タグが配列ではありません。さらに、Boolean は無効です。
inttag [3]	配列要素	Word	無効: タグが配列ではありません。
inttag {3}	オフセットがない配列	Word	無効: タグが配列ではありません。
inttag {1}	オフセットがない配列	Word	値 = [65534]
inttag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
inttag [3] {1}	オフセットがある配列	Word	無効: タグが配列ではありません。
inttag . 3	ビット	Boolean	値 = True
inttag . 0 {16}	オフセットがない配列	Boolean	値 = [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] 65534 のビット値
inttag / 1	文字列	String	無効: サポートされていない構文/データ型です。
inttag / 4	文字列	String	無効: サポートされていない構文/データ型です。

INT、UINT、WORD での配列タグ - intarraytag [4,4] = [[73,78,84,255],[256,257,258,259],[9,10,11,12],[13,14,15,16]]

サーバータグアドレス	クラス	データ型	注記
intarraytag	標準	Boolean	無効: タグが配列であってはなりません。
intarraytag	標準	Byte	無効: タグが配列であってはなりません。
intarraytag	標準	Word	無効: タグが配列であってはなりません。
intarraytag	標準	DWord	無効: タグが配列であってはなりません。
intarraytag	標準	Float	無効: タグが配列であってはなりません。
intarraytag [3]	配列要素	Word	無効: サーバータグで次元 2 のアドレスが欠落しています。
intarraytag [1,3]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
intarraytag [1,3]	配列要素	Word	値 = 259
intarraytag {10}	オフセットがない配列	Byte	値 = [73,78,84,255,255,255,255,255,9,10]
intarraytag {2} {5}	オフセットがない配列	Word	値 = [73,78,84,255,256] [257,258,259,9,10]
intarraytag {1}	オフセットがない配列	Word	値 = 73
intarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
intarraytag [1,3] {4}	オフセットがある配列	Word	値 = [259,9,10,11]
intarraytag . 3	ビット	Boolean	無効: タグはアトミックの場所を参照する必要があります。
intarraytag [1,3] . 3	ビット	Boolean	値 = 0
intarraytag [1,3] . 0 {16}	オフセットがない配列	Boolean	値 = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] 259 のビット値
intarraytag / 1	文字列	String	無効: サポートされていない構文/データ型です。
intarraytag / 3	文字列	String	無効: サポートされていない構文/データ型です。

DINT、UDINT、DWORD

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Byte、Char**、Word、Short、BCD***、DWord、Long、LBCD、Float****	ネイティブタグは配列でなければなりません。
オフセットがある配列	Byte 配列、Char 配列**、Word 配列、Short 配列、BCD 配列***、DWord 配列、Long 配列、LBCD 配列、Float 配列****	ネイティブタグは配列でなければなりません。
オフセットがない配列	Boolean 配列 Byte 配列、Char 配列**、Word 配列、Short 配列、BCD 配列***、DWord 配列、Long 配列、LBCD 配列、Float 配列****	1. DINT 内のビットを配列形式にするにはこの事例に従います。これは Boolean 表記の DINT の配列ではありません。 2. DINT 内のビットのみに適用されます。例: tag_1.0{32}。 3. ビットオフセットと配列サイズの和が 32 ビットを超えてはなりません。例: tag_1.1{32} は DINT を超えていますが、tag_1.0{32} は超えていません。 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。
ビット	Boolean	1. 範囲は 0 から 31 に制限されます。 2. ネイティブタグが配列である場合、ビットクラス参照の先頭に配列要素クラス参照を付ける必要があります。例: tag_1 [2,2,3].0。
標準	Boolean* Byte、Char** Word、Short、BCD*** DWord、Long、LBCD Float****	なし
文字列	String	1. 1 つの要素にアクセスする場合、コントローラタグが配列である必要はありません。 **注記:** 文字列の値は DINT 値 (255 にクランプ) に相当する ASCII 文字です。例: SINT = 65dec = "A"。 2. 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。文字列の値は文字列内のすべての DINT (255 にクランプ) に相当する Null 終端 ASCII 文字です。 文字列内の 1 文字 = 1 DINT、255 にクランプ。 DINT 文字列はパックされません。効率を上げるには、代わりに SINT 型文字列または STRING 型構造体を使用します。

*ゼロ以外の値は True にクランプされます。
**255 を超える値は 255 にクランプされます。
***65535 を超える値は 65535 にクランプされます。
****Float 型の値は Float 形式のネイティブタグの額面と等しくなります (非 IEEE 浮動小数点数)。

例

ハイライトされている例

DINT、UDINT、および DWORD 型アトミックタグ - dinttag = 70000 (10 進)

サーバータグアドレス	フォーマット	データ型	注記
dinttag	標準	Boolean	値 = True
dinttag	標準	Byte	値 = 255
dinttag	標準	Word	値 = 65535
dinttag	標準	DWord	値 = 70000
dinttag	標準	Float	値 = 70000.0
dinttag [3]	配列要素	Boolean	無効: タグが配列ではありません。さらに、Boolean は無効です。
dinttag [3]	配列要素	DWord	無効: タグが配列ではありません。
dinttag {3}	オフセットがない配列	DWord	無効: タグが配列ではありません。
dinttag {1}	オフセットがない配列	DWord	値 = [70000]
dinttag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です
dintag [3] {1}	オフセットがある配列	DWord	無効: タグが配列ではありません。
dinttag . 3	ビット	Boolean	値 = False
dinttag . 0 {32}	オフセットがない配列	Boolean	値 = [0,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,1,0,...0] 70000 のビット値
dinttag	文字列	String	無効: サポートされていない構文/データ型です。
dinttag	文字列	String	無効: サポートされていない構文/データ型です。

DINT、UDINT、および DWORD 配列タグ - dintarraytag [4,4] = [[68,73,78,84],[256,257,258,259], [9,10,11,12],[13,14,15,16]]

サーバータグアドレス	フォーマット	データ型	注記
dintarraytag	標準	Boolean	無効: タグが配列であってはなりません。
dintarraytag	標準	Byte	無効: タグが配列であってはなりません。
dintarraytag	標準	Word	無効: タグが配列であってはなりません。
dintarraytag	標準	DWord	無効: タグが配列であってはなりません。
dintarraytag	標準	Float	無効: タグが配列であってはなりません。
dintarraytag [3]	配列要素	DWord	無効: サーバータグで次元 2 のアドレスが欠落しています。
dintarraytag [1,3]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
dintarraytag [1,3]	配列要素	DWord	値 = 259
dintarraytag {10}	オフセットがない配列	Byte	値 = [68,73,78,84,255,255,255,255,9,10]
dintarraytag {2}{5}	オフセットがない配列	DWord	値 = [68,73,78,84,256] [257,258,259,9,10]
dintarraytag {1}	オフセットがない配列	DWord	値 = 68
dintarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
dintarraytag [1,3]{4}	オフセットがある配列	DWord	値 = [259,9,10,11]
dintarraytag . 3	ビット	Boolean	無効: タグはアトミックの場所を参照する必要があります。

サーバータグアドレス	フォーマット	データ型	注記
dintarraytag [1,3] . 3	ビット	Boolean	値 = 0
dintarraytag [1,3] .0 {32}	オフセットがない配列	Boolean	値 = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] 259 のビット値
dintarraytag	文字列	String	無効: サポートされていない構文/データ型です。
dintarraytag	文字列	String	無効: サポートされていない構文/データ型です。

LINT、ULINT、LWORD

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Double* Date**	ネイティブタグは配列でなければなりません。
オフセットがある配列	Double 配列*	ネイティブタグは配列でなければなりません。
オフセットがない配列	Double 配列*	複数の要素にアクセスする場合、コントローラタグは配列でなければなりません。
ビット	サポートされていません。	サポートされていません。
標準	Double* Date**	なし
文字列	サポートされていません。	サポートされていません。

*Double 型の値は Float 形式のコントローラタグの額面と等しくなります (非 IEEE 浮動小数点数)。
**Date 型の値は現地時刻ではなく協定世界時刻 (UTC) です。

例

ハイライトされている例は LINT、ULINT、LWORD での一般的な使用事例を示しています。

LINT、ULINT、LWORD アトミックタグ - linttag = 2007-01-01T16:46:40.000 (日付) == 1.16767E+15 (10 進)

サーバータグアドレス	フォーマット	データ型	注記
linttag	標準	Boolean	無効: Boolean はサポートされていません。
linttag	標準	Byte	無効: Byte はサポートされていません。
linttag	標準	Word	無効: Word はサポートされていません。
linttag	標準	Double	値 = 1.16767E+15
linttag	標準	Date	値 = 2007-01-01T16:46:40.000*
linttag [3]	配列要素	Boolean	無効: タグが配列ではありません。さらに、Boolean は無効です。
linttag [3]	配列要素	Double	無効: タグが配列ではありません。
linttag {3}	オフセットがない配列	Double	無効: タグが配列ではありません。
linttag {1}	オフセットがない配列	Double	値 = [1.16767E+15]
linttag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
lintag [3] {1}	オフセットがある配列	Double	無効: タグが配列ではありません。
linttag . 3	ビット	Boolean	無効: サポートされていない構文/データ型です。

サーバータグアドレス	フォーマット	データ型	注記
linttag / 1	String	String	無効: サポートされていない構文/データ型です。

*Date 型の値は現地時刻ではなく協定世界時刻 (UTC) です。

LINT、ULINT、LWORD 配列タグ -
dintarraytag [2,2] = [0, 1.16767E+15],[9.4666E+14, 9.46746E+14] ここで:
1.16767E+15 == 2007-01-01T16:46:40.000 (日付)
9.4666E+14 == 1999-12-31T17:06:40.000
9.46746E+14 == 2000-01-1T17:00:00.000
0 == 1970-01-01T00:00:00.000

サーバータグアドレス	フォーマット	データ型	注記
lintarraytag	標準	Boolean	無効: Boolean はサポートされていません。
lintarraytag	標準	Byte	無効: Byte はサポートされていません。
lintarraytag	標準	Word	無効: Word はサポートされていません。
lintarraytag	標準	Double	無効: タグが配列であってはなりません。
lintarraytag	標準	Date	無効: タグが配列であってはなりません。
lintarraytag [1]	配列要素	Double	無効: サーバータグで次元 2 のアドレスが欠落しています。
lintarraytag [1,1]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
lintarraytag [1,1]	配列要素	Double	値 = 9.46746E+14
lintarraytag [1,1]	配列要素	Date	値 = 2000-01-01T17:00:00.000*
lintarraytag {4}	オフセットがない配列	Double	値 = [0, 1.16767E+15, 9.4666E+14, 9.46746E+14]
lintarraytag {2} {2}	オフセットがない配列	Double	値 = [0, 1.16767E+15][9.4666E+14, 9.46746E+14]
lintarraytag {4}	オフセットがない配列	Date	無効: Date 配列はサポートされていません。
lintarraytag {1}	オフセットがない配列	Double	値 = 0
lintarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
lintarraytag [0,1] {2}	オフセットがある配列	Double	値 = [1.16767E+15, 9.4666E+14]
lintarraytag . 3	ビット	Boolean	無効: サポートされていない構文/データ型です。
lintarraytag / 1	String	String	無効: サポートされていない構文/データ型です。

*Date 型の値は現地時刻ではなく協定世界時刻 (UTC) です。

REAL

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Byte、Char**、Word、Short、BCD***、DWord、Long、LBCD、Float****	ネイティブタグは配列でなければなりません。
オフセットがある配列	Byte 配列、Char 配列**、Word 配列、Short 配列、BCD 配列***、DWord 配列、Long 配列、LBCD 配列、Float 配列****	ネイティブタグは配列でなければなりません。

フォーマット	サポートされるデータ型	注記
オフセットがない配列	Boolean 配列 Byte 配列、Char 配列**、Word 配列、Short 配列、BCD 配列***、DWord 配列、Long 配列、LBCD 配列、Float 配列****	1. REAL 内のビットを配列形式にするにはこの事例に従います。これは Boolean 表記の REAL の配列ではありません。 2. REAL 内のビットのみに適用されます。例: tag_1.0{32}。 3. ビットオフセットと配列サイズの和が 32 ビットを超えてはなりません。例: tag_1.1{32} は REAL を超えていますが、tag_1.0{32} は超えていません。 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。
ビット	Boolean	1. 範囲は 0 から 31 に制限されます。 2. ネイティブタグが配列である場合、ビットクラス参照の先頭に配列要素クラス参照を付ける必要があります。例: tag_1 [2,2,3].0。 **注記:** Float はビットの参照を可能にするために DWord にキャストされます。
標準	Boolean* Byte、Char** Word、Short、BCD*** DWord、Long、LBCD Float****	なし
文字列	String	1. 1 つの要素にアクセスする場合、コントローラタグが配列である必要はありません。 **注記:** 文字列の値は REAL 値 (255 にクランプ) に相当する ASCII 文字です。例: SINT = 65dec = "A"。 2. 複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。文字列の値は文字列内のすべての REAL (255 にクランプ) に相当する Null 終端 ASCII 文字です。 文字列内の 1 文字 = 1 REAL、255 にクランプ。 REAL 文字列はパックされません。効率を上げるには、代わりに SINT 型文字列または STRING 型構造体を使用します。

*ゼロ以外の値は True にクランプされます。
**255 を超える値は 255 にクランプされます。
***65535 を超える値は 65535 にクランプされます。
****Float 型の値は有効な IEEE 単精度浮動小数点数です。

例

ハイライトされている例は一般的な使用事例を示しています。

REAL アトミックタグ - realtag = 512.5 (10 進)

サーバータグアドレス	フォーマット	データ型	注記
realtag	標準	Boolean	値 = True
realtag	標準	Byte	値 = 255
realtag	標準	Word	値 = 512
realtag	標準	DWord	値 = 512
realtag	標準	Float	値 = 512.5
realtag [3]	配列要素	Boolean	無効: タグが配列ではありません。さらに、Boolean は無効です。
realtag [3]	配列要素	DWord	無効: タグが配列ではありません。
realtag {3}	オフセットがない配列	DWord	無効: タグが配列ではありません。
realtag {1}	オフセットがない配列	Float	値 = [512.5]
realtag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
realtag [3] {1}	オフセットがある配列	Float	無効: タグが配列ではありません。
realtag . 3	ビット	Boolean	値 = True
realtag . 0 {32}	オフセットがない配列	Boolean	値 = [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,...0] 512 のビット値
realtag	String	String	無効: サポートされていない構文/データ型です。
realtag	String	String	無効: サポートされていない構文/データ型です。

REAL 配列タグ - realarraytag [4,4] = [[82.1,69.2,65.3,76.4],[256.5,257.6,258.7,259.8],[9.0,10.0,11.0,12.0],[13.0,14.0,15.0,16.0]]

サーバータグアドレス	フォーマット	データ型	注記
realarraytag	標準	Boolean	無効: タグが配列であってはなりません。
realarraytag	標準	Byte	無効: タグが配列であってはなりません。
realarraytag	標準	Word	無効: タグが配列であってはなりません。
realarraytag	標準	DWord	無効: タグが配列であってはなりません。
realarraytag	標準	Float	無効: タグが配列であってはなりません。
realarraytag [3]	配列要素	Float	無効: サーバータグで次元 2 のアドレスが欠落しています。
realarraytag [1,3]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
realarraytag [1,3]	配列要素	Float	値 = 259.8
realarraytag {10}	オフセットがない配列	Byte	値 = [82,69,65,76,255,255,255,255,9,10]
realarraytag {2} {5}	オフセットがない配列	Float	値 = [82.1,69.2,65.3,76.4,256.5] [257.6,258.7,259.8,9,10]
realarraytag {1}	オフセットがない配列	Float	値 = 82.1
realarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
realarraytag [1,3] {4}	オフセットがある配列	Float	値 = [259.8,9.0,10.0,11.0]
realarraytag . 3	ビット	Boolean	無効: タグはアトミックの場所を参照する必要があります。
realarraytag [1,3] . 3	ビット	Boolean	値 = 0
realarraytag [1,3] . 0 {32}	オフセットがない配列	Boolean	値 = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] 259 のビット値

サーバータグアドレス	フォーマット	データ型	注記
realarraytag	String	String	無効: サポートされていない構文/データ型です。
realarraytag	String	String	無効: サポートされていない構文/データ型です。

LREAL

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	Double*	ネイティブタグは配列でなければなりません。
オフセットがある配列	Double 配列	ネイティブタグは配列でなければなりません。
オフセットがない配列	Double 配列	複数の要素にアクセスする場合、ネイティブタグは配列でなければなりません。
ビット	Boolean	無効: サポートされていない構文/データ型です。
標準	Double*	なし
文字列	String	無効: サポートされていない構文/データ型です。

*Double 型の値は有効な IEEE 倍精度浮動小数点数です。

例

ハイライトされている例は一般的な使用事例を示しています。

LREAL アトミックタグ - lrealtag = 512.5 (10 進)

サーバータグアドレス	フォーマット	データ型	注記
lrealtag	標準	Boolean	無効 : サポートされていないデータ型です。
lrealtag	標準	Byte	無効 : サポートされていないデータ型です。
lrealtag	標準	Word	無効 : サポートされていないデータ型です。
lrealtag	標準	DWord	無効 : サポートされていないデータ型です。
lrealtag	標準	Double	値 = 512.5
lrealtag [3]	配列要素	Boolean	無効: タグが配列ではなく、Boolean は無効です。
lrealtag [3]	配列要素	DWord	無効: タグが配列ではありません。
lrealtag {3}	オフセットがない配列	DWord	無効: タグが配列ではありません。
lrealtag {1}	オフセットがない配列	Double	値 = [512.5]
lrealtag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
lrealtag [3] {1}	オフセットがある配列	Float	無効: タグが配列ではありません。
lrealtag . 3	ビット	Boolean	無効 : サポートされていないデータ型です。
lrealtag . 0 {32}	オフセットがない配列	Boolean	無効 : サポートされていないデータ型です。
lrealtag	文字列	String	無効: サポートされていない構文/データ型です。
lrealtag	文字列	String	無効: サポートされていない構文/データ型です。

LREAL 配列タグ - realarraytag [4,4] = [[82.1,69.2,65.3,76.4],[256.5,257.6,258.7,259.8], [9.0,10.0,11.0,12.0],[13.0,14.0,15.0,16.0]]

サーバータグアドレス	フォーマット	データ型	注記
lrealarraytag	標準	Boolean	無効: タグが配列であってはなりません。
lrealarraytag	標準	Byte	無効: タグが配列であってはなりません。
lrealarraytag	標準	Word	無効: タグが配列であってはなりません。

サーバータグアドレス	フォーマット	データ型	注記
lrealarraytag	標準	DWord	無効: タグが配列であってはなりません。
lrealarraytag	標準	Double	無効: タグが配列であってはなりません。
lrealarraytag [3]	配列要素	Double	無効: サーバータグで次元 2 のアドレスが欠落しています。
lrealarraytag [1,3]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
lrealarraytag [1,3]	配列要素	Double	値 = 259.8
lrealarraytag {10}	オフセットがない配列	Byte	無効: サポートされていないデータ型です。
lrealarraytag {2} {5}	オフセットがない配列	Double	値 = [82.1,69.2,65.3,76.4,256.5] [257.6,258.7,259.8,9,10]
lrealarraytag {1}	オフセットがない配列	Double	値 = 82.1
lrealarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
lrealarraytag [1,3] {4}	オフセットがある配列	Double	値 = [259.8,9.0,10.0,11.0]
lrealarraytag . 3	ビット	Boolean	無効: タグはアトミックの場所を参照する必要があります。
lrealarraytag [1,3] . 3	ビット	Boolean	値 = 0
lrealarraytag [1,3] . 0 {32}	オフセットがない配列	Boolean	無効: サポートされていない構文/データ型です。
lrealarraytag	文字列	String	無効: サポートされていない構文/データ型です。
lrealarraytag	文字列	String	無効: サポートされていない構文/データ型です。

SHORT_STRING

このフォーマットの詳細については、アドレスのフォーマットを参照してください。

フォーマット	サポートされるデータ型	注記
配列要素	String	ネイティブタグは配列でなければなりません。
オフセットがある配列	対象外	なし
オフセットがない配列	対象外	なし
ビット	対象外	なし
標準	String	文字列の長さは、ネイティブタグに含まれている長さエンコーディングに基づきます。文字列に印刷不可文字が含まれている場合、これらも文字列に含まれます。
文字列	該当なし	タグアドレスで文字列の長さを指定する必要があります。

例

ハイライトされている例は一般的な使用事例を示しています。

SHORT_STRING アトミックタグ - stringtag = "mystring"

サーバータグアドレス	フォーマット	データ型	注記
stringtag	標準	String	値 = mystring。
stringtag	標準	Byte	無効: Byte はサポートされていません。

サーバータグアドレス	フォーマット	データ型	注記
stringtag	標準	Word	無効: Word はサポートされていません。
stringtag [3]	配列要素	Boolean	無効: タグが配列ではなく、Boolean は無効です。
stringtag [3]	配列要素	Double	無効: タグが配列ではありません。
stringtag {3}	オフセットがない配列	Double	無効: タグが配列ではありません。
stringtag {1}	オフセットがない配列	Double	値 = [1.16767E+15]。
stringtag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
lintag [3] {1}	オフセットがある配列	Double	無効: タグが配列ではありません。
stringtag . 3	ビット	Boolean	無効: サポートされていない構文/データ型です。
stringtag / 1	文字列	String	無効: サポートされていない構文/データ型です。

SHORT_STRING 配列タグ - stringarraytag[2,2] = [one,two].[three,four]

サーバータグアドレス	フォーマット	データ型	注記
stringarraytag	標準	Boolean	無効: Boolean はサポートされていません。
stringarraytag	標準	Byte	無効: Byte はサポートされていません。
stringarraytag	標準	Word	無効: Word はサポートされていません。
stringarraytag	標準	Double	無効: タグが配列であってはなりません。
stringarraytag	標準	Date	無効: タグが配列であってはなりません。
stringarraytag [1]	配列要素	Double	無効: サーバータグで次元 2 のアドレスが欠落しています。
stringarraytag [1,1]	配列要素	Boolean	無効: 配列要素には Boolean を使用できません。
stringarraytag [1,1]	配列要素	String	値: "four"
stringarraytag {4}	オフセットがない配列	String	無効: String 配列はサポートされていません。
stringarraytag {2} {2}	オフセットがない配列	String	無効: String 配列はサポートされていません。
stringarraytag {1}	オフセットがない配列	Boolean	無効: データ型が不適切です。
stringarraytag [0, 1] {2}	オフセットがある配列	String	値: "three"
stringarraytag . 3	ビット	Boolean	無効: サポートされていない構文/データ型です。
stringarraytag / 1	文字列	String	無効: サポートされていない構文です。

エラーコード

以降のセクションでは、サーバーのイベントログに記録されるエラーコードを定義しています。特定のエラーコードタイプの詳細については、以下のリストからリンクを選択してください。

カプセル化プロトコルエラーコード
CIP エラーコード

カプセル化プロトコルエラーコード

次のエラーコードは 16 進数で表示されます。

エラーコード	説明
0001	コマンドが処理されませんでした。
0002	コマンド用のメモリがありません。
0003	データの形式が不適切であるか不完全です。
0064	セッション ID が無効です。
0065	ヘッダーの長さが無効です。

エラーコード	説明
0069	要求されたプロトコルバージョンはサポートされていません。
0070	ターゲット ID が無効です。

CIP エラーコード

次のエラーコードは 16 進数で表示されます。

エラーコード	説明
0001	接続エラー*
0002	リソースが不足しています
0003	値が無効です
0004	IOI を解読できなかったかタグが存在しません
0005	宛先が不明です
0006	要求されたデータは応答パケットに収まりません
0007	接続が失われました
0008	サポートされていないサービスです
0009	データセグメントにエラーがあるか属性値が無効です
000A	属性リストエラー
000B	状態がすでに存在します
000C	オブジェクトモデルが競合しています
000D	オブジェクトがすでに存在します
000E	属性を設定できません
000F	Permission denied
0010	デバイスの状態が競合しています
0011	応答が大きすぎます
0012	プリミティブがフラグメント化されています
0013	サービスを実行するには指定されたコマンドデータ/パラメータでは不十分です
0014	属性がサポートされていません
0015	指定されたデータでは多すぎます
001A	ブリッジ要求が大きすぎます
001B	ブリッジ応答が大きすぎます
001C	属性リストが不足しています
001D	属性リストが無効です
001E	組み込みサービスエラー
001F	接続中にエラーが発生しました**
0022	無効な応答を受信しました
0025	キーセグメントエラー
0026	指定された IOI Word の数は IOI Word 数と一致しません
0027	リストに予期しない属性があります

- ***関連項目:** *0x0001 拡張エラーコード*
- ****関連項目:** *0x001F 拡張エラーコード*

Allen-Bradley 固有のエラーコード

エラーコード (16 進数)	説明
00FF	一般的なエラー*

- ***関連項目:** *0x00FF 拡張エラーコード*
- *一覧にないエラーコードについては、Rockwell Automation のドキュメントを参照してください。*

0x0001 拡張エラーコード

次のエラーコードは 16 進数で表示されます。

エラーコード	説明
0100	接続が使用中です。
0103	転送はサポートされていません。
0106	オーナーシップが競合しています。
0107	接続が見つかりません。
0108	接続タイプが無効です。
0109	接続サイズが無効です。
0110	モジュールが設定されていません。
0111	EPR はサポートされていません。
0114	モジュールが間違っています。
0115	デバイスタイプが間違っています。
0116	リビジョンが間違っています。
0118	構成フォーマットが無効です。
011A	アプリケーションが接続されていません。
0203	接続がタイムアウトになりました。
0204	未接続メッセージがタイムアウトになりました。
0205	未接続送信パラメータエラー。
0206	メッセージが大きすぎます。
0301	バッファメモリがありません。
0302	帯域幅を使用できません。
0303	スクリーナを使用できません。
0305	署名が一致しています。
0311	ポートを使用できません。
0312	リンクアドレスを使用できません。
0315	セグメントタイプが無効です。
0317	接続がスケジュールされていません。
0318	自己へのリンクアドレスは無効です。

一覧にないエラーコードについては、Rockwell Automation のドキュメントを参照してください。

0x0C 拡張エラーコード

次のエラーコードは 16 進数で表示されます。

エラーコード	説明
0203	接続がタイムアウトになりました。

一覧にないエラーコードについては、Rockwell Automation のドキュメントを参照してください。

0x00FF 拡張エラーコード

次のエラーコードは 16 進数で表示されます。

エラーコード	説明
2104	アドレスが範囲外です。
2105	データオブジェクトの末尾以降にアクセスしようとしました。

エラーコード	説明
2106	データは使用中です。
2107	データ型が無効であるかサポートされていません。

一覧にないエラーコードについては、Rockwell Automation のドキュメントを参照してください。

イベントログメッセージ

次の情報は、メインユーザーインタフェースの「イベントログ」枠に記録されたメッセージに関するものです。「イベントログ」詳細ビューのフィルタと並べ替えについては、サーバーのヘルプを参照してください。サーバーのヘルプには共通メッセージが多数含まれているので、これらも参照してください。通常は、可能な場合、メッセージのタイプ (情報、警告) とトラブルシューティングに関する情報が提供されています。

サポートされていないコントローラです。| ベンダー ID = <ベンダー>、製品タイプ = <タイプ>、製品コード = <コード>、製品名 = '<製品>'。

エラータイプ:
警告

デバイスから受信したフレームにエラーが含まれています。

エラータイプ:
警告

考えられる原因:

1. パケットに不整列が発生しています (原因は PC とデバイス間の接続/切断)。
2. デバイスのケーブル接続の不良によりノイズが発生しています。
3. 不正なフレームサイズを受信しました。
4. TNS の不一致があります。
5. デバイスから無効な応答コマンドが返されました。

解決策:
介入しなくてもドライバーはこのエラーから回復できますが、ケーブル接続またはデバイス自体に修正すべき問題がある可能性があります。

フレーミングエラーによりタグの書き込み要求が失敗しました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. 指定されたタグの書き込み要求は、不正な要求サービスコードにより、再試行を何度も繰り返した後に失敗しました。
2. 指定されたタグの書き込み要求は、予想されるバイト数よりも多いかまたは少ないバイト数を受信したため、再試行を何度も繰り返した後に失敗しました。

解決策:

1. ケーブル接続またはデバイス自体に問題がある可能性があります。
2. ドライバーがこのエラーから回復する可能性を高めるには、再試行回数を増やしてください。

フレーミングエラーによりタグの読み取り要求が失敗しました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. 指定されたタグの読み取り要求は、不正な要求サービスコードにより、再試行を何度も繰り返した後に失敗しました。
2. 指定されたタグの読み取り要求は、予想されるバイト数よりも多いかまたは少ないバイト数を受信したため、再試行を何度も繰り返した後に失敗しました。

解決策:

1. ケーブル接続またはデバイス自体に問題がある可能性があります。
2. ドライバーがこのエラーから回復する可能性を高めるには、再試行回数を増やしてください。

フレーミングエラーによりブロック読み取り要求が失敗しました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値> (要素)。

エラータイプ:
警告

考えられる原因:

1. 指定されたタグの読み取り要求は、不正な要求サービスコードにより、再試行を何度も繰り返した後に失敗しました。
2. 指定されたタグの読み取り要求は、予想されるバイト数よりも多いかまたは少ないバイト数を受信したため、再試行を何度も繰り返した後に失敗しました。

解決策:

1. ケーブル接続またはデバイス自体に問題がある可能性があります。
2. ドライバーがこのエラーから回復する可能性を高めるには、再試行回数を増やしてください。

デバイスにタグを書き込めません。| タグアドレス = '<アドレス>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
指定されたタグの書き込み要求時、パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。CIP および拡張コードの定義を参照してください。

関連項目:
CIP エラーコード

デバイスからタグを読み取れません。| タグアドレス = '<アドレス>'、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
指定されたタグの読み取り要求時、パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。CIP および拡張コードの定義を参照してください。

関連項目:
CIP エラーコード

デバイスからブロックを読み取れません。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値>、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:
指定されたタグのブロック読み取り要求時、パケットの CIP 部分の範囲でデバイスがエラーを返しました。

解決策:
返されたエラーコードによって解決策が異なります。CIP および拡張コードの定義を参照してください。

関連項目:
CIP エラーコード

デバイスにタグを書き込めません。コントローラタグのデータ型が不明です。| タグアドレス = '<アドレス>'、不明なデータ型 = <タイプ>。

エラータイプ:
警告

考えられる原因:
現在のところネイティブタグのデータ型がサポートされていないため、指定されたタグの書き込み要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

デバイスからタグを読み取れません。コントローラタグのデータ型が不明です。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'、不明なデータ型 = <タイプ>。

エラータイプ:
警告

考えられる原因:
現在のところネイティブタグのデータ型がサポートされていないため、指定されたタグの書き込み要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

デバイスからブロックを読み取れません。コントローラタグのデータ型が不明です。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値>、不明なデータ型 = <タイプ>。

エラータイプ:
警告

考えられる原因:
現在のところネイティブタグのデータ型がサポートされていないため、指定されたタグの書き込み要求は失敗しました。

解決策:
テクニカルサポートまでご連絡の上、この型に関するサポートの追加をご要望ください。

デバイスにタグを書き込めません。データ型がサポートされていません。| タグアドレス = '<アドレス>'、サポートされていないデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型がサポートされていないため、指定されたタグの書き込み要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。このエラーによりタグは非アクティブ化され、再度処理されることはありません。

関連項目:
アトミックデータ型のアドレス指定

デバイスからタグを読み取れません。データ型がサポートされていません。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'、サポートされていないデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型がサポートされていないため、指定されたタグの読み取り要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。このエラーによりタグは非アクティブ化され、再度処理されることはありません。

関連項目:
アトミックデータ型のアドレス指定

デバイスからブロックを読み取れません。データ型がサポートされていません。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値> (要素)、サポートされていないデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
クライアントタグのデータ型がサポートされていないため、指定されたタグの読み取り要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。このエラーによりタグは非アクティブ化され、再度処理されることはありません。

関連項目:
アトミックデータ型のアドレス指定

タグに書き込めません。タグには不正なデータ型です。| タグアドレス = '<アドレス>'、不正なデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
タグのデータ型がサポートされていないため、指定されたタグの要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のネイティブタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題は解決します。

関連項目:
アトミックデータ型のアドレス指定

デバイスからタグを読み取れません。このタグには不正なデータ型です。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'、不正なデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
タグのデータ型がサポートされていないため、指定されたタグの要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のネイティブタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題は解決します。

関連項目:
アトミックデータ型のアドレス指定

デバイスからブロックを読み取れません。このブロックには不正なデータ型です。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値> (要素)、不正なデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
タグのデータ型がサポートされていないため、指定されたタグの要求は失敗しました。

解決策:
タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のネイティブタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題は解決します。

関連項目:
アトミックデータ型のアドレス指定

デバイスにタグを書き込めません。タグは複数要素の配列をサポートしません。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
複数要素の配列からネイティブタグへのアクセスをドライバーがサポートしないため、指定されたタグへの読み取り要求は失敗しました。

解決策:
タグのデータ型またはアドレスをサポート対象のものに変更してください。

関連項目:
アトミックデータ型のアドレス指定

デバイスからタグを読み取れません。タグは複数要素の配列をサポートしません。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
複数要素の配列からネイティブタグへのアクセスをドライバーがサポートしないため、指定されたタグへの読み取り要求は失敗しました。

解決策:
タグのデータ型またはアドレスをサポート対象のものに変更してください。このエラーによりタグは非アクティブ化され、再度処理されることはありません。

関連項目:
アトミックデータ型のアドレス指定

デバイスからブロックを読み取れません。ブロックは複数要素の配列をサポートしません。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値>(要素)。

エラータイプ:
警告

考えられる原因:
複数要素の配列からネイティブタグへのアクセスをドライバーがサポートしないため、指定されたタグへの読み取り要求は失敗しました。

解決策:
タグのデータ型またはアドレスをサポート対象のものに変更してください。このエラーによりブロックは非アクティブ化され、再度処理されることはありません。

関連項目:
アトミックデータ型のアドレス指定

デバイスにタグを書き込めません。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間の接続が切断しています。
2. 接続の通信パラメータが不正です。
3. この名前のデバイスに不正なアドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定したアドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:
このエラーによりタグは非アクティブ化され、再度処理されることはありません。

デバイスからタグを読み取れません。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間の接続が切断しています。
2. 接続の通信パラメータが不正です。
3. この名前のデバイスに不正なアドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定したアドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:
このエラーによりタグは非アクティブ化され、再度処理されることはありません。

デバイスからブロックを読み取れません。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値>。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間の接続が切断しています。
2. 接続の通信パラメータが不正です。
3. この名前のデバイスに不正なアドレスが割り当てられている可能性があります。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定したアドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:
このエラーによりブロックは非アクティブ化され、再度処理されることはありません。

デバイスが CIP エラーを返しました。| ステータスコード = <コード>、拡張ステータスコード = <コード>。

エラータイプ:
警告

考えられる原因:
要求時、パケットの CIP 部分の範囲でデバイスがエラーを返しました。要求内のすべての読み取りと書き込みが失敗しました。

解決策:
返されたエラーコードによって解決策が異なります。CIP コードを参照してください。

メモリをタグに割り当てることができませんでした。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

考えられる原因:
タグの構築に必要なリソースを割り当てることができませんでした。タグはプロジェクトに追加されません。

解決策:
使用していないアプリケーションを終了する、仮想メモリの量を増やすなどをした後でもう一度試してください。

デバイスが DF1 エラーを返しました。

エラータイプ:
警告

考えられる原因:
サーバーが無効な応答を送信しました。

解決策:

1. ドライバーはこのエラーからの回復を試みます。
2. デバイスから返されたエラーコードによって解決策が異なります。

関連項目:
エラーマトリックス

デバイスからタグを読み取れません。内部メモリが無効です。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

デバイスからタグを読み取れません。タグには不正なデータ型です。| タグアドレス = '<アドレス>'、不正なデータ型 = '<タイプ>'。

エラータイプ:
警告

考えられる原因:
タグのデータ型がサポートされていないため、指定されたタグの要求は失敗しました。

解決策:

1. 要求したデータ型を確認し、修正してください。
2. タグのデータ型をサポート対象の型に変更してください。たとえば、BOOL 配列のネイティブタグにデータ型 Short は不正です。その場合、データ型を Boolean に変更すると問題は解決します。

関連項目:
アトミックデータ型のアドレス指定

デバイスからタグを読み取れません。内部メモリが無効です。タグは非アクティブ化されました。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

デバイスからブロックを読み取れません。内部メモリが無効です。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値> (要素)。

エラータイプ:
警告

デバイスのアドレスに書き込めません。内部メモリが無効です。| タグアドレス = '<アドレス>'。

エラータイプ:
警告

デバイスからブロックを読み取れません。ブロックは非アクティブ化されました。| ブロック先頭 = '<アドレス>'、ブロックサイズ = <数値>、CIP エラー = <コード>、拡張エラー = <コード>。

エラータイプ:
警告

考えられる原因:

1. デバイスとホスト PC 間の接続が切断しています。
2. 接続の通信パラメータが不正です。

解決策:

1. PC とデバイス間のケーブル接続を確認してください。
2. この名前のデバイスに正しいポートが指定されていることを確認してください。
3. この名前のデバイスに指定したアドレスが実際のデバイスのアドレスと一致することを確認してください。

注記:
このエラーによりブロックの要素は非アクティブ化され、再度処理されることはありません。

関連項目:
CIP エラーコード

デバイス識別情報の詳細。| ID = <ID>、ベンダー ID = <ベンダー>、製品タイプ = <タイプ>、製品コード = <コード>、リビジョン = '<リビジョン>'、製品名 = '<製品>'、製品シリアル番号 = <数値>。

エラータイプ:

情報

デバイスはフラグメントされた読み取り/書き込みサービスをサポートしません。自動的に非フラグメントサービスにフォールバックします。

エラータイプ:
情報

用語集

ネイティブタグベースのアドレス指定

用語	定義
配列要素	ネイティブ配列タグ内の要素。クライアント/サーバーアクセスの場合、この要素はアトミックでなければなりません。例: ARRAYTAG [0]。
オフセットがある配列	アドレスにネイティブ配列要素が指定されているクライアント/サーバー配列タグ。例: ARRAYTAG [0] {5}。
オフセットがない配列	アドレスにネイティブ配列要素が指定されていないクライアント/サーバー配列タグ。例: ARRAYTAG {5}。
アトミックデータ型	事前に定義されている非構造的ネイティブデータ型。例: SINT、DINT。
アトミックタグ	アトミックデータ型で定義されているネイティブタグ。
クライアント	OPC、DDE、または専用のクライアント/サーバープロトコルを利用してサーバーに接続する HMI/SCADA などのデータブリッジソフトウェアパッケージ。
クライアント/サーバーデータ型	サーバーで静的に定義されているかクライアントで動的に定義されるタグのデータ型。クライアントでサポートされるデータ型は、使用されているクライアントによって異なります。*
クライアント/サーバータグ	サーバーで静的に定義されているかクライアントで動的に定義されるタグ。これらのタグはネイティブタグとは別のものです。ネイティブタグが参照される際、ネイティブタグの名前はクライアント/サーバータグのアドレスになります。
クライアント/サーバー配列	サーバーと一部のクライアントによってサポートされる、行 x 列によるデータ表現フォーマット。すべてのクライアントが配列をサポートしているわけではありません。
CCW	Connected Components Workbench。
ネイティブデータ型	Micro800 コントローラ用 CCW で定義されているデータ型。
ネイティブタグ	Micro800 コントローラ用 CCW で定義されているタグ。
ネイティブ配列データ型	Micro800 コントローラ用 CCW では多次元配列 (1、2、3 次元が可能) がサポートされます。すべてのアトミックデータ型がネイティブ配列をサポートしています。一部の構造的データ型はネイティブ配列をサポートしていません。
配列タグ	ネイティブ配列データ型で定義されているネイティブタグ。
定義済みのデータ型	Micro800 コントローラ用 CCW でサポートされ事前に定義されているネイティブデータ型。*
ユーザー定義データ型	CCW でサポートされ Micro800 コントローラのユーザーによって定義されているネイティブデータ型。*
サーバー	このドライバーを利用している OPC/DDE/専用サーバー。
構造的データ型	データ型がアトミックまたは構造体であるメンバーから成る、定義済みまたはユーザー定義のデータ型。
構造体タグ	構造的データ型で定義されているネイティブタグ。

*サーバーでサポートされるデータ型はデータ型の説明で一覧しています。

索引

B

C

D

F

I

L

R

S

W

あ

い

え

お

か

き

く

さ

し

す

た

ち

て

と

ね

は

ふ

へ

ほ

め

も

ゆ

り

ろ

漢字