Modbus ASCII シリアルドライバー

目次

Modbus ASCII シリアルドライバー

ヘルプバージョン 1.061

目次

概要

Modbus ASCII シリアルドライバーは、Modbus ASCII シリアルデバイスが OPC クライアントアプリケーション (HMI、SCADA、Historian、MES、ERP など) やカスタムアプリケーションに接続するための信頼性の高い手段を提供します。これは Modbus ASCII プロトコルをサポートするシリアルデバイスで使用するためのものです。このドライバーには、1 回の要求でデバイスから要求されるデータの量、32 ビット Double レジスタ値の Word オーダー、32 ビットおよび 16 ビットレジスタ値のバイトオーダー、アドレスベースの調整を制御するための機能が組み込まれています。特定の RTS タイミングを必要とするラジオモデムで使用する RTS 回線処理も、このドライバーで制御することができます。

設定

サポートされるデバイス

Modbus ASCII 対応デバイス
Daniels/Omni/Elliot レジスタアドレス指定を使用しているフローコンピュータ

通信 プロトコル

Modbus ASCII プロトコル

チャネルとデバイスの制限値

このドライバーでサポートされているチャネルの最大数は 100 です。このドライバーでサポートされているデバイスの最大数は、1 つのチャネルにつき 247 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の通信が可能です。これは COM ID ダイアログでチャネルプロパティを介して設定できます。詳細については、サーバーのヘルプファイルを参照してください。

サポートされている通信プロパティ

「ボーレート」: 1200、2400、9600、19200
「パリティ」: 奇数、偶数、なし
データビット: 8
「ストップビット」: 1、2

注記: リストされている構成は一部のデバイスではサポートされていないことがあります。

フロー制御

RS232/RS485 コンバータを使用している場合、必要なフロー制御のタイプはコンバータの要件によって異なります。コンバータには、フロー制御を必要としないものと、RTS フローを必要とするものがあります。コンバータのフロー要件についてはコンバータのドキュメントを参照してください。自動フロー制御を備えた RS485 コンバータが推奨されます。

注記:

1. 製造メーカーから供給されている通信ケーブルを使用している場合、チャネルプロパティでフロー制御の設定として**「RTS」**または**「RTS 常時」**を選択する必要があることがあります。
2. Modbus ASCII シリアルドライバー では「RTS 手動」フロー制御オプションがサポートされています。この選択は、特別な RTS タイミング特性を必要とするラジオモデムで動作するようにドライバーを構成する際に使用します。RTS 手動フロー制御の詳細については、サーバーのヘルプファイルを参照してください。

通信シリアル化

Modbus ASCII シリアルドライバー では、データ転送を一度に 1 つのチャネルに制限するかどうかを指定する通信シリアル化がサポートされています。

チャネルのプロパティ - 一般

このサーバーでは、複数の通信ドライバーを同時に使用することができます。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識別情報を指定します。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情報を指定します。
「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」: このチャネル用のプロトコル/ドライバーを指定します。チャネル作成時に選択されたデバイスドライバーを指定します。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。大規模なクライアントアプリケーションを開発した場合は、プロパティを変更しないようにしてください。オペレータがプロパティを変更したりサーバーの機能にアクセスしたりすることを防ぐため、適切なユーザー役割を使用し、権限を正しく管理する必要があります。

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ます。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、サーバーのヘルプで「通信診断」を参照してください。

タグ数

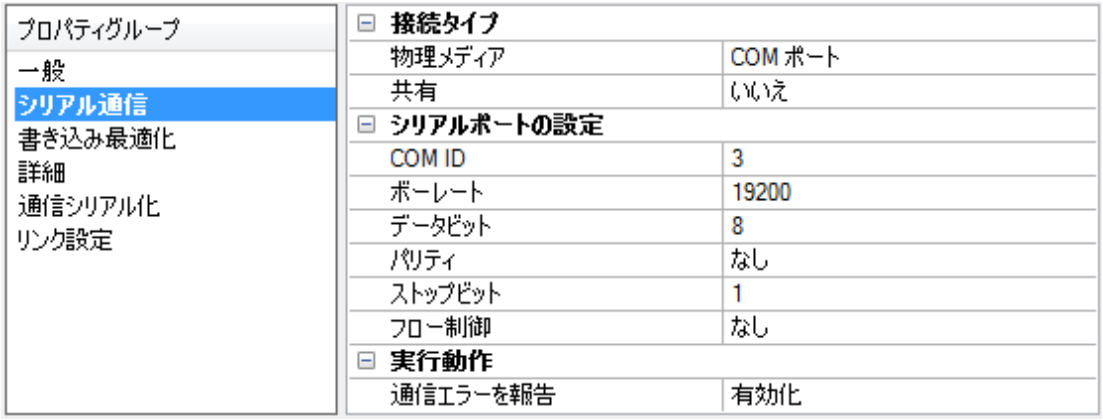
「静的タグ」: デバイスレベルまたはチャネルレベルで定義される静的タグの数を指定します。この情報は、トラブルシューティングと負荷分散を行う場合に役立ちます。

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。
次のいずれかのセクションをクリックしてください: 接続タイプ、シリアルポートの設定 またはイーサネット設定、および実行動作。

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。オペレータがプロパティを変更したりサーバーの機能にアクセスしたりすることを防ぐため、適切なユーザー役割を使用し、権限を正しく管理する必要があります。
- 使用する特定の通信パラメータを定義する必要があります。ドライバーによっては、チャネルが同一の通信パラメータを共有できる場合とできない場合があります。仮想ネットワークに設定できる共有シリアル接続は 1 つだけです (「チャネルのプロパティ - シリアル通信」を参照してください)。



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。次のオプションがあります:「モデム」、「イーサネットカプセル化」、「COM ポート」、「なし」。デフォルトは「COM ポート」です。

1. **「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。
2. **「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。
3. **「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。
4. **「イーサネットカプセル化」**: イーサネットカプセル化機能を使用して通信を行う場合は、このオプションを選択します。この機能については、イーサネット設定セクションを参照してください。
5. **「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選択します。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」**: データ転送の前にRTS 回線を事前にオン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。
  - **「遅延オフ」**: データ転送後にRTS 回線を解放するまでの時間を指定します。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」**: 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、サーバーヘルプの「Using Ethernet Encapsulation」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。

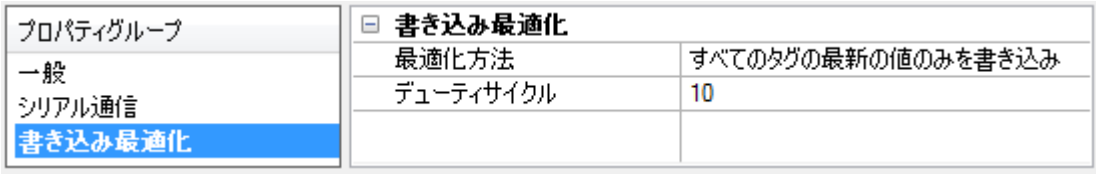
- **「アイドル接続を閉じる」**: チャネル上のクライアントによっていずれのタグも参照されなくなった場合、モデム接続を閉じます。デフォルトは「有効化」です。
- **「クローズするまでのアイドル時間」**: すべてのタグが除去されてからモデム接続を閉じるまでサーバーが待機する時間を指定します。デフォルトは 15 秒です。

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をデバイスに遅延なく届ける必要があります。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することができます。



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、このプロパティは無効になります。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。

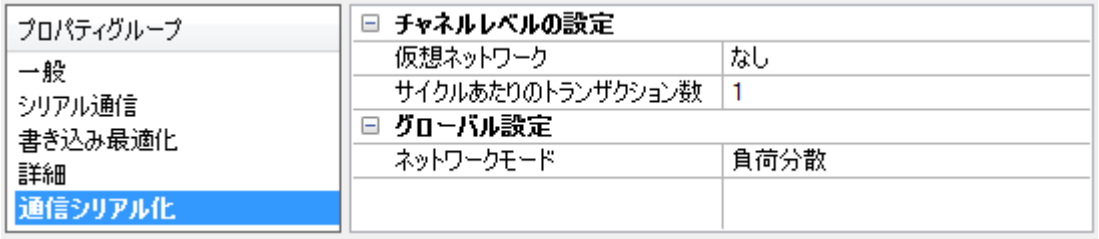
「デバイス間遅延」: 通信チャネルが同じチャネルの現在のデバイスからデータを受信した後、次のデバイスに新しい要求を送信するまで待機する時間を指定します。ゼロ (0) を指定すると遅延は無効になります。

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クライアント無線です。同じクライアント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」: 通信シリアル化のチャネルの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」: 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。

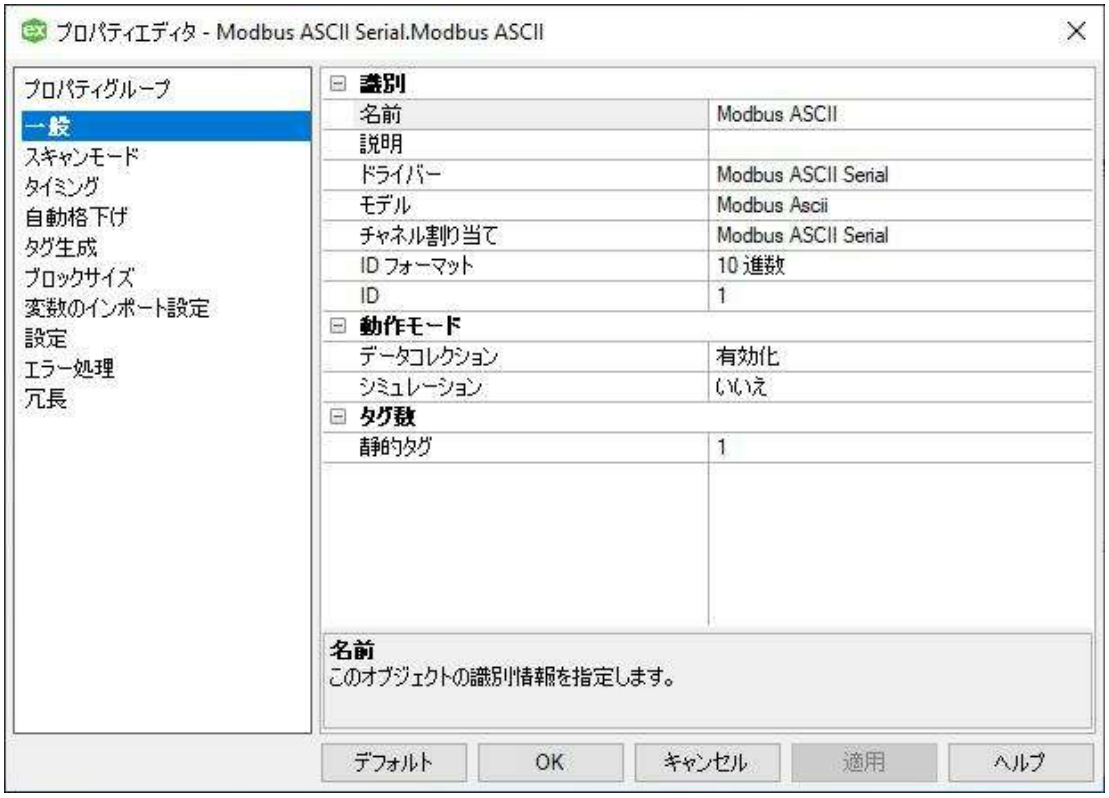
1. 書き込みが保留中になっているチャネルの優先順位が最も高くなります。
2. (内部のプラグインまたは外部のクライアントインタフェースによって) 明示的な読み取りが保留中になっているチャネルは、その読み取りの優先順位に基づいて優先順位が決まります。
3. スキャン読み取りおよびその他の定期的イベント (ドライバー固有)。

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?

デバイスのプロパティ - 一般



識別

「名前」: このデバイスのユーザー定義の識別情報。

「説明」: このデバイスに関するユーザー定義の情報。

「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」: このデバイスのバージョン。

「ID フォーマット」: デバイス識別情報のフォーマット方法を選択します。オプションには「10 進数」、「8 進数」、「16 進数」があります。

「ID」: 一意のデバイス番号。Modbus シリアルデバイスには 1 から 247 の範囲のデバイス ID が割り当てられます。

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。

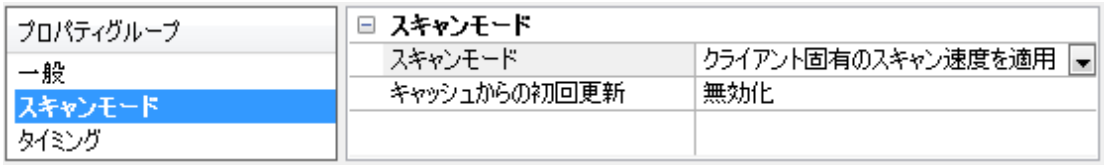
注記:

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 - 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。



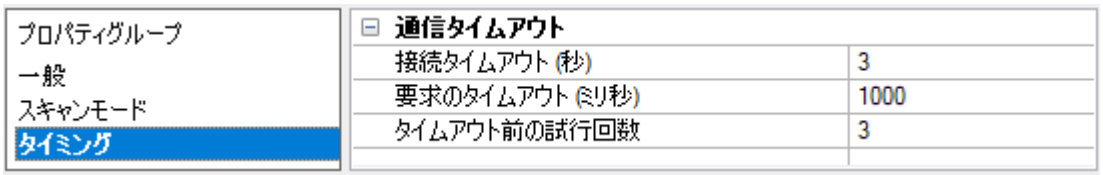
「スキャンモード」: 購読しているクライアントに送信される更新についてデバイス内のタグをどのようにスキャンするかを指定します。オプションの説明は次のとおりです。

- **「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、OPC 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期更新」: 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、初期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

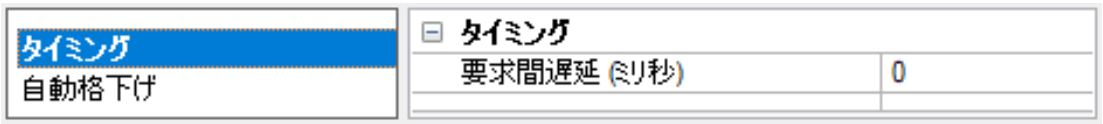
「接続タイムアウト」: このプロパティ (イーサネットベースのドライバーで主に使用) は、リモートデバイスとのソケット接続を確立するために必要な時間を制御します。デバイスの接続時間は、同じデバイスへの通常の通信要求よりも長くかかることがよくあります。有効な範囲は 1 から 30 秒です。デフォルトは通常は 3 秒ですが、各ドライバーの特性によって異なる場合があります。この設定がドライバーでサポートされていない場合、無効になります。
注記: UDP 接続の特性により、UDP を介して通信する場合には接続タイムアウトの設定は適用されません。

「要求のタイムアウト」: すべてのドライバーがターゲットデバイスからの応答の完了を待機する時間を決定するために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 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」: 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。

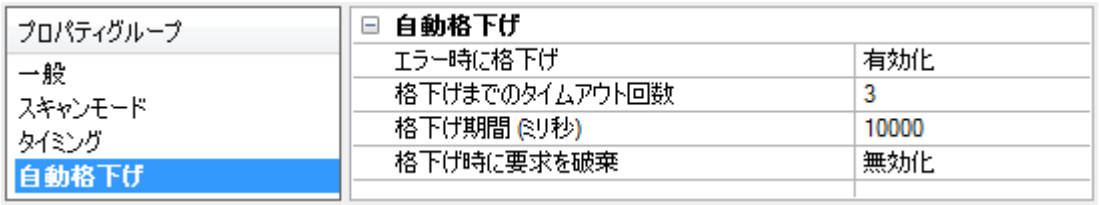
タイミング

「要求間遅延」: 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 - タグ生成

自動タグデータベース生成機能によって、アプリケーションの設定がプラグアンドプレイ操作になります。デバイス固有のデータに対応するタグのリストを自動的に構築するよう通信ドライバーを設定できます。これらの自動生成されたタグ (サポートしているドライバーの特性によって異なる) をクライアントからブラウズできます。

一部のデバイスやドライバーは自動タグデータベース生成のフル機能をサポートしていません。また、すべてのデバイスやドライバーが同じデータ型をサポートするわけではありません。詳細については、データ型の説明を参照するか、各ドライバーがサポートするデータ型のリストを参照してください。

ターゲットデバイスが独自のローカルタグデータベースをサポートしている場合、ドライバーはそのデバイスのタグ情報を読み取って、そのデータを使用してサーバー内にタグを生成します。デバイスが名前付きのタグをネイティブにサポートしていない場合、ドライバーはそのドライバー固有の情報に基づいてタグのリストを作成します。この 2 つの条件の例は次のとおりです。

1. データ取得システムが独自のローカルタグデータベースをサポートしている場合、通信ドライバーはデバイスで見つかったタグ名を使用してサーバーのタグを構築します。
2. イーサネット I/O システムが独自の使用可能な I/O モジュールタイプの検出をサポートしている場合、通信ドライバーはイーサネット I/O ラックにプラグイン接続している I/O モジュールのタイプに基づいてサーバー内にタグを自動的に生成します。

注記: 自動タグデータベース生成の動作モードを詳細に設定できます。詳細については、以下のプロパティの説明を参照してください。

プロパティグループ
一般
スキャンモード
タイミング
自動格下げ
タグ生成

タグ生成	
デバイス起動時	起動時に生成しない
重複タグ	作成時に削除
親グループ	
自動生成されたサブグループを許可	有効化

「プロパティ変更時」: デバイスが、特定のプロパティが変更された際の自動タグ生成をサポートする場合、**「プロパティ変更時」**オプションが表示されます。これはデフォルトで**「はい」**に設定されていますが、**「いいえ」**に設定してタグ生成を実行する時期を制御できます。この場合、タグ生成を実行するには**「タグを作成」**操作を手動で呼び出す必要があります。

「デバイス起動時」: OPC タグを自動的に生成するタイミングを指定します。オプションの説明は次のとおりです。

- **「起動時に生成しない」**: このオプションを選択した場合、ドライバーは OPC タグをサーバーのタグ空間に追加しません。これはデフォルトの設定です。
- **「起動時に常に生成」**: このオプションを選択した場合、ドライバーはデバイスのタグ情報を評価します。さらに、サーバーが起動するたびに、サーバーのタグ空間にタグを追加します。
- **「最初の起動時に生成」**: このオプションを選択した場合、そのプロジェクトが初めて実行されたときに、ドライバーがデバイスのタグ情報を評価します。さらに、必要に応じて OPC タグをサーバーのタグ空間に追加します。

注記: OPC タグを自動生成するオプションを選択した場合、サーバーのタグスペースに追加されたタグをプロジェクトとともに保存する必要があります。ユーザーは**「ツール」**|**「オプション」**メニューから、自動保存するようプロジェクトを設定できます。

「重複タグ」: 自動タグデータベース生成が有効になっている場合、サーバーが以前に追加したタグや、通信ドライバーが最初に作成した後で追加または修正されたタグを、サーバーがどのように処理するかを設定する必要があります。この設定では、自動生成されてプロジェクト内に現在存在する OPC タグをサーバーがどのように処理するかを制御します。これによって、自動生成されたタグがサーバーに累積することもなくなります。

たとえば、**「起動時に常に生成」**に設定されているサーバーのラックで I/O モジュールを変更した場合、通信ドライバーが新しい I/O モジュールを検出するたびに新しいタグがサーバーに追加されます。古いタグが削除されなかった場合、多数の未使用タグがサーバーのタグ空間内に累積することがあります。以下のオプションがあります。

- **「作成時に削除」**: このオプションを選択した場合、新しいタグが追加される前に、以前にタグ空間に追加されたタグがすべて削除されます。これはデフォルトの設定です。
- **「必要に応じて上書き」**: このオプションを選択した場合、サーバーは通信ドライバーが新しいタグに置き換えているタグだけ除去します。上書きされていないタグはすべてサーバーのタグ空間に残ります。
- **「上書きしない」**: このオプションを選択した場合、サーバーは以前に生成されたタグやサーバーにすでに存在するタグを除去しません。通信ドライバーは完全に新しいタグだけを追加できます。
- **「上書きしない、エラーを記録」**: このオプションには上記のオプションと同じ効果がありますが、タグの上書きが発生した場合にはサーバーのイベントログにエラーメッセージも書き込まれます。

注記: OPC タグの除去は、通信ドライバーによって自動生成されたタグ、および生成されたタグと同じ名前を使用して追加されたタグに影響します。ドライバーによって自動生成されるタグと一致する可能性がある名前を使用してサーバーにタグを追加しないでください。

「親グループ」: このプロパティでは、自動生成されたタグに使用するグループを指定することで、自動生成されたタグと、手動で入力したタグを区別します。グループの名前は最大 256 文字です。この親グループは、自動生成されたすべてのタグが追加されるルートブランチとなります。

「自動生成されたサブグループを許可」: このプロパティでは、自動生成されたタグ用のサブグループをサーバーが自動的に作成するかどうかを制御します。これはデフォルトの設定です。無効になっている場合、サーバーはグループを作成しないで、デバイスのタグをフラットリスト内に生成します。サーバープロジェクトで、生成されたタグには名前としてアドレスの値が付きます。たとえば、生成プロセス中はタグ名は維持されません。
注記: サーバーがタグを生成しているときに、タグに既存のタグと同じ名前が割り当てられた場合、タグ名が重複しないようにするため、番号が自動的に 1 つ増分します。たとえば、生成プロセスによってすでに存在する "AI22" という名前のタグが作成された場合、代わりに "AI23" としてタグが作成されます。

「作成」: 自動生成 OPC タグの作成を開始します。**「タグを作成」**が有効な場合、デバイスの構成が修正されると、ドライバーはタグ変更の可能性についてデバイスを再評価します。システムタグからアクセスできるため、クライアントアプリケーションはタグデータベース作成を開始できます。
注記: 構成がプロジェクトをオフラインで編集する場合、**「タグを作成」**は無効になります。

デバイスのプロパティ - ブロックサイズ

「コイル」

「出力コイル」: コイルは 8 から 2000 ポイント (ビット) の範囲で一度に読み取ることができます。ブロックサイズを大きくすると、1 回の要求でより多くのポイントがデバイスから読み取られます。デバイス内の連続しない位置からデータを読み取る必要がある場合、ブロックサイズを小さくすることができます。

「入力コイル」: コイルは 8 から 2000 ポイント (ビット) の範囲で一度に読み取ることができます。ブロックサイズを大きくすると、1 回の要求でより多くのポイントがデバイスから読み取られます。デバイス内の連続しない位置からデータを読み取る必要がある場合、ブロックサイズを小さくすることができます。

「レジスタ」

「内部レジスタ」: レジスタは 1 から 100 個の位置 (Word) から一度に読み取ることができます。ブロックサイズを大きくすると、1 回の要求でより多くのレジスタ値がデバイスから読み取られます。デバイス内の連続しない位置からデータを読み取る必要がある場合、ブロックサイズを小さくすることができます。

「保持レジスタ」: レジスタは 1 から 100 個の位置 (Word) から一度に読み取ることができます。ブロックサイズを大きくすると、1 回の要求でより多くのレジスタ値がデバイスから読み取られます。デバイス内の連続しない位置からデータを読み取る必要がある場合、ブロックサイズを小さくすることができます。

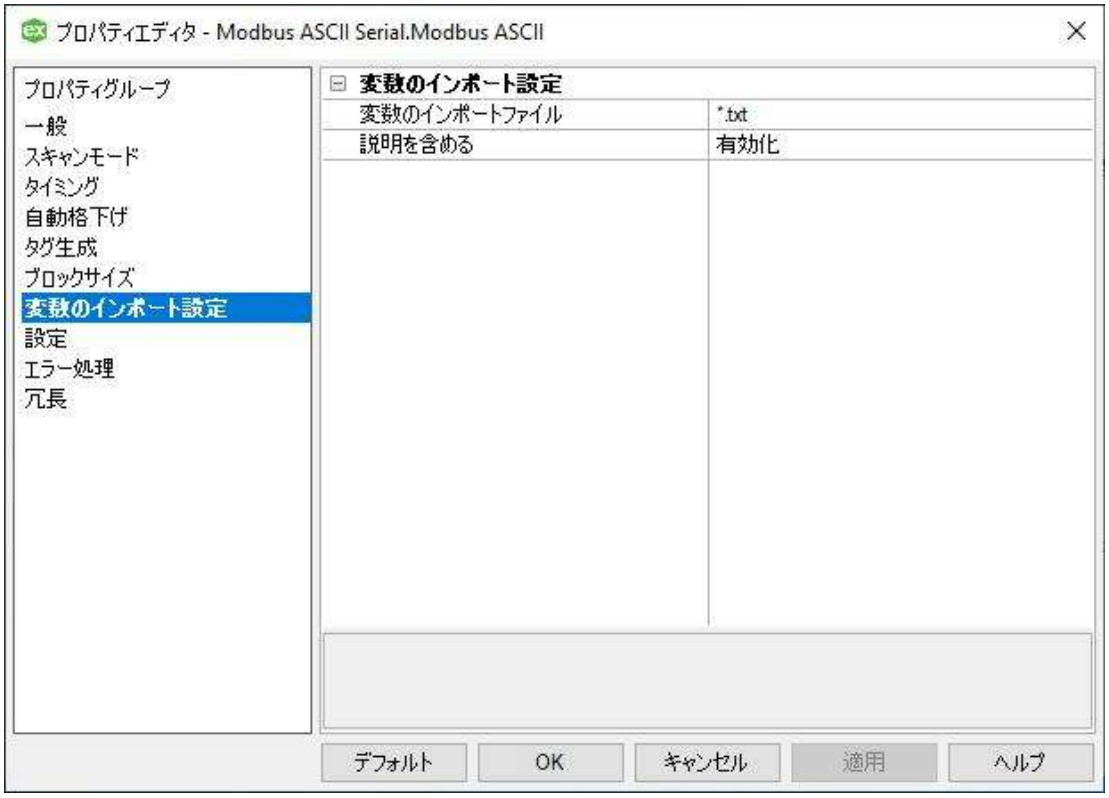
警告: レジスタのブロックサイズとして 120 より大きい値が設定され、任意のタグに 32 ビットまたは 64 ビットデータ型が使用されている場合、エラーが発生することがあります。このエラーが発生しないようにするには、ブロックサイズの値を 120 に減らしてください。

「ブロックサイズ」

「文字列のブロック読み取り」: このオプションを有効にした場合、通常は個別に読み取る文字列タグをブロックで読み取ります。このオプションを有効にした場合、文字列タグは選択したブロックサイズに応じてグループ化されます。ブロック読み取りは Modbus モデルの文字列タグに対してのみ実行できます。

デバイスのプロパティ - 変数のインポート設定

Modbusドライバー向けCSVファイルの詳細については、Modbusドライバー向けCSVファイルの作成を参照してください。



「変数のインポートファイル」: ドライバーが自動タグ生成に使用するセミコロン区切りのテキストファイルの正確な場所と名前を指定します。さまざまなアプリケーションから変数インポートファイルを作成できます。

「説明を含める」: 有効な場合、タグの説明がインポートされます (ファイル内に存在する場合)。デフォルト設定では有効になっています。

自動タグデータベース生成機能の設定および変数インポートファイルの作成については、自動タグデータベース生成を参照してください。

デバイスのプロパティ - 設定

データアクセス

「ゼロベースアドレス指定」: デバイスのアドレス番号付けの規則で番号がゼロではなく 1 で開始する場合、デバイスのプロパティを定義する際にこれを指定できます。デフォルトでは、Modbus デバイスと通信するためにフレームを構築する場合はユーザーが入力したアドレスから 1 が引かれます。デバイスがこの規則に従わない場合、「ゼロベースアドレス指定」を無効にできます。デフォルトの動作 (ゼロベース) は Modicon PLC の規則に従います。

「ゼロベースのビットアドレス指定」: Word 内のビットをブールとして参照可能なメモリタイプの場合、アドレス指定の表記は*<address><bit>* であり、ここで *<bit>* は Word 内のビット番号を表します。レジスタ内のアドレス指定によって、ある Word 内の 1 ビットを 2 つの方法によってアドレス指定できます。レジスタ内のゼロベースのビットアドレス指定は、最初のビットが 0 で始まることを意味します。レジスタ内の 1 ベースのアドレス指定は、最初のビットが 1 で始まることを意味します。Word データ型の場合、ゼロビットアドレス指定のビット範囲は 0-15 であるのに対し、1 ベースのビットアドレス指定の範囲は 1-16 です。

「保持レジスタのビット書き込み」: 保持レジスタ内のビット位置に書き込む際、ドライバーは対象のビットのみを修正する必要があります。一部のデバイスはレジスタ内の 1 ビットを操作するコマンドをサポートしています (ファンクションコード 0x16 (16 進) または 22 (10 進))。デバイスがこの機能をサポートしていない場合、ドライバーは 1 ビットだけが変更されるように読み取り/修正/書き込み操作を実行する必要があります。デバイスが保持レジスタのビットアクセスをサポートしている場合に有効にします。デフォルトでは無効に設定されています。この設定が有効になっている場合、ドライバーは「Modbus 関数 06」の設定に関係なく、レジスタへの書き込みにファンクションコード 0x16 を使用します。この設定が選択されていない場合、ドライバーは「Modbus 関数 06」の設定に応じて、ファンクションコード 0x06 または 0x10 を使用します。

注記: 「Modbus バイトオーダー」が選択されていない場合、このコマンドで送信されるマスクのバイトオーダーは Intel バイトオーダーになります。

「Modbus 関数 06」: Modbus ドライバーは、2 つの Modbus プロトコルファンクションを使用して保持レジスタのデータをターゲットデバイスに書き込むことができます。ほとんどの場合、このドライバーは書き込み対象のレジスタの数に基づいてこの 2 つのファンクションを切り替えます。単一の 16 ビットレジスタに書き込む場合、このドライバーは通常、Modbus ファンクション 06 を使用します。32 ビット値を 2 つのレジスタに書き込む場合、このドライバーは Modbus ファンクション 16 を使用します。標準の Modicon PLC では、このどちらのファンクションを使用しても問題ありません。ただし、Modbus プロトコルを使用する多くのサードパーティデバイスとこれらのデバイスの多くは、書き込み対象のレジスタの数に関係なく、保持レジスタへの書き込みに Modbus ファンクション 16 の使用のみをサポートしています。「ファンクション 06 の使用」を使用することで、必要な場合にドライバーに Modbus ファンクション 16 のみを使用させることができます。この選択はデフォ

ルトで有効になっています。これにより、ドライバーは必要に応じて 06 と 16 を切り替えることができます。デバイスが Modbus ファンクション 16 のみを使用してすべての書き込みを行う必要がある場合、この選択を無効にします。

注記: Word 内のビットの書き込みでは、「保持レジスタのビット書き込み」プロパティがこのプロパティ (Modbus 関数 06) よりも優先されます。「保持レジスタのビット書き込み」が選択されている場合、このプロパティの選択にかかわらず、ファンクションコード 0x16 が使用されます。「保持レジスタのビット書き込み」が選択されていない場合、Word 内のビットの書き込みには、このプロパティの選択に応じてファンクションコード 0x06 または 0x10 が使用されます。

使用「関数 05 の使用」: Modbus ドライバーは、2 つの Modbus プロトコルファンクションを使用して出力コイルのデータをターゲットデバイスに書き込むことができます。ほとんどの場合、このドライバーは書き込み対象のコイルの数に基づいてこの 2 つのファンクションを切り替えます。単一のコイルに書き込む場合、このドライバーは Modbus ファンクション 05 を使用します。コイルの配列に書き込む場合、このドライバーは Modbus ファンクション 15 を使用します。標準の Modicon PLC では、このどちらのファンクションを使用しても問題ありません。ただし、Modbus プロトコルを使用する多くのサードパーティデバイスとこれらのデバイスの多くは、書き込み対象のコイルの数に関係なく、出力コイルへの書き込みに Modbus ファンクション 15 の使用のみをサポートしています。「Modbus 関数 05」を使用することで、必要な場合にドライバーに Modbus ファンクション 15 のみを使用させることができます。この選択はデフォルトで有効になっています。これにより、ドライバーは必要に応じて 05 と 15 を切り替えることができます。デバイスが Modbus ファンクション 15 のみを使用してすべての書き込みを行う必要がある場合、この選択を無効にします。

データエンコーディング

「Modbus バイトオーダー」: この選択を使用することで、ドライバーのバイトオーダーをデフォルトの Modbus バイトオーダーから Intel バイトオーダーに変更できます。「Modbus バイトオーダー」では各レジスタ/16 ビット値のデータエンコーディングを設定します。この選択はデフォルトで有効になっており、Modbus 対応デバイスでは標準の設定です。デバイスが Intel バイトオーダーを使用する場合、この選択を無効にすることで、Modbus ドライバーは Intel フォーマットのデータを適切に読み取ることができます。

「最初の Word を下位とする」: Modbus デバイスでは 32 ビットデータ型に 2 つの連続するレジスタアドレスが使用されます。ドライバーが最初の Word を 32 ビット値および 64 ビット値の各 DWord の下位 Word とするか上位 Word とするかを指定できます。デフォルトの「最初の Word を下位とする」は、Modicon Modsoft プログラミングソフトウェアの規則に従います。

「最初の DWord を下位とする」: Modbus デバイスでは 64 ビットデータ型に 4 つの連続するレジスタアドレスが使用されます。ドライバーが最初の DWord を 64 ビット値の下位 DWord とするか上位 DWord とするかを指定できます。デフォルトの「最初の DWord を下位とする」は、64 ビットデータ型のデフォルトの規則に従います。

「Modicon ビットオーダー」: 有効な場合、ドライバーはレジスタに対する読み書きの際にビットオーダーを反転して Modicon Modsoft プログラミングソフトウェアの規則に従います。たとえば、このオプションが有効になっている場合、アドレス 40001.0/1 への書き込みはこのデバイスのビット 15/16 に影響します。このオプションはデフォルトで無効になっています。

注記: 次の例では、そのドライバーに設定されているレジスタ内のビットアドレス指定がゼロベースか 1 ベースかに応じて、1 から 16 番目のビットは 0-15 ビットまたは 1-16 ビットを表します。

MSB = 最上位ビット
LSB = 最下位ビット

「Modicon ビットオーダー」が有効

MSB															LSB
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

「Modicon ビットオーダー」が無効

MSB															LSB
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

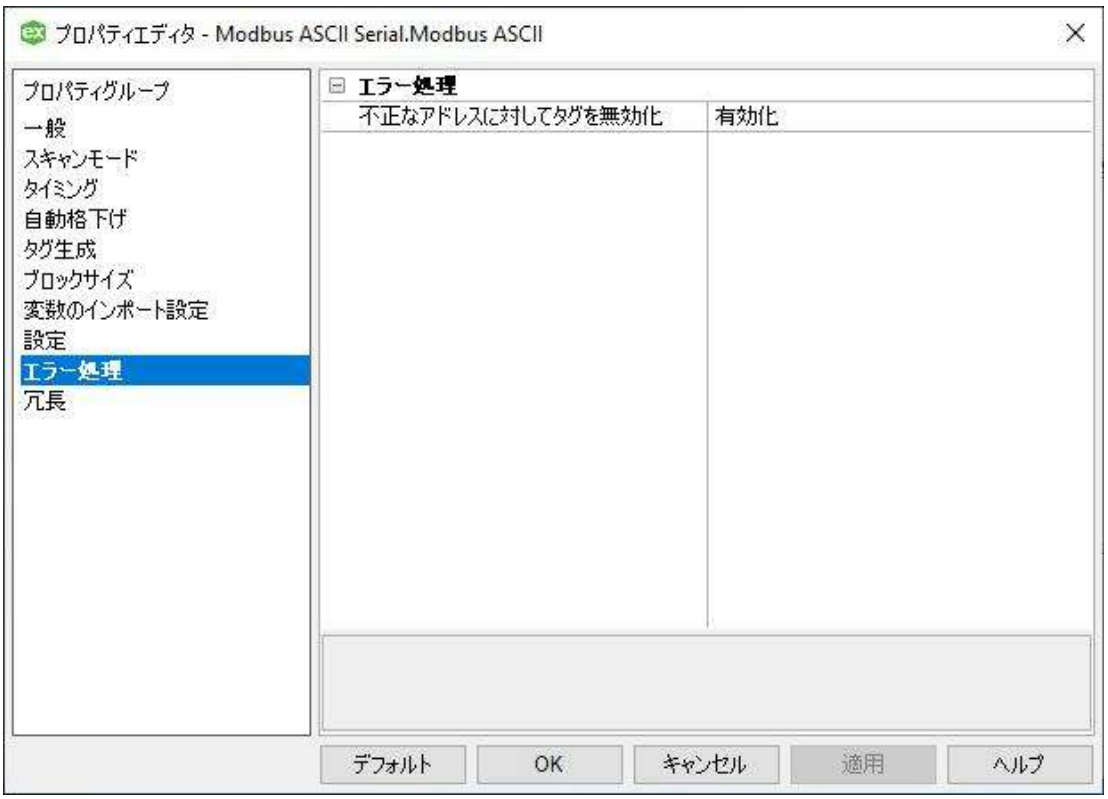
データ型	Modbus バイトオーダー	最初の Word を下位とする	最初の DWord を下位とする
Word、Short、BCD	適用可能	なし	なし
Float、DWord、Long、LBCD	適用可能	適用可能	なし

Double	適用可能	適用可能	適用可能

「データエンコーディング」グループのオプション	データエンコーディング	
「Modbus バイトオーダー」- 有効	上位バイト (15..8)	下位バイト (7..0)
「Modbus バイトオーダー」- 無効	下位バイト (7..0)	上位バイト (15..8)
「最初の Word を下位とする」- 無効	上位 Word (31..16) 64 ビットデータ型での DWord の上位 Word (63..48)	下位 Word (15..0) 64 ビットデータ型での DWord の下位 Word (47..32)
「最初の Word を下位とする」- 有効	下位 Word (15..0) 64 ビットデータ型での DWord の下位 Word (47..32)	上位 Word (31..16) 64 ビットデータ型での DWord の上位 Word (63..48)
「最初の DWord を下位とする」- 無効	上位 DWord (63..32)	下位 DWord (31..0)
「最初の DWord を下位とする」- 有効	下位 DWord (31..0)	上位 DWord (63..32)

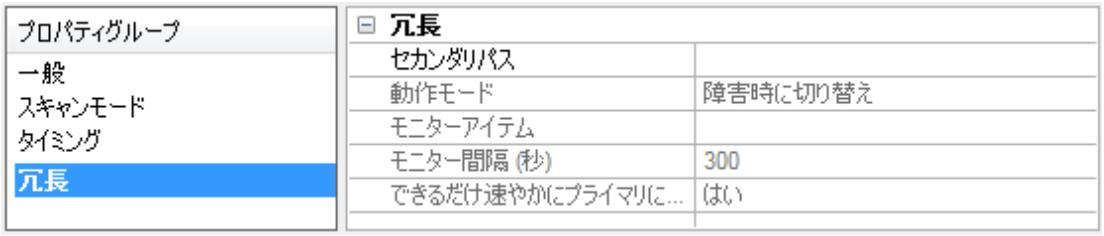
大部分の Modbus デバイスではデフォルト設定を使用するのが適していますが、個々のデバイスのドキュメントでデータエンコーディングオプションの正しい設定を調べることができます。

デバイスのプロパティ - エラー処理



「不正なアドレスでタグを無効化」: デバイスがブロックの読み取りに応答して Modbus 例外コード 2 (不正なアドレス) または 3 (ポイント数などの不正なデータ) を返した場合、ドライバーはエラーがあるブロックのポーリングを停止するか、ブロックのポーリングを続行できます。不正なアドレスのエラーが発生した場合にポーリングを停止するには、「有効化」を選択します。そのデータブロックを引き続きポーリングするには、「無効化」を選択します。非アクティブなブロックをアクティブ化するためにサーバーを再起動する必要はありません。デフォルト設定では有効になっています。

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。

自動タグデータベース生成

Modbus ASCII シリアルドライバーでは自動タグデータベース生成機能が利用されます。これにより、ドライバーはデバイスのラダープログラムによって使用されるデータポイントにアクセスするタグを自動的に作成できます。タグデータベースの構築に必要な情報をデバイスに対して照会可能な場合もありますが、このドライバーは代わりに変数インポートファイルを使用する必要があります。変数インポートファイルは Concept や ProWORX などのデバイスプログラミングアプリケーションを使用して生成できます。

変数インポートファイルの作成

このインポートファイルは、Concept デバイスプログラミングアプリケーションのデフォルトのエクスポートファイルフォーマットであるセミコロン区切りの .TXT フォーマットでなければなりません。ProWORX プログラミングアプリケーションはこのフォーマットで変数データをエクスポートできます。

Concept および ProWORX からの変数インポートファイルの作成方法については、技術情報「Modbus ドライバー向け CSV ファイルの作成」を参照してください。

サーバー構成

アプリケーションのニーズに合わせて自動タグデータベース生成機能をカスタマイズできます。プライマリ制御オプションは、デバイスウィザードのデータベース作成ステップの実行中に指定します。または、後で指定する場合は、デバイスを選択してから**「プロパティ」**|**「タグを作成」**の順に選択します。

詳細については、サーバーのヘルプドキュメントを参照してください。

このドライバーは、自動タグデータベース生成をサポートするすべてのドライバーに共通する基本設定に加え、特別なプロパティを必要とします。このような特別なプロパティとして変数インポートファイルの名前や場所を指定する必要があります。この情報は、デバイスウィザードの「変数のインポート設定」ステップの実行中に指定します。または、後で指定する場合は、デバイスを選択してから**「プロパティ」**|**「変数のインポート設定」**の順に選択します。

*詳細については、**変数のインポート設定**を参照してください。*

操作

構成に応じて、タグ生成はサーバープロジェクトが開始したときに自動的に開始するか、後から手動で開始できます。「イベントログ」には、タグ生成プロセスの開始時刻、変数インポートファイルの処理中に発生したエラー、このプロセスの完了時刻が示されます。

データ型の説明

データ型	説明
Boolean	1 ビット
Word	符号なし 16 ビット値 ビット 0 が下位ビット ビット 15 が上位ビット
Short	符号付き 16 ビット値 ビット 0 が下位ビット ビット 14 が上位ビット ビット 15 が符号ビット
DWord	符号なし 32 ビット値 ビット 0 が下位ビット ビット 31 が上位ビット
Long	符号付き 32 ビット値 ビット 0 が下位ビット ビット 30 が上位ビット ビット 31 が符号ビット
BCD	2 バイトパックされた BCD 値の範囲は 0-9999 です。この範囲外の値には動作が定義されていません。
LBCD	4 バイトパックされた BCD 値の範囲は 0-99999999 です。この範囲外の値には動作が定義されていません。
String	Null 終端 ASCII 文字列 Modbus モデルでサポートされ、バイトオーダーを HiLo/LoHi から選択できます。
Double*	64 ビット浮動小数点値 ドライバーは最後の 2 つのレジスタを上位 DWord、最初の 2 つのレジスタを下位 DWord とすることで、連続する 4 つのレジスタを倍精度値として解釈します。
Double の例	レジスタ 40001 が Double として指定されている場合、レジスタ 40001 のビット 0 は 64 ビットデータ型のビット 0 になり、レジスタ 40004 のビット 15 は 64 ビットデータ型のビット 63 になります。
Float*	32 ビット浮動小数点値 ドライバーは最後のレジスタを上位 Word、最初のレジスタを下位 Word とすることで、連続する 2 つのレジスタを単精度値として解釈します。
Float の例	レジスタ 40001 が Float として指定されている場合、レジスタ 40001 のビット 0 は 32 ビットデータ型のビット 0 になり、レジスタ 40002 のビット 15 は 32 ビットデータ型のビット 31 になります。

*上記の説明は、64 ビットデータ型では最初の DWord を下位とし、32 ビットデータ型では最初の Word を下位とするデフォルト設定のデータ処理を前提としています。

アドレスの説明

アドレスの仕様は使用されているモデルによって異なります。対象のモデルのアドレス情報を取得するには、次のリストからリンクを選択してください。

Modbus ASCII のアドレス指定
フローコンピュータのアドレス指定
フローオートメーションのアドレス指定

Modbus ASCII のアドレス指定

5 桁のアドレス指定と 6 桁のアドレス指定

Modbus のアドレス指定では、アドレスの最初の桁はプライマリテーブルを示します。以降の桁はデバイスのデータアイテムを表します。最大値は 2 バイトの符号なし整数 (65,535) です。アドレステーブルとアイテム全体を表すのに 6 桁が必要です。このため、デバイスのマニュアルで 0xxxx、1xxxx、3xxxx、4xxxx として指定されているアドレスは、Modbus タグのアドレスフィールドに適用されると追加のゼロが 1 つパディングされます。

プライマリテーブル	説明
0	出力コイル
1	入力コイル
3	内部レジスタ
4	保持レジスタ

Modbus ASCII のアドレス指定

動的に定義されるタグのデフォルトのデータ型を**太字**で示しています。

注意事項と制約については、パックコイルタグ、文字列のサポート、および配列のサポートを参照してください。

アドレス	範囲	データ型	アクセス	ファンクションコード*
出力コイル	000001-065536 000001#1-065521#16	**Boolean** Word (パックコイルタグ)	読み取り/書き込み	01, 05, 15**
入力コイル	100001-165536 100001#1-165521#16	**Boolean** Word (パックコイルタグ)	読み取り専用	02**
内部レジスタ	300001-365536 300001-365535 300001-365533 3xxxxx.0/1-3xxxxx.15/16***	**Word**、Short、BCD Float、DWord、Long、LBCD Double Boolean	読み取り専用	04
String の内部レジスタ、HiLo バイトオーダー	300001.2H-365536.240H ピリオドの後ろのビット番号は 2 から 240 バイトの範囲の文字列長を示します。	**String**	読み取り専用	04
String の内部レジスタ、LoHi バイトオーダー	300001.2L-365536.240L ピリオドの後ろのビット番号は 2 から 240 バイトの範囲の文字列長を示します。	**String**	読み取り専用	04
保持レジスタ	400001-465536 400001-465535 400001-465533 4xxxxx.0/1-4xxxxx.15/16***	**Word**、Short、BCD Float、DWord、Long、LBCD Double Boolean	読み取り/書き込み	03, 06, 16 03, 06, 16, 22

アドレス	範囲	データ型	アクセス	ファンクションコード*
String の保持レジスタ、HiLo バイトオーダー	400001.240H-465536.2H ピリオドの後ろのビット番号は文字列長 (2 から 240 バイトの範囲) を示します。	String	読み取り/書き込み	03, 16
String の保持レジスタ、LoHi バイトオーダー	400001.2L-465536.240L ピリオドの後ろのビット番号は文字列長 (2 から 240 バイトの範囲) を示します。	String	読み取り/書き込み	03, 16

*サポートされているファンクションコードは 10 進数で表示されます。詳細については、*ファンクションコードの説明*を参照してください。
***詳細については、パックコイルタグを参照してください。*
****詳細については、設定のサブトピック「ゼロベースのビットアドレス指定の使用」を参照してください。*

書き込み専用アクセス

"W40001" などのように、アドレスの先頭に "W" を付けることによって、すべての読み取り/書き込み可能アドレスを書き込み専用として設定でき、これによってドライバーはレジスタの指定したアドレスを読み取れなくなります。クライアントが書き込み専用タグを読み取ろうとすると、指定したアドレスへの最後に成功した書き込みの値が取得されます。成功した書き込みがない場合、クライアントは数値/文字列値の初期値である 0/NULL を受信します。

警告: 書き込み専用タグのクライアントアクセス権限を読み取り専用に設定した場合、これらのタグへの書き込みは失敗し、クライアントは数値/文字列値に対して必ず 0 / NULL を受信します。

パックコイルタグ

パックタイプのコイルアドレスでは、複数の連続するコイルにアナログ値としてアクセスできます。この機能は Modbus ASCII モデルでのみ使用できます。有効な唯一のデータ型が Word です。構文は以下のとおりです。

出力コイル: 0xxxxx#nn Word 読み取り/書き込み
入力コイル: 1xxxxx#nn Word 読み取り専用

ここで、xxxxx は 1 つ目のコイルのアドレスであり、nn はアナログ値にパックされるコイルの数 (1-16) です。

開始アドレスがアナログ値の LSB (最下位ビット) となるビットオーダーになります。

文字列のサポート

Modbus モデルでは保持レジスタメモリを ASCII 文字列として読み書きできます。文字列データに保持レジスタを使用している場合、各レジスタに 2 バイトの ASCII データが格納されます。文字列を定義する際に、そのレジスタにおける ASCII データの順序を選択できます。文字列の長さは 2 から 240 バイトの範囲で指定でき、ビット番号の位置に入力します。この長さは偶数として入力する必要があります。バイトオーダーはアドレスの末尾に "H" または "L" を付けることによって指定します。

Modbus モデルの文字列タグに対してブロック読み取りを実行する方法については、*ブロックサイズ*を参照してください。

文字列の例

1. 40200 で開始し、長さが 100 バイト、HiLo バイトオーダーの文字列をアドレス指定するには、40200.100H と入力します。
2. 40500 で開始し、長さが 78 バイト、LoHi バイトオーダーの文字列をアドレス指定するには、40500.78L と入力します。

注記: デバイスで許可される書き込み要求の最大サイズによって文字列の長さが制限されることがあります。文字列タグを使用しているときに、「デバイス <デバイス> のアドレス <アドレス> に書き込めません: デバイスは例外コード 3 を返しました」というエラーメッセージを受信した場合、文字列の長さがそのデバイスに適していませんでした。可能な場合、文字列を短くしてみてください。

通常のアドレスの例

1. 255 番目の出力コイルのアドレスは、10 進アドレス指定を使用した場合には '0255' となります。
2. 一部のドキュメントでは Modbus のアドレスはファンクションコードと位置によって示されています。たとえば、ファンクションコード 3、位置 2000 のアドレスは '42000' となります (先頭の '4' は保持レジスタ、つまりファンクションコード 3 を表します)。
3. 一部のドキュメントでは Modbus のアドレスはファンクションコードと位置によって示されています。たとえば、ファンクションコード 5、位置 100 を設定するとアドレスは '0100' となります (先頭の '0' は出力コイル、つまりファンクションコード 5 を表します)。このアドレスに 1 を書き込むとコイルが設定され、0 を書き込むとコイルがリセットされます。

配列のサポート

内部レジスタと保持レジスタの位置では Boolean と String 以外のすべてのデータ型で配列がサポートされています。入力コイルと出力コイルでも配列がサポートされます (Boolean データ型)。配列のアドレス指定には 2 つの方法があります。例では保持レジスタの位置が使用されています。

4xxxx [行数] [列数]
4xxxx [列数] - この方法では行数が 1 であるものと見なされます

配列の場合、行数と列数を掛けた値が、デバイスのレジスタ/コイルタイプに割り当てられているブロックサイズを超えてはなりません。32 ビットデータ型のレジスタ配列の場合、行数と列数を掛けた値を 2 倍にした値がブロックサイズを超えてはなりません。

ファンクションコードの説明

10 進	16 進	説明
01	0x01	コイルのステータスを読み取り
02	0x02	入力ステータスを読み取り
03	0x03	保持レジスタを読み取り
04	0x04	内部レジスタを読み取り
05	0x05	単一コイルを適用
06	0x06	単一レジスタをプリセット
15	0x0F	複数コイルを適用
16	0x10	複数レジスタをプリセット
22	0x16	レジスタへのマスク書き込み

フローコンピュータのアドレス指定

動的に定義されるタグのデフォルトのデータ型を**太字**で示しています。

アドレス	範囲	データ型	アクセス
出力コイル	000001-065536	Boolean	読み取り/書き込み
入力コイル	100001-165536	Boolean	読み取り専用
内部レジスタ	300001-365536 300001-365535	**Word**、Short、BCD Float、DWord、Long、LBCD	読み取り専用
保持レジスタ	400001-465536 400001-465535	**Word**、Short、BCD*、 Float、DWord、Long、LBCD	読み取り/書き込み
フローコンピュータのレジスタ	405000-406800 407000-407800	**Long**、DWord、LBCD **Float**、Long、DWord	読み取り/書き込み

*****アドレス範囲 405000 から 406800 と 407000 から 407800 は 32 ビットレジスタです。405000 から 406800 の範囲のアドレスではデフォルトのデータ型として Long が使用されます。407000 から 407800 の範囲のアドレスではデフォルトのデータ型として Float が使用されます。これらのアドレスレジスタは 32 ビットなので、Float、DWord、Long、LBCD データ型のみを使用できます。これらの特別なアドレス範囲では配列を使用できません。

配列

内部レジスタと保持レジスタの位置では Boolean 以外のすべてのデータ型で配列がサポートされています。配列のアドレス指定には 2 つの方法があります。例では保持レジスタの位置が使用されています。

4xxxx [行数] [列数]
4xxxx [列数] - この方法では行数が 1 であるものと見なされます

行数と列数を掛けた値が、デバイスのレジスタタイプに割り当てられているブロックサイズを超えてはなりません。32 ビットデータ型の配列の場合、行数と列数を掛けた値を 2 倍にした値がブロックサイズを超えてはなりません。

フローオートメーションのアドレス指定

動的に定義されるタグのデフォルトのデータ型を**太字**で示しています。

アドレス	範囲	データ型	アクセス
フローコンピュータのレジスタ	40001-465535	**Float**	読み取り/書き込み

フローオートメーションのフローコンピュータはすべてのデータを 32 ビット浮動小数点値として処理します。デバイスの保持レジスタ空間内のすべてのアドレスが 32 ビット浮動小数点数として読み取られます。フローオートメーションのマニュアルのカスタムレポートのセクションに、フローオートメーション制御の完全なメモリマップが掲載されています。

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リングとソートについては、OPC 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。

ブロックに不良アドレスがあります。| ブロック範囲 = <アドレス> から <アドレス>。

エラータイプ:
エラー

考えられる原因:
指定されたデバイスに存在しない位置を参照しようとしました。

解決策:
デバイスの指定された範囲のアドレスに割り当てられたタグを確認し、無効な位置を参照するタグを削除してください。

不良配列。| 配列範囲 = <開始> ～ <終了>。

エラータイプ:
エラー

考えられる原因:
アドレス空間の末端を超えてアドレスの配列が定義されています。

解決策:
デバイスのメモリ空間のサイズを確認し、配列長を適切に再定義してください。

タグデータベースのインポート用のファイルを開くときにエラーが発生しました。| OS エラー = '<エラー>'。

エラータイプ:
エラー

受信したブロック長が予想ブロック長と一致しません。| 受信した長さ = <数値> (バイト)、予想される長さ = <数値> (バイト)。

エラータイプ:
警告

デバイスに対するブロック要求で例外が返されました。| ブロック範囲 = <アドレス> から <アドレス>、例外 = <コード>。

エラータイプ:
警告

考えられる原因:
デバイスとの通信に成功しましたが、デバイスから問題が報告されました。

解決策:
表示されたエラーコードの詳細については、デバイスに付属のドキュメントを参照してください。

関連項目:
Modbus 例外コード

デバイスのアドレスに書き込めません。デバイスは例外を返しました。| アドレス = '<アドレス>'、例外 = <コード>。

エラータイプ:
警告

考えられる原因:
デバイスとの通信に成功しましたが、デバイスから問題が報告されました。

解決策:
表示されたエラーコードの詳細については、デバイスに付属のドキュメントを参照してください。

関連項目:
Modbus 例外コード

デバイスのアドレスから読み取れません。デバイスは例外を返しました。| アドレス = '<アドレス>'、例外 = <コード>。

エラータイプ:
警告

考えられる原因:
デバイスとの通信に成功しましたが、デバイスから問題が報告されました。

解決策:
表示されたエラーコードの詳細については、デバイスに付属のドキュメントを参照してください。

関連項目:
Modbus 例外コード

メモリリソース量の低下によりタグインポートが失敗しました。

エラータイプ:
警告

考えられる原因:
ドライバーは変数インポートファイルの処理に必要なメモリを割り当てることができませんでした。

解決策:
不要なアプリケーションをシャットダウンしてから、もう一度試してください。

タグのインポート中にファイル例外が発生しました。

エラータイプ:
警告

考えられる原因:
変数インポートファイルを読み取れませんでした。

解決策:
変数インポートファイルを修正または再生成してください。

インポートファイルのレコードの解析でエラーが発生しました。| レコード番号 = <数値>、フィールド = <数値>。

エラータイプ:
警告

考えられる原因:

変数インポートファイルの指定されたフィールドが予想より長いか無効なため、解析できませんでした。

解決策:
変数インポートファイルを編集してフィールドを修正してください。

インポートファイルのレコードの説明が切り詰められました。| レコード番号 = <数値>。

エラータイプ:
警告

考えられる原因:
指定されたレコード内のタグの説明が長すぎます。

解決策:
説明が必要に応じて切り詰められます。このエラーが今後発生しないようにするには、変数インポートファイルを編集して、説明を短くしてください。

インポートされたタグ名が無効のため変更されました。| タグ名 = '<タグ>'、変更後のタグ名 = '<タグ>'。

エラータイプ:
警告

考えられる原因:
変数インポートファイル内のタグ名に無効な文字が含まれていました。

解決策:
変数インポートファイルに基づいて有効な名前が構築されました。今後このエラーが発生しないようにして名前の一貫性を維持するには、エクスポートされた変数の名前を変更してください。

データ型がサポートされていないため、タグをインポートできませんでした。| タグ名 = '<タグ>'、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:
変数インポートファイルで指定されたデータ型は、このドライバーでサポートされている型ではありません。

解決策:
インポートファイルで指定されているデータ型を、サポートされている型に変更してください。構造体の変数である場合、ファイルを編集して構造体に必要な各タグを定義するか、サーバーで必要なタグを設定してください。

関連項目:
Concept からの変数のエクスポート

タグデータベースをインポートしています。| ソースファイル = '<パス>'。

エラータイプ:
情報

Modbus 例外コード

以下のデータは Modbus Application Protocol Specifications ドキュメントからのものです。

コード 10 進 /16 進	名前	意味
01/0x01	ILLEGAL FUNCTION (不正なファンクション)	クエリーで受信したファンクションコードを、サーバーに対する操作として実行することはできません。このファンクションコードは新しいデバイスにだけ適用できるか、選択したユニットに実装されていないことが原因である可能性があります。また、サーバーが、このタイプの要求を処理する状態になっていない可能性もあります (レジスタ値を返す必要があるにもかかわらず、サーバーがそのように設定されていない場合など)。
02/0x02	ILLEGAL DATA ADDRESS (不正なデータアドレス)	クエリーで受信したデータアドレスを、サーバーに対するアドレスとして使用することはできません。具体的には、参照番号と転送長さの組み合わせが無効です。レジスタが 100 個あるコントローラの場合、オフセット 96 と長さ 4 の要求では成功します。オフセット 96 と長さ 5 の要求では例外 02 が生成されます。
03/0x03	ILLEGAL DATA VALUE (不正なデータ値)	クエリーデータフィールドに含まれている値を、サーバーに対する値として使用することはできません。これは、示された長さが正しくないなど、複合型要求の残りの構造体に誤りがあることを示しています。Modbus プロトコルでは個々のレジスタのそれぞれの値の有意性は認識されないので、これはレジスタのストレージにサブミットされたデータアイテムの値がアプリケーションプログラムでの予想の範囲外であることを必ずしも意味しません。
04/0x04	SERVER DEVICE FAILURE (サーバーデバイスの障害)	サーバーが要求された操作を実行しようとしたときに回復不可能なエラーが発生しました。
05/0x05	ACKNOWLEDGE	サーバーは要求を受け入れて処理していますが、処理が完了するまで時間がかかります。クライアントでタイムアウトエラーが発生しないようにするために、この応答が返されます。クライアントは次にプログラム完了ポーリングメッセージを送信して、処理が完了したかどうかを判断します。
06/0x06	SERVER DEVICE BUSY (サーバーデバイスがビジー状態)	サーバーは、時間がかかるプログラムコマンドを処理しています。クライアントは、サーバーによる処理の完了後にメッセージを再送信する必要があります。
07/0x07	NEGATIVE ACKNOWLEDGE (否定応答)	サーバーは、クエリーで受信したプログラムファンクションを実行できません。このコードはファンクションコード 13 または 14 (10 進) を使用したプログラミング要求が成功しなかった場合に返されます。クライアントは、サーバーに対して診断情報またはエラー情報を要求する必要があります。
08/0x08	MEMORY PARITY ERROR (メモリパリティエラー)	サーバーが拡張メモリを読み取ろうとしたときに、メモリ内でパリティエラーが検出されました。クライアントはこの要求を再試行できますが、サーバーデバイス上でサービスが必要になる場合があります。
10/0x0A	GATEWAY PATH UNAVAILABLE (ゲートウェイパスを使用できません)	ゲートウェイが使用されている場合、ゲートウェイが要求を処理するために入力ポートから出力ポートへの内部通信パスを割り当てることができなかったことを示します。これは通常、ゲートウェイの設定に誤りがあるかオーバーロードされていることを意味します。
11/0x0B	GATEWAY TARGET DEVICE FAILED TO RESPOND (ゲートウェイのターゲットデバイスが応答しませんでした)	ゲートウェイが使用されている場合、ターゲットデバイスから応答がなかったことを示します。これは通常、デバイスがネットワーク上に存在しないことを意味します。

注記: このドライバーでは、「サーバー」と「非送信請求」という用語は同じ意味になります。

エラーマスクの定義

B = ハードウェアの故障を検出
F = フレーミングエラー
E = I/O エラー
O = 文字バッファオーバーラン

R = RX バッファオーバーラン
P = 受信バイトパリティエラー
T = TX バッファフル

索引

F

I

L

M

O

P

R

S

T

W

あ

い

え

お

き

く

こ

さ

し

す

せ

た

ち

て

と

な

ね

は

ひ

ふ

ほ

め

も

れ

漢字