Allen-Bradley Micro800 Ethernet Driver

Inhaltsverzeichnis

Allen-Bradley Micro800 Ethernet Driver

Hilfeversion 1.028

INHALT

Übersicht

Allen-Bradley Micro800 Ethernet Driver bietet eine zuverlässige Möglichkeit, Micro800-Ethernet-Controller von Allen-Bradley mit OPC-Client-Anwendungen, u.a. HMI, SCADA, Historian, MES, ERP und zahlreichen benutzerdefinierten Anwendungen, zu verbinden.

Setup

Unterstützte Geräte

Micro850 über eingebetteten Ethernet-Port.

Kommunikationsprotokoll

Ethernet/IP (CIP über Ethernet) mit TCP/IP.

Maximale Anzahl von Kanälen und Geräten

Die maximale Anzahl unterstützter Kanäle ist 256. Die maximale Anzahl von unterstützten Geräten pro Kanal liegt bei 1024.

Kanal-Setup

Kanal-Setup schließt Konfiguration der folgenden Eigenschaftsgruppen ein:

Allgemein
Ethernet-Kommunikation
Schreiboptimierungen
Erweitert
Kommunikationsserialisierung

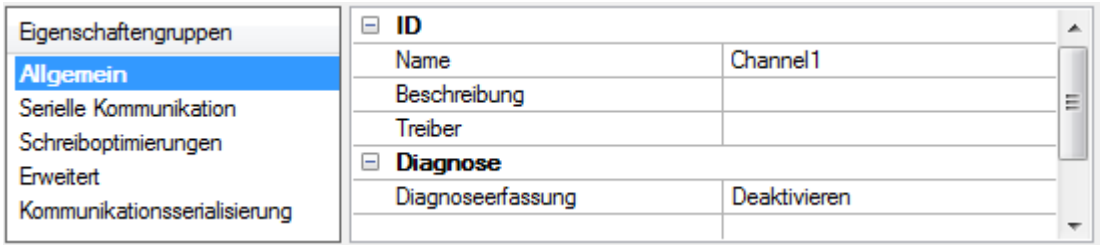
Geräte-Setup

Geräte-Setup schließt Konfiguration der folgenden Eigenschaftsgruppen ein:

Allgemein
Scan-Modus
Zeitvorgabe
Automatische Herabstufung
Kommunikationsparameter
Optionen
Redundanz

Kanaleigenschaften – Allgemein

Dieser Server unterstützt die Verwendung von gleichzeitigen Mehrfachkommunikationstreibern. Jedes Protokoll oder jeder Treiber, das/der in einem Serverprojekt verwendet wird, wird als Kanal bezeichnet. Ein Serverprojekt besteht unter Umständen aus vielen Kanälen mit demselben Kommunikationstreiber oder mit eindeutigen Kommunikationstreibern. Ein Kanal fungiert als grundlegender Baustein eines OPC-Links. Diese Gruppe wird verwendet, um allgemeine Kanaleigenschaften (wie z.B. die ID-Attribute und den Betriebsmodus) anzugeben.



ID

Name: Benutzerdefinierte ID dieses Kanals. Bei jedem Serverprojekt muss jeder Kanalname eindeutig sein. Zwar können Namen bis zu 256 Zeichen lang sein, doch haben einige Client-Anwendungen beim Durchsuchen des Tag-Raums des OPC-Servers ein eingeschränktes Anzeigefenster. Der Kanalname ist ein Teil der OPC-Browserinformationen.

Informationen über reservierte Zeichen finden Sie in der Serverhilfe unter „So benennen Sie Kanäle, Geräte, Tags und Tag-Gruppen richtig“.

Beschreibung: Benutzerdefinierte Informationen über diesen Kanal.
Viele dieser Eigenschaften, einschließlich der Beschreibung, verfügen über ein zugeordnetes System-Tag.

Treiber: Ausgewähltes Protokoll/ausgewählter Treiber für diesen Kanal. Diese Eigenschaft gibt den Gerätetreiber an, der während der Kanalerstellung ausgewählt wurde. Es ist eine deaktivierte Einstellung in den Kanaleigenschaften.

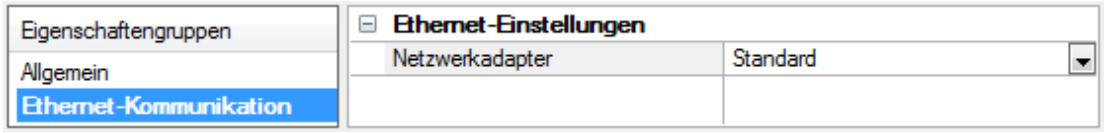
Hinweis: Beim Online-Vollzeitbetrieb des Servers können diese Eigenschaften jederzeit geändert werden. Dies schließt das Ändern des Kanalnamens ein, um zu verhindern, dass Clients Daten am Server registrieren. Wenn ein Client bereits ein Element vom Server abgerufen hat, bevor der Kanalname geändert wurde, sind die Elemente davon nicht beeinflusst. Wenn die Client-Anwendung das Element nach der Änderung des Kanalnamens freigibt und versucht, es mit dem alten Kanalnamen erneut abzurufen, wird das Element nicht akzeptiert. Unter Berücksichtigung dessen sollten keine Änderungen an den Eigenschaften erfolgen, sobald eine große Client-Anwendung entwickelt wurde. Verwenden Sie den Benutzermanager, um zu verhindern, dass Operatoren Eigenschaften ändern, und um Zugriffsrechte auf Serverfunktionen zu beschränken.

Diagnose

Diagnoseerfassung: Wenn diese Option aktiviert ist, stehen die Diagnoseinformationen des Kanals für OPC-Anwendungen zur Verfügung. Da für die Diagnosefunktionen des Servers eine minimale Mehraufwandsverarbeitung erforderlich ist, wird empfohlen, dass sie bei Bedarf verwendet werden und ansonsten deaktiviert sind. Die Standardeinstellung ist deaktiviert.
Hinweis: Diese Eigenschaft ist deaktiviert, wenn der Treiber Diagnosen nicht unterstützt.
Weitere Informationen dazu finden Sie in der Serverhilfe unter „Kommunikationsdiagnosen“.

Kanaleigenschaften – Ethernet-Kommunikation

Ethernet-Kommunikation kann für die Kommunikation mit Geräten verwendet werden.

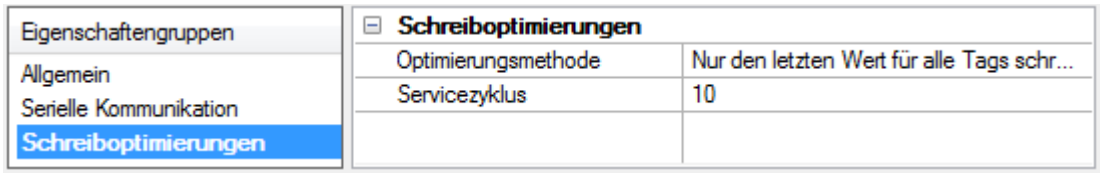


Ethernet-Einstellungen

Netzwerkadapter: Geben Sie den zu bindenden Netzwerkadapter an. Wenn "Standard" ausgewählt ist, wählt das Betriebssystem den Standardadapter aus.

Kanaleigenschaften – Schreiboptimierungen

Wie bei jedem OPC-Server ist das Schreiben von Daten auf das Gerät unter Umständen der wichtigste Aspekt der Anwendung. Der Server soll sicherstellen, dass die von der Client-Anwendung geschriebenen Daten rechtzeitig auf das Gerät gelangen. In Anbetracht dieses Ziels stellt der Server Optimierungseigenschaften bereit, anhand derer die jeweiligen Anforderungen erfüllt oder die Reaktionsfähigkeit der Anwendungen verbessert werden können.



Schreiboptimierungen

Optimierungsmethode: Mit dieser Option wird gesteuert, wie Schreibdaten an den zugrunde liegenden Kommunikationstreiber weitergeleitet werden. Die Optionen sind:

- **Alle Werte für alle Tags schreiben**: Mit dieser Option wird der Server gezwungen, für jeden Wert einen Schreibvorgang auf dem Controller zu versuchen. In diesem Modus sammelt der Server weiterhin

Schreibanforderungen und fügt sie der internen Schreibwarteschlange des Servers hinzu. Der Server verarbeitet die Schreibwarteschlange und versucht, sie zu leeren, indem er so schnell wie möglich Daten auf das Gerät schreibt. In diesem Modus wird sichergestellt, dass alles, was von den Client-Anwendungen geschrieben wird, an das Zielgerät gesendet wird. Dieser Modus sollte ausgewählt werden, wenn die Reihenfolge des Schreibvorgangs oder der Inhalt des Schreibelements eindeutig auf dem Zielgerät zu finden sein muss.

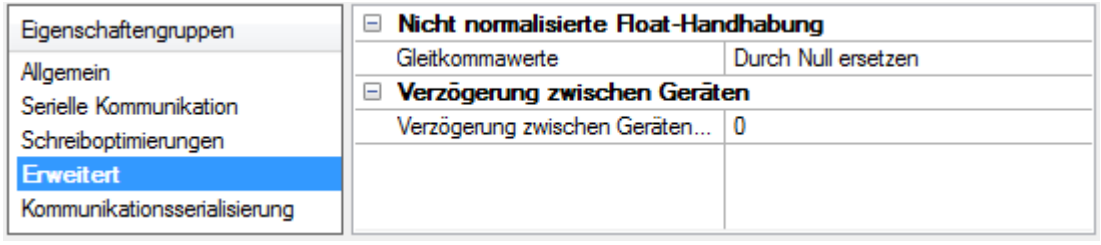
- **Nur den letzten Wert für nicht boolesche Tags schreiben**: Viele aufeinander folgende Schreibvorgänge für denselben Wert können sich aufgrund der Zeit, die tatsächlich zum Senden der Daten auf das Gerät erforderlich ist, in der Schreibwarteschlange ansammeln. Wenn der Server einen Schreibwert aktualisiert, der bereits in die Schreibwarteschlange eingefügt wurde, sind weitaus weniger Schreibvorgänge erforderlich, um denselben Endausgabewert zu erhalten. Auf diese Weise sammeln sich keine zusätzlichen Schreibvorgänge in der Warteschlange des Servers an. Wenn der Benutzer den Schiebeschalter nicht mehr verschiebt, erreicht der Wert im Gerät praktisch in derselben Zeit den richtigen Wert. Dem Modus entsprechend wird jeder Wert, der kein boolescher Wert ist, in der internen Warteschlange des Servers aktualisiert und bei der nächstmöglichen Gelegenheit an das Gerät gesendet. Dies kann die Anwendungsleistung erheblich verbessern.
  Hinweis: Mit dieser Option wird nicht versucht, Schreibvorgänge in Boolesche Werte zu optimieren. Dadurch können Benutzer den HMI-Datenvorgang optimieren, ohne Probleme mit Booleschen Operationen (z.B. eine vorübergehende Schaltfläche) zu verursachen.
- **Nur den letzten Wert für alle Tags schreiben**: Mit dieser Option wird die hinter der zweiten Optimierungsmethode stehende Theorie auf alle Tags angewendet. Sie ist besonders nützlich, wenn die Anwendung nur den letzten Wert an das Gerät senden muss. In diesem Modus werden alle Schreibvorgänge optimiert, indem die derzeit in der Schreibwarteschlange befindlichen Tags vor dem Senden aktualisiert werden. Dies ist der Standardmodus.

Servicezyklus: Wird verwendet, um das Verhältnis von Schreib- und Lesevorgängen zu steuern. Das Verhältnis basiert immer auf einem Lesevorgang für jeden zehnten Schreibvorgang. Für den Servicezyklus wird standardmäßig 10 festgelegt. Dies bedeutet, dass 10 Schreibvorgänge für jeden Lesevorgang erfolgen. Zwar führt die Anwendung eine große Anzahl fortlaufender Schreibvorgänge durch, doch muss sichergestellt werden, dass es für Lesedaten weiterhin Verarbeitungszeit gibt. Die Einstellung 1 hat zur Folge, dass ein Lesevorgang für jeden Schreibvorgang erfolgt. Wenn es keine durchzuführenden Schreibvorgänge gibt, werden Lesevorgänge fortlaufend verarbeitet. Dies ermöglicht eine Optimierung für Anwendungen mit fortlaufenden Schreibvorgängen gegenüber einem ausbalancierteren Datenzufluss und -abfluss.

Hinweis: Es wird empfohlen, dass für die Anwendung die Kompatibilität mit den Verbesserungen zur Schreiboptimierung charakteristisch ist, bevor sie in einer Produktionsumgebung verwendet wird.

Kanaleigenschaften - Erweitert

Diese Gruppe wird verwendet, um erweiterte Kanaleigenschaften anzugeben. Nicht alle Treiber unterstützen alle Eigenschaften; so wird die Gruppe "Erweitert" für jene Geräte nicht angezeigt.



Nicht normalisierte Float-Handhabung: Durch nicht normalisierte Float-Handhabung können Benutzer festlegen, wie ein Treiber mit nicht normalisierten IEEE-754-Gleitkommadaten umgeht. Ein nicht normalisierter Wert wird als "Unendlich", "Nichtzahlenwert (NaN)" oder als "Denormalisierte Zahl" definiert. Die Standardeinstellung ist Durch Null ersetzen. Für Treiber, die eine native Float-Handhabung aufweisen, wird standardmäßig unter Umständen "Nicht geändert" verwendet. Es folgen Beschreibungen der Optionen:

- **Durch Null ersetzen**: Diese Option ermöglicht es einem Treiber, nicht normalisierte IEEE-754-Gleitkommawerte durch Null zu ersetzen, bevor sie an Clients übertragen werden.

- **Nicht geändert**: Diese Option ermöglicht es einem Treiber, denormalisierte, normalisierte IEEE-754-Nichtzahlenwerte und unendliche IEEE-754-Werte ohne jegliche Konvertierung oder Änderungen an Clients zu senden.

Hinweis: Diese Eigenschaft ist deaktiviert, wenn der Treiber keine Gleitkommawerte unterstützt, oder wenn er nur die angezeigte Option unterstützt. Gemäß der Float-Normalisierungseinstellung des Kanals unterliegen nur Echtzeit-Treiber-Tags (wie z.B. Werte und Arrays) der Float-Normalisierung. Beispielsweise werden EFM-Daten nicht durch diese Einstellung beeinflusst.

Weitere Informationen über die Gleitkommawerte finden Sie unter "So arbeiten Sie mit nicht normalisierten Gleitkommawerten" in der Serverhilfe.

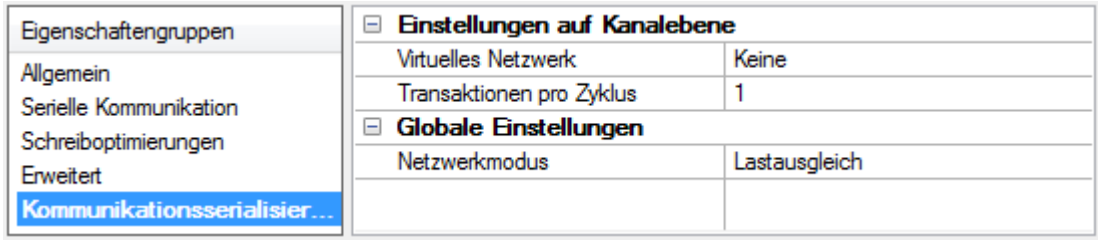
Verzögerung zwischen Geräten: Geben Sie die Zeitdauer an, in der der Kommunikationskanal das Senden einer Anforderung an das nächste Gerät verzögert, nachdem Daten vom aktuellen Gerät in demselben Kanal empfangen wurden. Null (0) deaktiviert die Verzögerung.

Hinweis: Diese Eigenschaft ist nicht für alle Treiber, Modelle und abhängige Einstellungen verfügbar.

Kanaleigenschaften – Kommunikationsserialisierung

Die Multithreading-Architektur des Servers ermöglicht Kanälen die parallele Kommunikation mit Geräten. Zwar ist das effizient, doch kann die Kommunikation in Fällen mit physischen Netzwerkeinschränkungen (wie Ethernet-Funksignale) serialisiert werden. Kommunikationsserialisierung schränkt die Kommunikation auf einen Kanal gleichzeitig innerhalb eines virtuellen Netzwerks ein.

Der Begriff "virtuelles Netzwerk" beschreibt eine Sammlung von Kanälen und zugeordneten Geräten, die dieselbe Pipeline für die Kommunikation verwenden. Beispielsweise ist die Pipeline eines Ethernet-Radios das Master-Radio. Alle Kanäle mit demselben Master-Radio werden demselben virtuellen Netzwerk zugeordnet. Kanäle dürfen jeweils nacheinander im Round-Robin-Verfahren kommunizieren. Standardmäßig kann ein Kanal eine Transaktion verarbeiten, bevor die Kommunikation an einen anderen Kanal übergeben wird. Eine Transaktion kann einen oder mehrere Tags einschließen Wenn der steuernde Kanal ein Gerät enthält, das nicht auf eine Anfrage antwortet, kann der Kanal die Steuerung erst bis zum Timeout der Transaktion freigeben. Dies hat Datenaktualisierungsverzögerungen für die anderen Kanäle im virtuellen Netzwerk zur Folge.



Einstellungen auf Kanalebene

Virtuelles Netzwerk Mit dieser Eigenschaft wird der Modus der Kommunikationsserialisierung des Kanals festgelegt. Zu den Optionen gehören "Keine" sowie "Netzwerk 1 – Netzwerk 50". Die Standardeinstellung ist "Keine". Es folgen Beschreibungen der Optionen:

- **Keine**: Mit dieser Option wird die Kommunikationsserialisierung für den Kanal deaktiviert.
- **Netzwerk 1 – Netzwerk 50**: Mit dieser Option wird das virtuelle Netzwerk angegeben, dem der Kanal zugewiesen wird.

Transaktionen pro Zyklus Mit dieser Eigenschaft wird die Anzahl einzelner blockierter/nicht blockierter Lese-/Schreibtransaktionen festgelegt, die auf dem Kanal vorkommen können. Wenn einem Kanal die Gelegenheit zur Kommunikation gegeben wird, wird diese Anzahl von Transaktionen versucht. Der gültige Bereich liegt zwischen 1 und 99. Die Standardeinstellung ist 1.

Globale Einstellungen

- **Netzwerkmodus**: Mit dieser Eigenschaft wird gesteuert, wie die Kanalkommunikation delegiert wird. Im Modus **Lastausgleich** wird jedem Kanal die Möglichkeit gegeben, nacheinander zu kommunizieren. Im Modus **Priorität** wird Kanälen die Möglichkeit gegeben, nach den folgenden Regeln (von der höchsten zur niedrigsten Priorität) zu kommunizieren:
  - Kanäle mit ausstehenden Schreibvorgängen haben den höchsten Vorrang.
  - Kanäle mit ausstehenden expliziten Lesevorgängen (durch interne Plugins oder externe Client-Schnittstellen) werden je nach Priorität des Lesevorgangs priorisiert.
  - Gescannte Lesevorgänge und andere periodische Ereignisse (treiberspezifisch).

  Die Standardeinstellung ist "Lastausgleich" und wirkt sich auf *alle* virtuellen Netzwerke und Kanäle aus.

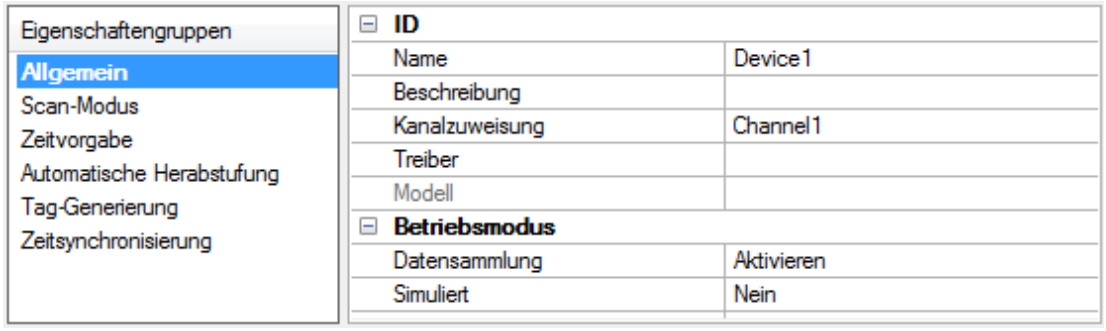
Geräte, die sich auf unaufgeforderte Antworten verlassen, sollten nicht in ein virtuelles Netzwerk eingefügt werden. In Situationen, wo die Kommunikationen serialisiert werden muss, wird empfohlen, dass "Automatische Herabstufung" aktiviert wird.

Aufgrund von Unterschieden in der Art und Weise, wie Treiber Daten lesen und schreiben (wie z.B. einzelne blockierte oder nicht blockierte Transaktionen) muss die Eigenschaft "Transaktionen pro Zyklus" der Anwendung möglicherweise angepasst werden. Berücksichtigen Sie dabei die folgenden Faktoren:

- Wie viele Tags müssen von jedem Kanal gelesen werden?
- Wie oft werden Daten in jeden Kanal geschrieben?
- Verwendet der Kanal einen seriellen oder einen Ethernet-Treiber?
- Liest der Treiber Tags in separaten Anfragen, oder werden mehrere Tags in einem Block gelesen?
- Wurden die Zeitvorgabe-Eigenschaften des Geräts (wie z.B. Anforderungs-Timeout und Fehlgeschlagen nach *x* aufeinander folgenden Timeouts) für das Kommunikationsmedium des virtuellen Netzwerks optimiert?

Geräteeigenschaften – Allgemein

Ein Gerät stellt ein einzelnes Ziel in einem Kommunikationskanal dar. Wenn der Treiber mehrere Controller unterstützt, müssen Benutzer eine Geräte-ID für jeden Controller eingeben.



Identifikation

Name: Diese Eigenschaft gibt den Namen des Geräts an. Es ist ein logischer, benutzerdefinierter Name, der bis zu 256 Zeichen lang sein und auf mehreren Kanälen verwendet werden kann.

Hinweis: Zwar sind beschreibende Namen allgemein eine gute Idee, doch haben einige OPC-Client-Anwendungen beim Durchsuchen des Tag-Raums des OPC-Servers möglicherweise ein eingeschränktes Anzeigefenster. Der Geräte- und Kanalname werden ebenfalls Teil der Informationen zum Durchsuchen der Hierarchiebaumstruktur. Innerhalb eines OPC-Clients würde die Kombination aus Kanalname und Gerätename als "ChannelName.DeviceName" angezeigt werden.
Weitere Informationen dazu finden Sie in der Serverhilfe unter "So benennen Sie Kanäle, Geräte, Tags und Tag-Gruppen richtig".

Beschreibung: Benutzerdefinierte Informationen über dieses Gerät.
Viele dieser Eigenschaften, einschließlich der Beschreibung, verfügen über ein zugeordnetes System-Tag.

Kanalzuweisung: Benutzerdefinierter Name des Kanals, zu dem dieses Gerät derzeit gehört.

Treiber: Ausgewählter Protokolltreiber für dieses Gerät. Diese Eigenschaft gibt den während der Kanalerstellung ausgewählten Treiber an. Sie ist in den Kanaleigenschaften deaktiviert.

Modell: Diese Eigenschaft gibt den bestimmten Typ des Geräts an, das dieser ID zugeordnet ist. Der Inhalt des Dropdown-Menüs hängt vom Typ des verwendeten Kommunikationstreibers ab. Modelle, die von einem Treiber nicht unterstützt werden, sind deaktiviert. Wenn der Kommunikationstreiber mehrere Gerätemodelle unterstützt, kann die Modellauswahl nur geändert werden, wenn keine Client-Anwendungen mit dem Gerät verbunden sind.

Hinweis: Wenn der Kommunikationstreiber mehrere Modelle unterstützt, sollten Benutzer versuchen, die Modellauswahl mit dem physischen Gerät abzugleichen. Wenn das Gerät im Dropdown-Menü nicht dargestellt wird, wählen Sie ein Modell aus, das dem Zielgerät am ehesten entspricht. Einige Treiber unterstützen die Modellauswahl "Offen", wodurch Benutzer kommunizieren können, ohne bestimmte Details des Zielgeräts zu kennen. Weitere Informationen dazu finden Sie in der Hilfedokumentation des Treibers.

ID: Diese Eigenschaft gibt die Station, den Knoten, die ID oder die Adresse des Geräts an. Der Typ der eingegebenen ID hängt vom verwendeten Kommunikationstreiber ab. Für viele Treiber ist die ID ein numerischer Wert. Treiber, die eine numerische ID unterstützen, stellen Benutzern die Option zum Eingeben eines numerischen Werts bereit, dessen Format den Anforderungen der Anwendung oder der Charakteristik des ausgewählten Kommunikationstreibers entsprechend angepasst werden kann. Das ID-Format kann Dezimal, Oktal oder Hexadezimal sein. Wenn der Treiber Ethernet-basiert ist oder eine unkonventionelle Station oder einen unkonventionellen Knotennamen unterstützt, kann die TCP/IP-Adresse des Geräts ggf. als Geräte-ID verwendet werden. TCP/IP-Adressen bestehen aus vier Werten, die durch Punkte getrennt sind, wobei jeder Wert im Bereich von 0 bis 255 liegt. Einige Geräte-IDs sind zeichenfolgenbasiert. Abhängig vom Treiber gibt es möglicherweise zusätzliche zu konfigurierende Eigenschaften innerhalb des ID-Felds.

Betriebsmodus

Datensammlung: Diese Eigenschaft steuert den aktiven Status des Geräts. Zwar sind Gerätekommunikationen standardmäßig aktiviert, doch kann diese Eigenschaft verwendet werden, um ein physisches Gerät zu deaktivieren. Kommunikationen werden nicht versucht, wenn ein Gerät deaktiviert ist. Vom Standpunkt eines Clients werden die Daten als ungültig markiert und Schreibvorgänge werden nicht akzeptiert. Diese Eigenschaft kann jederzeit durch diese Eigenschaft oder die System-Tags des Geräts geändert werden.

Simuliert: Diese Option versetzt das Gerät in den Simulationsmodus. In diesem Modus versucht der Treiber nicht, mit dem physischen Gerät zu kommunizieren, aber der Server gibt weiterhin gültige OPC-Daten zurück. Durch Auswählen von "Simuliert" wird die physische Kommunikation mit dem Gerät angehalten, OPC-Daten können jedoch als gültige Daten dem OPC-Client zurückgegeben werden. Im Simulationsmodus behandelt der Server alle Gerätedaten als reflektierend: was auch immer in das simulierte Gerät geschrieben wird, wird zurückgelesen, und jedes OPC-Element wird einzeln behandelt. Die Speicherzuordnung des Elementes basiert auf dem Gruppenaktualisierungsintervall. Die Daten werden nicht gespeichert, wenn der Server das Element entfernt (z.B., wenn der Server neu initialisiert wird). Die Standardeinstellung ist "Nein".

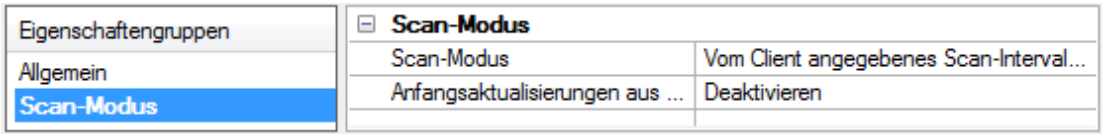
Hinweise:

1. Dieses System-Tag (_Simulated) ist schreibgeschützt und kann für den Laufzeitschutz nicht geschrieben werden. Das System-Tag ermöglicht es, dass diese Eigenschaft vom Client überwacht wird.
2. Im Simulationsmodus basiert die Speicherzuordnung des Elements auf Client-Aktualisierungsintervallen (Gruppenaktualisierungsintervall für OPC-Clients oder Scan-Intervall für native und DDE-Schnittstellen). Das bedeutet, dass zwei Clients, die dasselbe Element mit unterschiedlichen Aktualisierungsintervallen referenzieren, verschiedene Daten zurückgeben.

Der Simulationsmodus ist nur für Test- und Simulationszwecke. Es sollte niemals in einer Produktionsumgebung nie verwendet werden.

Geräteeigenschaften – Scan-Modus

Der Scan-Modus gibt das vom abonnierten Client angeforderte Scan-Intervall für Tags an, die Gerätekommunikation erfordern. Synchrone und asynchrone Lese- und Schreibvorgänge des Geräts werden so bald wie möglich verarbeitet; unbeeinflusst von den Eigenschaften für den Scan-Modus.



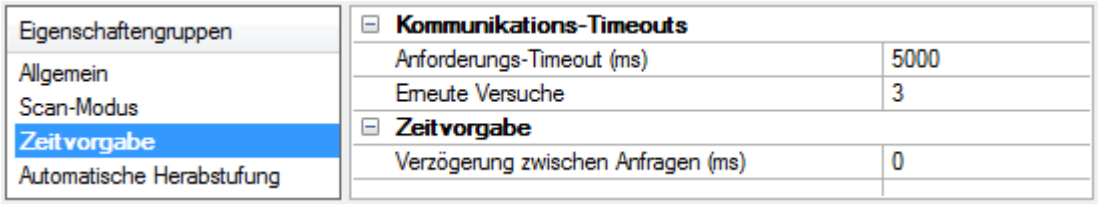
Scan-Modus: Gibt an, wie Tags im Gerät auf an abonnierte Clients gesendete Aktualisierungen gescannt werden. Es folgen Beschreibungen der Optionen:

- **Vom Client angegebenes Scan-Intervall berücksichtigen**: Dieser Modus verwendet das vom Client angeforderte Scan-Intervall.
- **Datenanfrage nicht schneller als Scan-Intervall**: Dieser Modus gibt das maximale Scan-Intervall an, das verwendet werden soll. Der gültige Bereich liegt zwischen 10 und 99999990 Millisekunden. Die Standardeinstellung ist 1000 Millisekunden.
  Hinweis: Wenn der Server über einen aktiven Client und Elemente für das Gerät verfügt und der Wert für das Scan-Intervall erhöht wird, werden die Änderungen sofort wirksam. Wenn der Wert für das Scan-Intervall verringert wird, werden die Änderungen erst wirksam, wenn alle Client-Anwendungen getrennt wurden.
- **Alle Datenanfragen im Scan-Intervall**: Dieser Modus erzwingt, dass Tags im angegebenen Intervall nach abonnierten Clients gescannt werden. Der gültige Bereich liegt zwischen 10 und 99999990 Millisekunden. Die Standardeinstellung ist 1000 Millisekunden.
- **Nicht scannen, nur Abruf anfordern**: In diesem Modus werden Tags, die zum Gerät gehören, nicht periodisch abgerufen, und es wird auch kein Lesevorgang durchgeführt, um den Anfangswert eines Elements abzurufen, sobald es aktiv wird. Es liegt in der Verantwortung des Clients, nach Aktualisierungen abzurufen, entweder durch Schreiben in das _DemandPoll-Tag oder durch Ausgeben expliziter Lesevorgänge des Geräts für einzelne Elemente. *Weitere Informationen finden Sie unter "Geräte-Bedarfsabruf" in der Serverhilfe*.
- **Durch Tag angegebenes Scan-Intervall berücksichtigen**: Dieser Modus erzwingt das Scannen statischer Tags im Intervall, das in ihrer statischen Konfiguration Tag-Eigenschaften angegeben wurde. Dynamische Tags werden in dem vom Client angegebenen Scan-Intervall gescannt.

Anfangsaktualisierungen aus Cache: Wenn diese Option aktiviert ist, kann der Server die ersten Aktualisierungen für neu aktivierte Tag-Referenzen aus gespeicherten (Cache-)Daten zur Verfügung stellen. Cache-Aktualisierungen können nur bereitgestellt werden, wenn die neue Elementreferenz dieselben Eigenschaften für Adresse, Scan-Intervall, Datentyp, Client-Zugriff und Skalierung gemeinsam nutzt. Ein Lesevorgang des Geräts wird nur für die Anfangsaktualisierung für die erste Client-Referenz verwendet. Der Standardeinstellung ist "Deaktiviert"; immer wenn ein Client eine Tag-Referenz aktiviert, versucht der Server, den Anfangswert vom Gerät zu lesen.

Geräteeigenschaften - Zeitvorgabe

Mithilfe der Zeitvorgabe-Eigenschaften des Geräts kann die Antwort des Treibers auf Fehlerbedingungen so angepasst werden, dass sie den Anforderungen der Anwendung entspricht. In vielen Fällen erfordert die Umgebung für eine optimale Leistung Änderungen an diesen Eigenschaften. Faktoren wie elektrisch generiertes Rauschen, Modemverzögerungen und fehlerhafte physische Verbindungen können beeinflussen, wie viele Fehler oder Timeouts ein Kommunikationstreiber feststellt. Zeitvorgabe-Eigenschaften sind für jedes konfigurierte Gerät spezifisch.



Kommunikations-Timeouts

Verbindungs-Timeout: Mit dieser Eigenschaft (die in erster Linie von Ethernet-basierten Treibern verwendet wird) wird die Zeitdauer gesteuert, die zum Herstellen einer Socket-Verbindung mit einem Remote-Gerät erforderlich ist. Die Verbindungszeit des Gerät ist häufig länger als normale Kommunikationsanforderungen mit demselben Gerät. Der gültige Bereich liegt zwischen 1 und 30 Sekunden. Die Standardeinstellung ist

normalerweise 3 Sekunden, kann jedoch abhängig vom jeweiligen Treiber unterschiedlich sein. Wenn diese Einstellung nicht vom Treiber unterstützt wird, ist sie deaktiviert.
Hinweis: Aufgrund der Art der UDP-Verbindungen ist die Einstellung für Verbindungs-Timeout nicht anwendbar, wenn die Kommunikation über UDP erfolgt.

Anforderungs-Timeout: Mit dieser Eigenschaft wird ein von allen Treibern verwendetes Intervall festgelegt, um zu bestimmen, wie lange der Treiber abschließend auf eine Antwort vom Zielgerät wartet. Der gültige Bereich liegt zwischen 50 und 9.999.999 Millisekunden (167,6667 Minuten). Die Standardeinstellung ist im Allgemeinen 1000 Millisekunden, kann jedoch abhängig vom Treiber unterschiedlich sein. Das Standard-Timeout für die meisten seriellen Treiber basiert auf einer Baudrate von 9600 Baud oder besser. Wenn ein Treiber bei niedrigeren Baudraten verwendet wird, erhöhen Sie das Timeout, um die erhöhte Zeit auszugleichen, die zum Abrufen von Daten erforderlich ist.

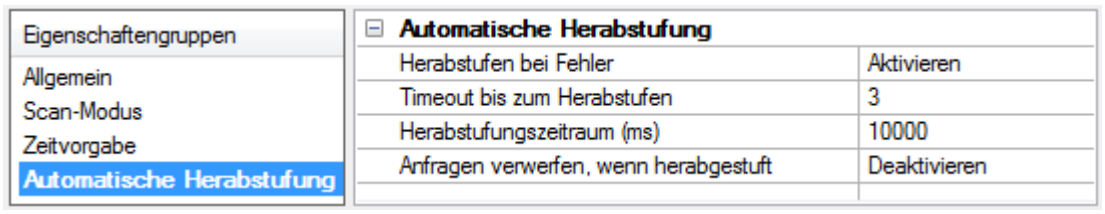
Erneute Versuche: Mit dieser Eigenschaft wird festgelegt, wie häufig der Treiber eine Kommunikationsanforderung wiederholt, bevor er die Anforderung als fehlgeschlagen und das Gerät als fehlerhaft erachtet. Der gültige Bereich liegt zwischen 1 und 10. Die Standardeinstellung ist normalerweise 3, kann sich jedoch abhängig vom jeweiligen Treiber ändern. Die Anzahl der für eine Anwendung konfigurierten Wiederholungen hängt größtenteils von der Kommunikationsumgebung ab. Diese Eigenschaft trifft sowohl auf Verbindungsversuche als auch auf Anforderungsversuche zu.

Zeitvorgabe

Verzögerung zwischen Anfragen: Mit dieser Eigenschaft wird festgelegt, wie lange der Treiber wartet, bevor er die nächste Anforderung an das Zielgerät sendet. Sie setzt das dem Gerät zugewiesene normale Tag-Abfrageintervall sowie einmalige Lese- und Schreibvorgänge außer Kraft. Diese Verzögerung kann bei Geräten mit langsamen Durchlaufzeiten und in Situationen nützlich sein, in denen die Netzwerklast problematisch ist. Das Konfigurieren einer Verzögerung für ein Gerät wirkt sich auf die Kommunikation mit allen anderen Geräten im Kanal aus. Es wird empfohlen, dass Benutzer jedes Gerät trennen, das eine Verzögerung zwischen Anfragen für einen separaten Kanal erfordert (sofern möglich). Andere Kommunikationseigenschaften (wie z.B. Kommunikationsserialisierung) können diese Verzögerung verlängern. Der gültige Bereich liegt zwischen 0 und 300000 Millisekunden; jedoch können einige Treiber ggf. den maximalen Wert wegen einer Funktion ihrer spezifischen Konstruktion beschränken. Die Standardeinstellung ist 0. Dies weist darauf hin, dass es keine Verzögerung zwischen Anfragen mit dem Zielgerät gibt.
Hinweis: Nicht alle Treiber unterstützen Verzögerung zwischen Anfragen. Diese Einstellung wird nicht angezeigt, wenn sie nicht zur Verfügung steht.

Geräteeigenschaften – Automatische Herabstufung

Die Eigenschaften für automatische Herabstufung können ein Gerät vorübergehend in den Nicht-Scan-Modus versetzen, falls das Gerät nicht antwortet. Dadurch, dass ein nicht reagierendes Gerät für einen bestimmten Zeitraum offline gestellt wird, kann der Treiber weiterhin seine Kommunikation mit anderen Geräten in demselben Kanal optimieren. Nach Ablauf dieses Zeitraums versucht der Treiber die Kommunikation mit dem nicht reagierenden Gerät erneut. Wenn das Gerät reagiert, wird es wieder zum Scannen freigegeben. Andernfalls wird sein Nicht-Scan-Zeitraum erneut gestartet.



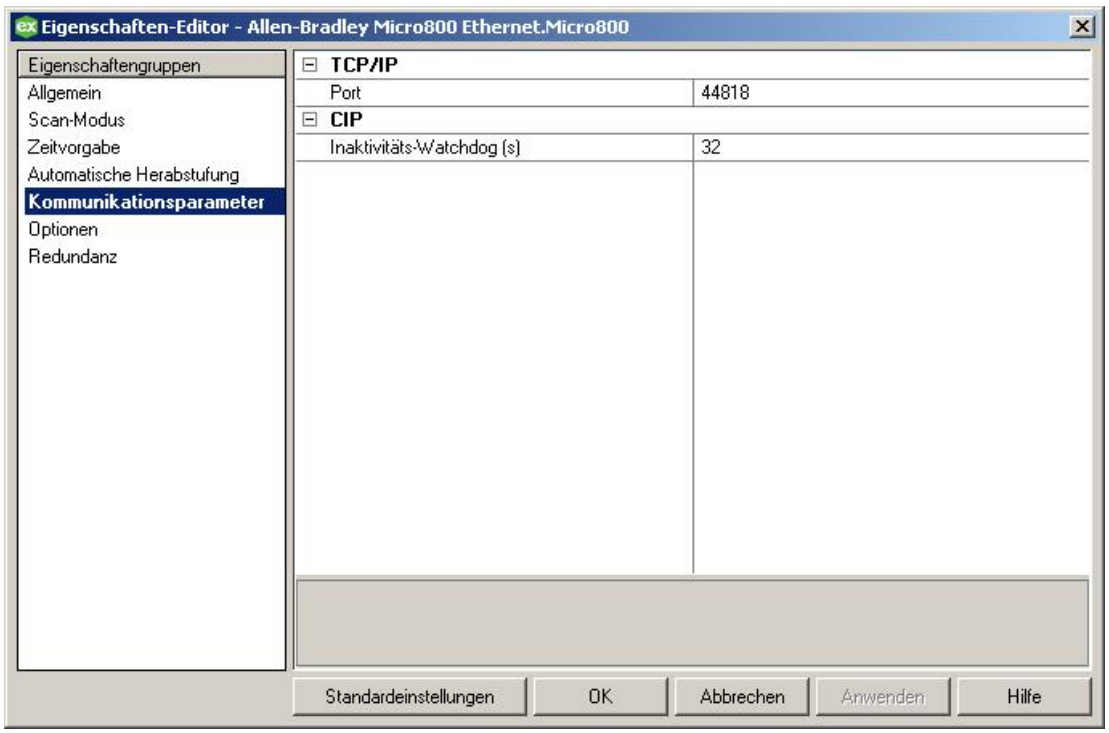
Herabstufen bei Fehler: Wird diese Option aktiviert, wird das Gerät automatisch in den Nicht-Scan-Modus versetzt, bis es wieder antwortet.
Tipp: Ermitteln Sie, wenn sich ein Gerät im Nicht-Scan-Modus befindet, indem Sie seinen herabgestuften Status mit dem _AutoDemoted-System-Tag überwachen.

Timeout bis zum Herabstufen: Legen Sie fest, wie viele aufeinander folgende Zyklen von Anforderungs-Timeouts und Wiederholungen vorkommen, bevor das Gerät in den Nicht-Scan-Modus versetzt wird. Der gültige Bereich ist 1 bis 30 aufeinander folgende Fehlschläge. Die Standardeinstellung ist 3.

Herabstufungszeitraum: Gibt an, wie lange das Gerät im Nicht-Scan-Modus sein sollte, wenn der Timeout-Wert erreicht wird. Während dieses Zeitraums werden keine Leseanforderungen an das Gerät gesendet, und für alle den Leseanforderungen zugeordneten Daten wird schlechte Qualität festgelegt. Wenn dieser Zeitraum abgelaufen ist, versetzt der Treiber das Gerät in den Scan-Modus und ermöglicht einen weiteren Kommunikationsversuch. Der gültige Bereich liegt zwischen 100 und 3600000 Millisekunden. Die Standardeinstellung ist 10000 Millisekunden.

Anfragen verwerfen, wenn herabgestuft: Durch Aktivieren dieser Option wird ausgewählt, ob Schreibanforderungen während des Nicht-Scan-Zeitraums versucht werden sollten. Deaktivieren Sie diese Option, damit Schreibanforderungen unabhängig vom Herabstufungszeitraum immer gesendet werden. Aktivieren Sie diese Option, um Schreibvorgänge zu verwerfen; auf dem Server schlägt jede von einem Client empfangene Schreibanforderung automatisch fehl, und es wird keine Meldung im Ereignisprotokoll angezeigt.

Geräteeigenschaften – Kommunikationsparameter



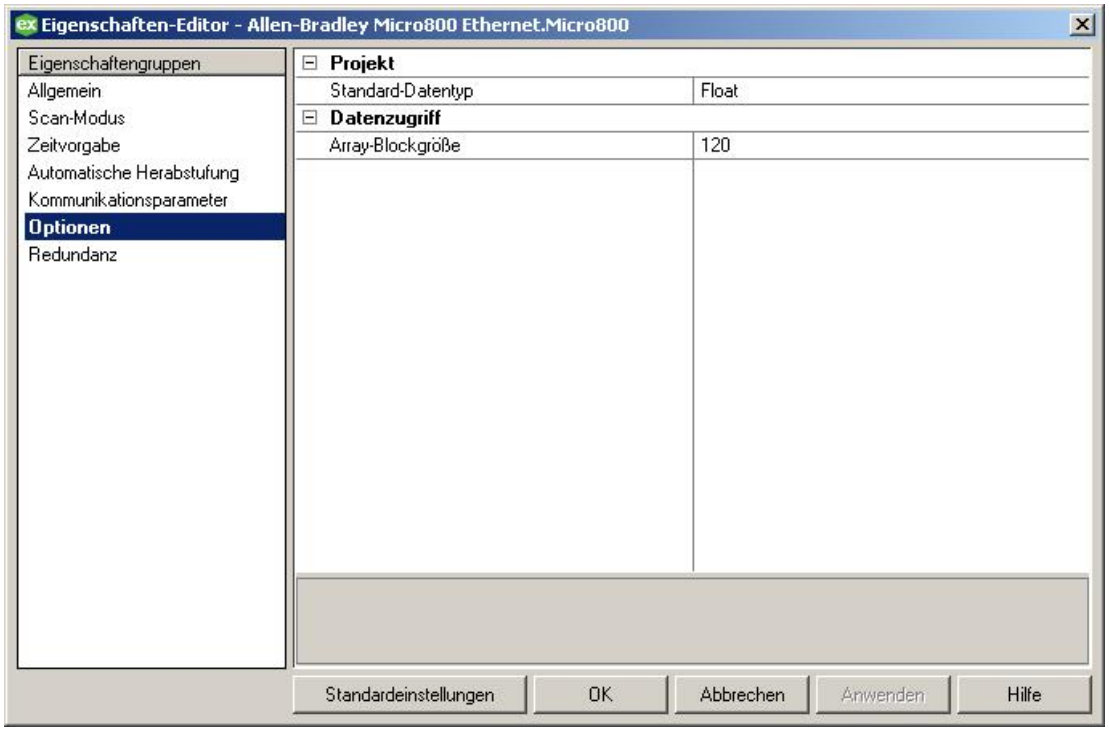
TCP/IP

Port: Gibt die Port-Nummer an, für deren Verwendung das Remote-Gerät konfiguriert ist. Der gültige Bereich liegt zwischen 0 und 65535. Die Standardeinstellung ist 44818.

CIP

Inaktivitätsüberwachung: Gibt die Dauer in Sekunden an, die eine Verbindung inaktiv (ohne Lese-/Schreibtransaktionen) bleiben kann, bevor sie vom Controller geschlossen wird. Allgemein gilt: Je größer der Wert für Inaktivitätsüberwachung, desto mehr Zeit wird benötigt, bis Verbindungsressourcen durch den Controller freigegeben werden (und umgekehrt). Der Standardwert beträgt 32 Sekunden.

Geräteeigenschaften – Optionen



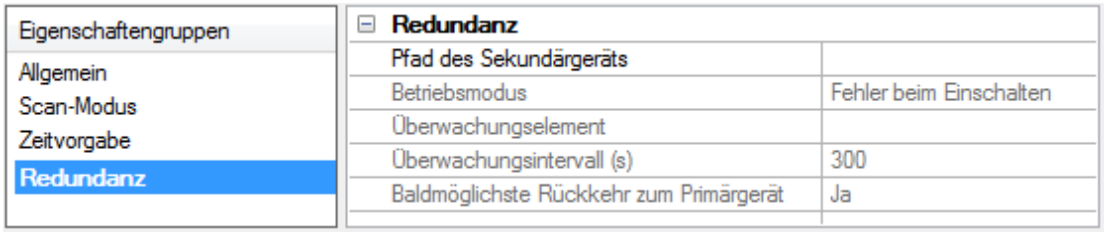
Projekt

Standard-Datentyp: Wählen Sie den einem Client-/Server-Tag zugewiesenen Datentyp aus, wenn der Standardtyp während des Hinzufügens, Änderns oder Importierens des Tags ausgewählt wird. Die Standardeinstellung ist Float. Der Standard-Datentyp wird Tags zugewiesen, wenn ein dynamisches Tag im Client mit "Nativ" als seinem zugewiesenen Datentyp erstellt und wenn ein statisches Tag im Server mit "Standard" als seinem zugewiesenen Datentyp erstellt wird.

Datenzugriff

Array-Blockgröße: Gibt die maximale Anzahl unteilbarer Array-Elemente an, die in einer einzelnen Transaktion gelesen werden. Der Bereich liegt zwischen 30 und 3840 Elementen. Der Standardwert ist 120 Elemente.
Für boolesche Arrays wird ein einzelnes Element als Bit-Array mit 32 Elementen angesehen. Durch Festlegen der Blockgröße auf 30 Elemente werden 960 Bit-Elemente konvertiert, wohingegen bei 3840 Elementen 122880 Bit-Elemente konvertiert werden.

Geräteeigenschaften - Redundanz



Redundanz steht mit dem Plugin für Redundanz auf Medienebene zur Verfügung.

Weitere Informationen dazu erhalten Sie auf der Website, von einem Vertriebsrepräsentanten oder im Benutzerhandbuch.

Leistungsoptimierungen

Zwar ist Allen-Bradley Micro800 Ethernet Driver schnell, doch können ggf. einige Richtlinien zum Erzielen maximaler Leistung angewendet werden. Wenn Sie weitere Informationen zur Optimierung auf Kommunikations- und Anwendungsebene erhalten möchten, wählen Sie eine Verknüpfung in der Liste unten aus.

Kommunikation optimieren
Anwendungen optimieren

Kommunikation optimieren

Wie bei jedem programmierbaren Controller gibt es viele Möglichkeiten, die Gesamtleistung und Systemkommunikation zu verbessern.

Native Tag-Namen kurz halten

Wenn native Tags vom Gerät gelesen und in das Gerät geschrieben werden, wird der zugehörige symbolische Name in der Kommunikationsanforderung angegeben. Dementsprechend gilt: Je länger der Tag-Name, desto größer die Anforderung.

Array-Elemente blockiert

Um das Lesen unteilbarer Array-Elemente zu optimieren, lesen Sie einen Array-Block in einer einzelnen Anfrage statt individuell. Je mehr Elemente in einem Block gelesen werden, desto größer die Leistung. Da der Transaktionsaufwand und die -verarbeitung die meiste Zeit beansprucht, führen Sie so wenige Transaktionen wie möglich aus, während Sie so viele gewünschte Tags wie möglich scannen. Dies ist wesentlich für das Blockieren von Array-Elementen.

Blockgrößen werden als eine Elementanzahl angegeben. Eine Blockgröße von 120 Elementen bedeutet, dass maximal 120 Array-Elemente bei einer Anforderung gelesen werden. Die maximale Blockgröße beträgt 3840 Elemente. Boolean-Arrays werden unterschiedlich behandelt: Im Protokoll ist ein Boolean-Array ein 32-Bit-Array. Somit fordert das anfordernde Element 0 die Bits 0 bis 31 an. Um bei der Diskussion Konsistenz beizubehalten, wird ein Boolean-Array-Element als einzelnes Bit angesehen. Zusammenfassend betrachtet sieht die maximale Anzahl von Array-Elementen (basierend auf Blockgröße 3840), die angefordert werden können, wie folgt aus: 122880 BOOL, 3840 SINT, 3840 INT, 3840 DINT, 3840 LINT und 3840 REAL.

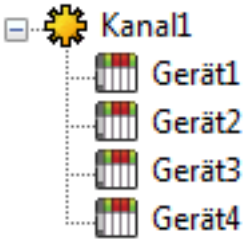
Die Blockgröße ist anpassbar und sollte je nach aktuellem Projekt ausgewählt werden. Beispiel: Wenn die Array-Elemente 0-26 und das Element 3839 zu lesende Tags sind, ist das Verwenden einer Blockgröße von 3840 nicht nur zu viel Aufwand, sondern auch schädlich für die Leistung des Treibers. Dies liegt daran, dass alle Elemente zwischen 0 und 3839 bei jeder Anforderung gelesen werden, selbst wenn nur 28 dieser Elemente von Bedeutung sind. In diesem Fall ist eine Blockgröße von 30 geeigneter. Die Elemente 0-26 würden in einer Anforderung bedient und das Element 3839 in der nächsten Anforderung.

Siehe auch: *Optionen*

Anwendungen optimieren

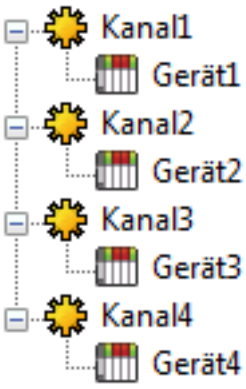
Allen-Bradley Micro800 Ethernet Driver wurde dafür konzipiert, eine optimale Leistung mit der geringsten Auswirkung auf die Gesamtleistung des Systems zu erzielen. Zwar ist der Treiber schnell, doch gibt es einige Richtlinien zum Erzielen maximaler Leistung.

Der Server bezeichnet Kommunikationsprotokolle wie Micro800-Ethernet von Allen-Bradley als Kanal. Jeder in der Anwendung definierte Kanal stellt einen separaten Ausführungspfad im Server dar. Sobald ein Kanal festgelegt wurde, muss eine Reihe von Geräten unter diesem Kanal definiert werden. Jedes dieser Geräte stellt eine einzelne Micro800-CPU dar, von der Daten gesammelt werden. Zwar ermöglicht diese Methode zum Definieren der Anwendung ein hohes Leistungsniveau, doch kann damit nicht vollständig Nutzen aus Allen-Bradley Micro800 Ethernet Driver oder dem Netzwerk gezogen werden. Ein Beispiel dafür, wie die Anwendung bei Konfiguration mit einem Kanal möglicherweise aussehen kann, wird im Folgenden gezeigt.



Jedes Gerät wird unter einem einzelnen Kanal (sog. "Micro800") angezeigt. In dieser Konfiguration muss sich der Treiber schnellstmöglich von einem Gerät zum nächsten bewegen, um Informationen in einem effektiven Intervall zu sammeln. Je mehr Geräte hinzugefügt oder je mehr Informationen von einem einzelnen Gerät angefordert werden, desto mehr leidet das Aktualisierungsintervall insgesamt.

Wenn Allen-Bradley Micro800 Ethernet Driver nur einen Kanal definieren könnte, würde das Beispiel oben die einzige verfügbare Option darstellen. Der Treiber kann jedoch bis zu 256 Kanäle definieren. Durch Verwenden mehrerer Kanäle wird die Arbeitsbelastung bei der Datensammlung verteilt, indem mehrere Anfragen gleichzeitig an das Netzwerk gestellt werden. Ein Beispiel dafür, wie dieselbe Anwendung aussehen kann, wenn sie mit mehreren Kanälen konfiguriert wird, um die Leistung zu verbessern, wird im Folgenden gezeigt.



Jedes Gerät wurde jetzt unter seinem eigenen Kanal festgelegt. In dieser neuen Konfiguration wird ein einziger Ausführungspfad dediziert für das Sammeln von Daten von jedem Gerät eingesetzt. Wenn die Anwendung über 256 oder weniger Geräte verfügt, kann sie genauso optimiert werden wie hier gezeigt.

Die Leistung verbessert sich, selbst wenn die Anwendung mehr als 256 Geräte aufweist. Zwar sind 256 oder weniger Geräte möglicherweise ideal, jedoch zieht die Anwendung Nutzen aus zusätzlichen Kanälen. Obwohl das Verteilen der Gerätelast auf alle Kanäle zur Folge hat, dass sich der Server erneut von Gerät zu Gerät bewegt, kann dies nun mit weit weniger zu verarbeitenden Geräten auf einem einzigen Kanal erfolgen.

Datentypbeschreibung

Datentyp	Beschreibung
Boolean	Einzelnes Bit
Byte	8-Bit-Wert ohne Vorzeichen
Char	8-Bit-Wert mit Vorzeichen
Word	16-Bit-Wert ohne Vorzeichen
Short	16-Bit-Wert mit Vorzeichen
DWord	32-Bit-Wert ohne Vorzeichen
Long	32-Bit-Wert mit Vorzeichen
BCD	BCD mit zwei Byte gepackt, vier Dezimalstellen
LBCD	BCD mit vier Byte gepackt, acht Dezimalstellen
Float	32-Bit-IEEE-Gleitkommazahl
Double	64-Bit-IEEE-Gleitkommazahl
Datum	64-Bit-Datum/Uhrzeit
String	Mit Null beendetes Zeichen-Array

Adressbeschreibungen

Micro800 verwendet Tags oder eine symbolbasierte Adressierungsstruktur, die als native Tags bezeichnet werden. Diese Tags unterscheiden sich von herkömmlichen SPS-Datenelementen dahingehend, dass der Tag-Name selbst die Adresse ist und keine Datei oder Registernummer.

Allen-Bradley Micro800 Ethernet Driver ermöglicht Benutzern den Zugriff auf die unteilbaren Datentypen des Controllers: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, LINT, ULINT, LWORD, REAL, LREAL und SHORT_STRING. Zwar handelt es sich bei einigen der vordefinierten Typen um Strukturen, doch basieren sie letztendlich auf diesen unteilbaren Datentypen. Somit sind alle nicht strukturierten (unteilbaren) Mitglieder einer Struktur zugänglich. Beispiel: Ein TIMER (Zeitgeber) kann keinem Server-Tag zugewiesen werden, wohingegen ein unteilbares TIMER-Mitglied dem Tag zugewiesen werden kann (beispielsweise TIMER.EN, TIMER.ACC usw.). Wenn es sich bei einem Strukturmitglied um eine Struktur selbst handelt, müssen beide Strukturen entsprechend erweitert werden, damit auf ein unteilbares Mitglied der Substruktur zugegriffen werden kann. Dies ist üblicherweise bei benutzer- und moduldefinierten Typen und bei keinem der vordefinierten Typen zu finden.

Unteilbarer Datentyp	Beschreibung	Client-Typ	Bereich
BOOL	Einzel-Bit-Wert	VT_BOOL	0, 1
SINT	8-Bit-Wert mit Vorzeichen	VT_I1	-128 bis 127
USINT	8-Bit-Wert ohne Vorzeichen	VT_UI1	0 bis 255
BYTE	Bit-String (8 Bit)	VT_UI1	0 bis 255
INT	16-Bit-Wert mit Vorzeichen	VT_I2	-32.768 bis 32.767
UINT	16-Bit-Wert ohne Vorzeichen	VT_UI2	0 bis 65535
WORD	Bit-String (16 Bit)	VT_UI2	0 bis 65535
DINT	32-Bit-Wert mit Vorzeichen	VT_I4	-2.147.483.648 bis 2.147.483.647
UDINT	32-Bit-Wert ohne Vorzeichen	VR_UI4	0 bis 4294967296
DWORD	Bit-String (32 Bit)	VR_UI4	0 bis 4294967296
LINT	64-Bit-Wert mit Vorzeichen	VT_R8	-1.798E+308 bis -2.225E-308, 0, 2.225E-308 bis 1.798E+308
ULINT	64-Bit-Wert ohne Vorzeichen	VT_R8	-1.798E+308 bis -2.225E-308, 0, 2.225E-

Unteilbarer Datentyp	Beschreibung	Client-Typ	Bereich
			308 bis 1.798E+308
LWORD	Bit-String (64 Bit)	VT_R8	-1.798E+308 bis -2.225E-308, 0, 2.225E-308 bis 1.798E+308
REAL	32-Bit-IEEE-Gleitkommazahl	VT_R4	1.1755 E-38 bis 3.403E38, 0, -3.403E-38 bis -1.1755
LREAL	64-Bit-IEEE-Gleitkommazahl	VT_R8	-1.798E+308 bis -2.225E-308, 0, 2.225E-308 bis 1.798E+308
SHORT_ STRING	Zeichenfolge. Maximal 80 Zeichen sind zulässig.	VT_ BSTR	

Siehe auch: *Erweiterte Anwendungsfälle*

Tag-Adressregeln für Client/Server
Native Tag-Namen entsprechen Tag-Adressen für Client/Server. Für native Tag-Namen (Eingabe erfolgt über die Connected Components Workbench) und Tag-Adressen für Client/Server gelten die IEC 1131-3-ID-Regeln. Die Regeln lassen sich wie folgt beschreiben:

- Müssen mit einem Buchstaben oder Unterstrich beginnen.
- Dürfen nur Buchstaben und Unterstriche enthalten.
- Dürfen bis zu 40 Zeichen lang sein.
- Dürfen keine aufeinander folgenden Unterstriche enthalten.
- Bei Zeichen wird die Groß-/Kleinschreibung nicht beachtet.

Tag-Namensregeln für Client/Server
Die Zuweisung des Tag-Namens im Server unterscheidet sich von der Adresszuweisung dahingehend, dass Namen nicht mit einem Unterstrich beginnen dürfen.

Siehe auch: *Leistungsoptimierungen*

Adressformate

Native Tags können ggf. auf verschiedene Weise statisch im Server oder dynamisch vom Client aus adressiert werden. Das Format des Tags hängt von seinem Typ und seiner beabsichtigten Verwendung ab. Beispiel: Das Bit-Format würde beim Zugriff auf ein Bit innerhalb eines Tags des SINT-Typs verwendet werden. Informationen zum Adressformat und zur Syntax finden Sie in der Tabelle unten.

Hinweis: Jedes Format ist für Connected Components Workbench (CCW) nativ (mit Ausnahme der Array-Formate). Aus diesem Grund könnte ein CCW-Tag-Name beim Referenzieren eines unteilbaren Datentyps kopiert und in das Tag-Adressfeld des Servers eingefügt werden und gültig sein.

Siehe auch: *Erweiterte Anwendungsfälle*

Format	Syntax
Array-Element	<Nativer Tag-Name> [dim1, dim2, dim3]
Array mit Offset*	<Nativer Tag-Name> {Spaltenanzahl} <Nativer Tag-Name> {Zeilenanzahl}{Spaltenanzahl}
Array ohne Offset*	<Nativer Tag-Name> {Spaltenanzahl} <Nativer Tag-Name> {Zeilenanzahl}{Spaltenanzahl}
Bit	<Nativer Tag-Name>.bit <Nativer Tag-Name>.[Bit]
Standard	<Nativer Tag-Name>
Zeichenfolge	<Nativer Tag-Name>

*Da diese Formate möglicherweise mehr als ein Element anfordern, hängt die Reihenfolge, in der Array-Daten weitergeleitet werden, von der Dimension des Array-Tags ab. Beispiel: Wenn "Zeilen" multipliziert mit "Spalten" = 4 und das native Tag ein 3X3-Element-Array ist, handelt es sich bei den Elementen, die referenziert

werden, um array_tag [0,0], array_tag [0,1], array_tag [0,2] und array_tag [1,0] in dieser genauen Reihenfolge. Die Ergebnisse wären unterschiedlich, wenn es sich beim nativen Tag um ein 2X10-Element-Array handeln würde. Weitere Informationen dazu finden Sie unter Reihenfolge von Array-Daten.

Erweiterte Adressformate

Array-Element

Mindestens 1 Dimension (jedoch nicht mehr als 3) muss angegeben sein.

Syntax	Beispiel	Hinweise
<Nativer Tag-Name> [dim1]	tag_1 [5]	k.A.
<Nativer Tag-Name> [dim1, dim2]	tag_1 [2, 3]	k.A.
<Nativer Tag-Name> [dim1, dim2, dim3]	tag_1 [2, 58, 547]	k.A.

Array mit Offset

Da diese Klasse möglicherweise mehr als ein Element anfordert, hängt die Reihenfolge, in der Array-Daten weitergeleitet werden, von der Dimension des Array-Tags ab.

Syntax	Beispiel	Hinweise
<Nativer Tag-Name> [Offset] {Spaltenanzahl}	tag_1 [5] {8}	Die Anzahl der Elemente zum Lesen/Schreiben entspricht der Anzahl Zeilen multipliziert mit der Anzahl Spalten. Wenn keine Zeilen angegeben werden, liegt die Zeilenanzahl standardmäßig bei 1. Mindestens 1 Element des Arrays muss adressiert werden. Das Array beginnt bei einem Null-Offset (Array-Index gleich 0 für alle Dimensionen).
<Nativer Tag-Name> [Offset] {Zeilenanzahl} {Spaltenanzahl}	tag_1[5] {2} {4}	

Hinweis: Wenn Zeilen*Spalten = 4 und das native Tag ein 3X3-Element-Array ist, handelt es sich bei den Elementen, die referenziert werden, um array_tag [0,0], array_tag [0,1], array_tag [0,2] und array_tag [1,0] in dieser genauen Reihenfolge. Die Ergebnisse wären unterschiedlich, wenn es sich beim nativen Tag um ein 2X10-Element-Array handeln würde.

Array ohne Offset

Da diese Klasse möglicherweise mehr als ein Element anfordert, hängt die Reihenfolge, in der Array-Daten weitergeleitet werden, von der Dimension des Array-Tags ab.

Syntax	Beispiel	Hinweise
<Nativer Tag-Name> {Spaltenanzahl}	tag_1 {8}	Die Anzahl der Elemente zum Lesen/Schreiben entspricht der Anzahl Zeilen multipliziert mit der Anzahl Spalten. Wenn keine Zeilen angegeben werden, liegt die Zeilenanzahl standardmäßig bei 1. Mindestens 1 Element des Arrays muss adressiert werden. Das Array beginnt bei einem Null-Offset (Array-Index gleich 0 für alle Dimensionen).
<Nativer Tag-Name> {Zeilenanzahl}{Spaltenanzahl}	tag_1 {2} {4}	

Hinweis: Beispiel: Wenn Zeilen*Spalten = 4 und das native Tag ein 3X3-Element-Array ist, handelt es sich bei den Elementen, die referenziert werden, um array_tag [0,0], array_tag [0,1], array_tag [0,2] und array_tag [1,0] in dieser genauen Reihenfolge. Die Ergebnisse wären unterschiedlich, wenn es sich beim nativen Tag um ein 2X10-Element-Array handeln würde.

Bit

Syntax	Beispiel	Hinweise
<Nativer Tag-Name> . bit	tag_1 . 0	k.A.
<Nativer Tag-Name>. [Bit]	tag_1 . [0]	k.A.

Standard

Syntax	Beispiel	Hinweise
<Nativer Tag-Name>	tag_1	k.A.

String

Syntax	Beispiel	Hinweise
<Nativer Tag-Name>	tag_1	Die Anzahl Zeichen zum Lesen/Schreiben entspricht der Zeichenfolgenlänge und muss mindestens gleich 1 sein.

Informationen zum Referenzieren von Elementen für Arrays mit 1, 2 und 3 Dimensionen finden Sie unter Reihenfolge von Array-Daten.

Tag-Umfang

Der Umfang von Variablen kann für ein Programm lokal oder für einen Controller global sein.

- Lokale Variablen werden einem bestimmten Programm im Projekt zugewiesen; sie stehen nur diesem Programm zur Verfügung.
- Globale Variablen gehören zum Controller im Projekt; sie stehen jedem Programm im Projekt zur Verfügung.

Lokale Variablen

Auf lokale Variablen (Tags mit Programm-Umfang) kann über den Kommunikations-Port des Controllers nicht direkt zugegriffen werden, sodass sie innerhalb des Treibers nicht direkt unterstützt werden. Wenn Zugriff erforderlich ist, schneiden Sie die Tags in der Tabelle mit den lokalen Variablen aus und fügen Sie sie in der Tabelle mit den globalen Variablen ein.

Globale Variablen

Bei globalen Variablen (dem Controller zugewiesene Tags) handelt es sich um native Tags, die über globalen Umfang im Controller verfügen. Jedes Programm bzw. jede Aufgabe kann auf globale Tags zugreifen. Die Anzahl der Möglichkeiten, wie ein globales Tag referenziert werden kann, ist jedoch von seinem nativen Datentyp und dem verwendeten Adressformat abhängig.

Benutzerdefinierte Datentypen

Benutzer können ggf. eindeutige Datentypen (z.B. STRING) mit 12 statt 80 Zeichen erstellen. Diese benutzerdefinierten Datentypen werden möglicherweise als lokale oder globale Variablen verwendet.

Strukturierte Variablen

Es gibt keine strukturierten Variablen in Micro800-Controllern. Benutzer können ggf. eindeutige Datentypen erstellen, aber jedes Mitglied muss einen eindeutigen Namen haben.

Adressieren unteilbarer Datentypen

In der Tabelle unten sind empfohlene Verwendungs- und Adressierungsmöglichkeiten für jeden nativen Datentyp unter Angabe der verfügbaren Adressformate enthalten. Wenn Sie erweiterte Adressierungsmöglichkeiten für jeden Datentyp benötigen, klicken Sie auf **Erweitert**.

Hinweis: Leere Zellen weisen nicht zwangsläufig auf mangelnde Unterstützung hin.

BOOL

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Boolean	Boolean (BOOL - 1-dimensionales Array)	Boolean-Array (BOOL - 1-dimensionales Array)		
Beispiel	BOOLTAG	BOOLARR[0]	BOOLARR[0]{32}		

SINT, USINT und BYTE

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Byte, Char	Byte, Char	Byte-Array, Char-Array (SINT - 1-, 2-, 3-dimensionales Array)	Boolean (Bit in SINT)	
Beispiel	SINTTAG	SINTARR[0]	SINTARR[0]{4}	SINTTAG.0	

INT, UINT und WORD

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Word, Short	Word, Short	Word-Array, Short Array (INT - 1, 2, 3-dimensionales Array)	Boolean (Bit in INT)	
Beispiel	INTTAG	INTARR[0]	INTARR[0]{4}	INTTAG.0	

DINT, UDINT und DWORD

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	DWord, Long	DWord, Long	DWord-Array, Long Array	Boolean (Bit in DINT)	
Beispiel	DINTTAG	DINTARR[0]	DINTARR[0]{4}	DINTTAG.0	

LINT, ULINT und LWORD

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Double, Date	Double, Date	Double-Array		
Beispiel	LINTTAG	LINTARR[0]	LINTARR[0]{4}		

REAL

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Float	Float	Float-Array		
Beispiel	REALTAG	REALARR[0]	REALARR[0]{4}		

LREAL

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Double	Double	Double-Array		
Beispiel	LREALTAG	LREALARR[0]	LREALARR[0]{4}		

SHORT_STRING

Tag	Standard	Array-Element	Array mit/ohne Offset	Bit	String
Datentyp Erweitert	Zeichenfolge	String			
Beispiel	STRINGTAG	STRINGARR[0]			

Siehe auch: Adressformate

Adressierung Strukturierte Datentypen

Strukturen können nicht auf Strukturebene referenziert werden: Nur die unteilbaren Strukturmitglieder können adressiert werden. Weitere Informationen dazu finden Sie in den Beispielen unten.

Natives Tag
MyTimer @ TIMER

Gültiges Client/Server-Tag
Adresse = MyTimer.ACC
Datentyp = DWord

Ungültiges Client/Server-Tag
Adresse = MyTimer
Datentyp = ??

Reihenfolge von Array-Daten

Eindimensionale Arrays – Array [dim1]

1-dimensionale Array-Daten werden dem bzw. vom Controller in aufsteigender Reihenfolge weitergeleitet.

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

Beispiel: Array mit 3 Elementen
array [0]
array [1]
array [2]

Zweidimensionale Arrays – Array [dim1, dim2]

2-dimensionale Array-Daten werden dem bzw. vom Controller in aufsteigender Reihenfolge weitergeleitet.

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

Beispiel: Array mit 3x3 Elementen
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]

Dreidimensionale Arrays – Array [dim1, dim2, dim3]

3-dimensionale Array-Daten werden dem bzw. vom Controller in aufsteigender Reihenfolge weitergeleitet.

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

Beispiel: Array mit 3x3x3 Elementen
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]

Erweiterte Anwendungsfälle

Wenn Sie weitere Informationen zu erweiterten Anwendungsfällen für einen bestimmten unteilbaren Datentyp erhalten möchten, wählen Sie eine Verknüpfung in der Liste unten aus.

BOOL
SINT, USINT und BYTE
INT, UINT und WORD
DINT, UDINT und DWORD
LINT, ULINT und LWORD
REAL
LREAL
SHORT_STRING

BOOL

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Boolean	Das native Tag muss ein eindimensionales Array sein.
Array mit Offset	Boolean-Array	1. Das native Tag muss ein eindimensionales Array sein. 2. Das Offset muss auf einer 32-Bit-Grenze liegen. 3. Die Anzahl der Elemente muss einem Faktor von 32 entsprechen.
Array ohne Offset	Boolean-Array	1. Das native Tag muss ein eindimensionales Array sein. 2. Die Anzahl der Elemente muss einem Faktor von 32 entsprechen.
Bit	Boolean	1. Das native Tag muss ein eindimensionales Array sein. 2. Der Bereich ist von 0 bis 31 begrenzt.
Standard	Boolean, Byte, Char, Word, Short, BCD, DWord, Long, LBCD, Float*	Keine
Zeichenfolge	Wird nicht unterstützt.	

*Der Float-Wert entspricht dem Nennwert des nativen Tags in Gleitkommazahlform (Nicht-IEEE-Gleitkommazahl).

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle dar.

Unteilbares BOOL-Tag - booltag = Wahr

Server-Tag-Adresse	Format	Datentyp	Hinweise
booltag	Standard	Boolean	Wert = Wahr
booltag	Standard	Byte	Wert = 1
booltag	Standard	Word	Wert = 1
booltag	Standard	DWord	Wert = 1
booltag	Standard	Float	Wert = 1.0
booltag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array.
booltag [3]	Array-Element	Word	Ungültig: Tag ist kein Array.
booltag {1}	Array ohne Offset	Word	Ungültig: Nicht unterstützt.
booltag {1}	Array ohne Offset	Boolean	Ungültig: Nicht unterstützt.
booltag [3] {32}	Array mit Offset	Boolean	Ungültig: Tag ist kein Array.
booltag . 3	Bit	Boolean	Ungültig: Tag ist kein Array.
booltag / 1	Zeichenfolge	String	Ungültig: Nicht unterstützt.
booltag / 4	Zeichenfolge	String	Ungültig: Nicht unterstützt.

BOOL-Array-Tag - bitarraytag = [0,1,0,1]

Server-Tag-Adresse	Format	Datentyp	Hinweise
bitarraytag	Standard	Boolean	Ungültig: Tag kann kein Array sein.
bitarraytag	Standard	Byte	Ungültig: Tag kann kein Array sein.
bitarraytag	Standard	Word	Ungültig: Tag kann kein Array sein.
bitarraytag	Standard	DWord	Ungültig: Tag kann kein Array sein.
bitarraytag	Standard	Float	Ungültig: Tag kann kein Array sein.
bitarraytag [3]	Array-Element	Boolean	Wert = Wahr
bitarraytag [3]	Array-Element	Word	Ungültig: Fehlerhafter Datentyp.
bitarraytag {3}	Array ohne Offset	Word	Ungültig: Tag kann kein Array sein.
bitarraytag {1}	Array ohne Offset	Word	Ungültig: Tag kann kein Array sein.
bitarraytag {1}	Array ohne Offset	Boolean	Ungültig: Array-Größe muss einem Faktor von 32 entsprechen.
bitarraytag {32}	Array ohne Offset	Boolean	Wert = [0,1,0,1,...]
bitarraytag [3] {32}	Array mit Offset	Boolean	Offset muss an 32-Bit-Grenze beginnen.
bitarraytag[0]{32}	Array mit Offset	Boolean	Wert = [0,1,0,1,...]
bitarraytag[32]{64}	Array mit Offset	Boolean	Syntax gültig. Element ist außerhalb des zulässigen Bereichs.
bitarraytag . 3	Bit	Boolean	Wert = Wahr
bitarraytag / 1	Zeichenfolge	String	Ungültig: Nicht unterstützt.
bitarraytag / 4	Zeichenfolge	String	Ungültig: Nicht unterstützt.

SINT, USINT und BYTE

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Byte, Char Word, Short, BCD DWord, Long, LBCD Float***	Das native Tag muss ein Array sein.
Array mit Offset	Byte-Array, Char-Array, Word-Array, Short-Array, BCD-Array**, DWord-Array, Long-Array, LBCD-Array**, Float-Array**,***	Das native Tag muss ein Array sein.
Array ohne Offset	Boolean-Array Byte-Array, Char-Array, Word-Array, Short-Array, BCD-Array**, DWord-Array, Long-Array, LBCD-Array**, Float-Array**,***	1. Verwenden Sie diesen Fall, damit die Bit innerhalb von SINT in Array-Form vorliegen. Dies ist kein Array von SINTs in boolescher Schreibweise. 2. Gilt nur für Bit innerhalb SINT. Beispiel: tag_1.0{8}. 3. Die Summe aus .bit und Array-Größe darf 8 Bit nicht überschreiten. Beispiel: tag_1.1{8} überschreitet SINT, tag_1.0{8} nicht. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein.
Bit	Boolean	1. Der Bereich ist von 0 bis 7 begrenzt. 2. Wenn das native Tag ein Array ist, muss der Bit-Klassenreferenz eine Array-Element-Klassenreferenz vorangestellt werden. Beispiel: tag_1[2,2,3].0.
Standard	Boolean*, Byte, Char, Word, Short, BCD, DWord, Long, LBCD, Float***	Keine
Zeichenfolge	String	1. Wenn auf ein einzelnes Element zugegriffen wird, muss das native Tag kein Array sein. **Hinweis:** Der Wert der Zeichenfolge ist das ASCII-Äquivalent des SINT-Werts. Beispiel: SINT = 65 (dezimal) = "A". 2. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein. Der Wert der Zeichenfolge ist das mit Null beendete ASCII-Äquivalent aller SINTs in der Zeichenfolge. 1 Zeichen in Zeichenfolge = 1 SINT.

*Werte ungleich Null sind auf "Wahr" fixiert.
**Jedes Element des Arrays entspricht einem Element im SINT-Array. Arrays sind nicht gepackt.
***Der Float-Wert entspricht dem Nennwert des nativen Tags in Gleitkommazahlform (Nicht-IEEE-Gleitkommazahl).

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle für SINT, USINT und BYTE dar.

Unteilbares Tag für SINT, USINT und BYTE - sinttag = 122 (dezimal)

Server-Tag-Adresse	Format	Datentyp	Hinweise
sinttag	Standard	Boolean	Wert = Wahr
sinttag	Standard	Byte	Wert = 122
sinttag	Standard	Word	Wert = 122
sinttag	Standard	DWord	Wert = 122
sinttag	Standard	Float	Wert = 122.0
sinttag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array. Boolean ist auch ungültig.
sinttag [3]	Array-Element	Byte	Ungültig: Tag ist kein Array.
sinttag {3}	Array ohne Off-set	Byte	Ungültig: Tag ist kein Array.
sinttag {1}	Array ohne Off-set	Byte	Wert = [122]
sinttag {1}	Array ohne Off-set	Boolean	Ungültig: Fehlerhafter Datentyp.
sinttag [3] {1}	Array mit Offset	Byte	Ungültig: Tag ist kein Array.
sinttag . 3	Bit	Boolean	Wert = Wahr
sinttag . 0 {8}	Array ohne Off-set	Boolean	Wert = [0,1,0,1,1,1,1,0] Bit-Wert von 122
sinttag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
sinttag / 4	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

SINT, USINT und BYTE - Array-Tag - sintarraytag [4,4] = [[83,73,78,84],[5,6,7,8],[9,10,11,12], [13,14,15,16]]

Server-Tag-Adresse	Format	Datentyp	Hinweise
sintarraytag	Standard	Boolean	Ungültig: Tag kann kein Array sein.
sintarraytag	Standard	Byte	Ungültig: Tag kann kein Array sein.
sintarraytag	Standard	Word	Ungültig: Tag kann kein Array sein.
sintarraytag	Standard	DWord	Ungültig: Tag kann kein Array sein.
sintarraytag	Standard	Float	Ungültig: Tag kann kein Array sein.
sintarraytag [3]	Array-Element	Byte	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.
sintarraytag [1,3]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
sintarraytag [1,3]	Array-Element	Byte	Wert = 8
sintarraytag {10}	Array ohne Off-set	Byte	Wert = [83,73,78,84,5,6,7,8,9,10]
sintarraytag {2} {5}	Array ohne Off-set	Word	Wert = [83,73,78,84,5] [6,7,8,9,10]
sintarraytag {1}	Array ohne Off-set	Byte	Wert = 83
sintarraytag {1}	Array ohne Off-set	Boolean	Ungültig: Fehlerhafter Datentyp.
sintarraytag [1,3] {4}	Array mit Off-set	Byte	Wert = [8,9,10,11]
sintarraytag . 3	Bit	Boolean	Ungültig: Tag muss unteilbare Position referenzieren.
sintarraytag [1,3] . 3	Bit	Boolean	Wert = 1
sintarraytag [1,3] . 0	Array ohne Off-	Boolean	Wert = [0,0,0,1,0,0,0,0]

Server-Tag-Adresse	Format	Datentyp	Hinweise
{8}	set		
sintarraytag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
sintarraytag / 4	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

INT, UINT und WORD

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Byte, Char** Word, Short, BCD DWord, Long, LBCD Float****	Das native Tag muss ein Array sein.
Array mit Offset	Byte-Array, Char-Array** Word-Array, Short-Array, BCD-Array DWord-Array, Long-Array, LBCD-Array*** Float-Array ***,****	Das native Tag muss ein Array sein.
Array ohne Offset	Boolean-Array Byte-Array, Char-Array**, Word-Array, Short-Array, BCD-Array, DWord-Array, Long-Array, LBCD-Array***, Float-Array***,****	1. Verwenden Sie diesen Fall, damit die Bit innerhalb von INT in Array-Form vorliegen. Dies ist kein Array von INTs in boolescher Schreibweise. 2. Gilt nur für Bit innerhalb INT. Beispiel: tag_1.0{16}. 3. Die Summe aus .bit und Array-Größe darf 16 Bit nicht überschreiten. Beispiel: tag_1.1{16} überschreitet INT, tag_1.0{16} nicht. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein.
Bit	Boolean	1. Der Bereich ist von 0 bis 15 begrenzt. 2. Wenn das native Tag ein Array ist, muss der Bit-Klassenreferenz eine Array-Element-Klassenreferenz vorangestellt werden. Beispiel: tag_1[2,2,3].0.
Standard	Boolean*, Byte, Char**, Word, Short, BCD, DWord, Long, LBCD, Float****	Keine.
Zeichenfolge	String	1. Wenn auf ein einzelnes Element zugegriffen wird, muss das native Tag kein Array sein. Hinweis: Der Wert der Zeichenfolge ist das ASCII-Äquivalent des INT-Werts (auf 255 fixiert). Beispiel: INT = 65 (dez) = "A". 2. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein. Der Wert der Zeichenfolge ist das mit Null beendete ASCII-Äquivalent aller INTs (auf 255 fixiert) in der Zeichenfolge.

Format	Unterstützte Datentypen	Hinweise
		1 Zeichen in Zeichenfolge = 1 INT, auf 255 fixiert. **Hinweis:** INT-Zeichenfolgen sind nicht gepackt. Verwenden Sie für eine größere Effizienz stattdessen SINT-Zeichenfolgen oder die STRING-Struktur.

*Werte ungleich Null sind auf "Wahr" fixiert.
**Werte, die 255 überschreiten, sind auf 255 fixiert.
*** Jedes Element des Arrays entspricht einem Element im INT-Array. Arrays sind nicht gepackt.
****Der Float-Wert entspricht dem Nennwert des nativen Tags in Gleitkommazahlform (Nicht-IEEE-Gleitkommazahl).

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle für INT, UINT und WORD dar.

Unteilbares Tag für INT, UINT und WORD - inttag = 65534 (dezimal)

Server-Tag-Adresse	Klasse	Datentyp	Hinweise
inttag	Standard	Boolean	Wert = Wahr
inttag	Standard	Byte	Wert = 255
inttag	Standard	Word	Wert = 65534
inttag	Standard	DWord	Wert = 65534
inttag	Standard	Float	Wert = 65534.0
inttag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array. Boolean ist auch ungültig.
inttag [3]	Array-Element	Word	Ungültig: Tag ist kein Array.
inttag {3}	Array ohne Offset	Word	Ungültig: Tag ist kein Array.
inttag {1}	Array ohne Offset	Word	Wert = [65534]
inttag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
inttag [3] {1}	Array mit Offset	Word	Ungültig: Tag ist kein Array.
inttag . 3	Bit	Boolean	Wert = Wahr
inttag . 0 {16}	Array ohne Offset	Boolean	Wert = [0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] Bit-Wert von 65534
inttag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
inttag / 4	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

Array-Tag für INT, UINT und WORD - intarraytag [4,4] = [[73,78,84,255],[256,257,258,259],[9,10,11,12],[13,14,15,16]]

Server-Tag-Adresse	Klasse	Datentyp	Hinweise
intarraytag	Standard	Boolean	Ungültig: Tag kann kein Array sein.
intarraytag	Standard	Byte	Ungültig: Tag kann kein Array sein.
intarraytag	Standard	Word	Ungültig: Tag kann kein Array sein.
intarraytag	Standard	DWord	Ungültig: Tag kann kein Array sein.
intarraytag	Standard	Float	Ungültig: Tag kann kein Array sein.
intarraytag [3]	Array-Element	Word	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.

Server-Tag-Adresse	Klasse	Datentyp	Hinweise
intarraytag [1,3]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
intarraytag [1,3]	Array-Element	Word	Wert = 259
intarraytag {10}	Array ohne Offset	Byte	Wert = [73,78,84,255,255,255,255,255,9,10]
intarraytag {2} {5}	Array ohne Offset	Word	Wert = [73,78,84,255,256] [257,258,259,9,10]
intarraytag {1}	Array ohne Offset	Word	Wert = 73
intarraytag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
intarraytag [1,3] {4}	Array mit Offset	Word	Wert = [259,9,10,11]
intarraytag . 3	Bit	Boolean	Ungültig: Tag muss unteilbare Position referenzieren.
intarraytag [1,3] . 3	Bit	Boolean	Wert = 0
intarraytag [1,3] . 0 {16}	Array ohne Offset	Boolean	Wert = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit-Wert für 259
intarraytag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
intarraytag / 3	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

DINT, UDINT und DWORD

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Byte, Char**, Word, Short, BCD***, DWord, Long, LBCD, Float****	Das native Tag muss ein Array sein.
Array mit Offset	Byte-Array, Char-Array** Word-Array, Short-Array, BCD-Array*** DWord-Array, Long-Array, LBCD-Array Float-Array****	Das native Tag muss ein Array sein.
Array ohne Offset	Boolean-Array	1. Verwenden Sie diesen Fall, damit die Bit innerhalb von DINT in Array-Form vorliegen. Dies ist kein Array von DINTs in boolescher Schreibweise. 2. Gilt nur für Bit innerhalb DINT. • Beispiel: tag_1.0{32}. Die Summe aus .bit und Array-Größe darf 32 Bit nicht überschreiten. • Beispiel: tag_1.1{32} überschreitet DINT, tag_1.0{32} nicht. • Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein.
	Byte-Array, Char-Array** Word-Array, Short-Array, BCD-Array*** DWord-Array, Long-Array, LBCD-Array Float-Array****	

Format	Unterstützte Datentypen	Hinweise
Bit	Boolean	1. Der Bereich ist von 0 bis 31 begrenzt. 2. Wenn das native Tag ein Array ist, muss der Bit-Klassenreferenz eine Array-Element-Klassenreferenz vorangestellt werden. Beispiel: tag_1 [2,2,3].0.
Standard	Boolean*, Byte, Char**, Word, Short, BCD***, DWord, Long, LBCD, Float****	Keine.
Zeichenfolge	String	1. Wenn auf ein einzelnes Element zugegriffen wird, muss das native Tag kein Array sein. **Hinweis:** Der Wert der Zeichenfolge ist das ASCII-Äquivalent des DINT-Werts (auf 255 fixiert). Beispiel: SINT = 65 (dezimal) = "A". 2. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein. Der Wert der Zeichenfolge ist das mit Null beendete ASCII-Äquivalent aller DINTs (auf 255 fixiert) in der Zeichenfolge. 1 Zeichen in Zeichenfolge = 1 DINT, auf 255 fixiert. **Hinweis:** DINT-Zeichenfolgen sind nicht gepackt. Verwenden Sie für eine größere Effizienz stattdessen SINT-Zeichenfolgen oder die STRING-Struktur.

*Werte ungleich Null sind auf "Wahr" fixiert.
**Werte, die 255 überschreiten, sind auf 255 fixiert.
***Werte, die 65535 überschreiten, sind auf 65535 fixiert.
****Der Float-Wert entspricht dem Nennwert des nativen Tags in Gleitkommazahlform (Nicht-IEEE-Gleitkommazahl).

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle für DINT, UDINT und DWORD dar.

Unteilbares Tag für DINT, UDINT und DWORD - dinttag = 70000 (dezimal)

Server-Tag-Adresse	Format	Datentyp	Hinweise
dinttag	Standard	Boolean	Wert = Wahr
dinttag	Standard	Byte	Wert = 255
dinttag	Standard	Word	Wert = 65535
dinttag	Standard	DWord	Wert = 70000
dinttag	Standard	Float	Wert = 70000.0
dinttag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array. Boolean ist auch ungültig.
dinttag [3]	Array-Element	DWord	Ungültig: Tag ist kein Array.
dinttag {3}	Array ohne Offset	DWord	Ungültig: Tag ist kein Array.
dinttag {1}	Array ohne Offset	DWord	Wert = [70000]
dinttag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.

Server-Tag-Adresse	Format	Datentyp	Hinweise
dintag [3] {1}	Array mit Offset	DWord	Ungültig: Tag ist kein Array.
dinttag . 3	Bit	Boolean	Wert = falsch
dinttag . 0 {32}	Array ohne Offset	Boolean	Wert = [0,0,0,0,1,1,1,0,1,0,0,0,1,0,0,0,1,0,...0] Bit-Wert für 70000
dinttag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
dinttag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

Array-Tag für DINT, UDINT und DWORD - dintarraytag [4,4] = [[68,73,78,84],[256,257,258,259], [9,10,11,12],[13,14,15,16]]

Server-Tag-Adresse	Format	Datentyp	Hinweise
dintarraytag	Standard	Boolean	Ungültig: Tag kann kein Array sein.
dintarraytag	Standard	Byte	Ungültig: Tag kann kein Array sein.
dintarraytag	Standard	Word	Ungültig: Tag kann kein Array sein.
dintarraytag	Standard	DWord	Ungültig: Tag kann kein Array sein.
dintarraytag	Standard	Float	Ungültig: Tag kann kein Array sein.
dintarraytag [3]	Array-Element	DWord	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.
dintarraytag [1,3]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
dintarraytag [1,3]	Array-Element	DWord	Wert = 259
dintarraytag {10}	Array ohne Offset	Byte	Wert = [68,73,78,84,255,255,255,255,9,10]
dintarraytag {2}{5}	Array ohne Offset	DWord	Wert = [68,73,78,84,256] [257,258,259,9,10]
dintarraytag {1}	Array ohne Offset	DWord	Wert = 68
dintarraytag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
dintarraytag [1,3]{4}	Array mit Offset	DWord	Wert = [259,9,10,11]
dintarraytag . 3	Bit	Boolean	Ungültig: Tag muss unteilbare Position referenzieren.
dintarraytag [1,3] . 3	Bit	Boolean	Wert = 0
dintarraytag [1,3] .0 {32}	Array ohne Offset	Boolean	Wert = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit-Wert für 259
dintarraytag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
dintarraytag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

LINT, ULINT und LWORD

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Double* Date**	Das native Tag muss ein Array sein.
Array mit Offset	Double-Array*	Das native Tag muss ein Array sein.
Array ohne Offset	Double-Array*	Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein.

Format	Unterstützte Datentypen	Hinweise
Bit	Wird nicht unterstützt.	Wird nicht unterstützt.
Standard	Double* Date**	Keine.
Zeichenfolge	Wird nicht unterstützt.	Wird nicht unterstützt.

*Der Double-Wert entspricht dem Nennwert des nativen Tags in Gleitkommazahlform (Nicht-IEEE-Gleitkommazahl).
**Datumswerte liegen in UTC-Zeit und nicht in lokalisierter Zeit vor.

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle für LINT, ULINT und LWORD dar.

Unteilbares Tag für LINT, ULINT und LWORD - linttag = 2007-01-01T16:46:40.000 (Datum) == 1.16767E+15 (dezimal)

Server-Tag-Adresse	Format	Datentyp	Hinweise
linttag	Standard	Boolean	Ungültig: Boolean wird nicht unterstützt.
linttag	Standard	Byte	Ungültig: Byte wird nicht unterstützt.
linttag	Standard	Word	Ungültig: Word wird nicht unterstützt.
linttag	Standard	Double	Wert = 1.16767E+15
linttag	Standard	Date	Wert = 2007-01-01T16:46:40.000*
linttag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array. Boolean ist auch ungültig.
linttag [3]	Array-Element	Double	Ungültig: Tag ist kein Array.
linttag {3}	Array ohne Offset	Double	Ungültig: Tag ist kein Array.
linttag {1}	Array ohne Offset	Double	Wert = [1.16767E+15]
linttag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
linttag [3] {1}	Array mit Offset	Double	Ungültig: Tag ist kein Array.
linttag . 3	Bit	Boolean	Ungültig: Syntax-/Datentyp nicht unterstützt.
linttag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

*Datumswerte liegen in UTC-Zeit und nicht in lokalisierter Zeit vor.

Array-Tag für LINT, ULINT und LWORD -
dintarraytag [2,2] = [0, 1.16767E+15],[9.4666E+14, 9.46746E+14] Dabei gilt:
1.16767E+15 == 2007-01-01T16:46:40.000 (Datum)
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

Server-Tag-Adresse	Format	Datentyp	Hinweise
lintarraytag	Standard	Boolean	Ungültig: Boolean nicht unterstützt.
lintarraytag	Standard	Byte	Ungültig: Byte wird nicht unterstützt.
lintarraytag	Standard	Word	Ungültig: Word wird nicht unterstützt.
lintarraytag	Standard	Double	Ungültig: Tag kann kein Array sein.

Server-Tag-Adresse	Format	Datentyp	Hinweise
lintarraytag	Standard	Date	Ungültig: Tag kann kein Array sein.
lintarraytag [1]	Array-Element	Double	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.
lintarraytag [1,1]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
lintarraytag [1,1]	Array-Element	Double	Wert = 9.46746E+14
lintarraytag [1,1]	Array-Element	Date	Wert = 2000-01-01T17:00:00.000*
lintarraytag {4}	Array ohne Offset	Double	Wert = [0, 1.16767E+15, 9.4666E+14, 9.46746E+14]
lintarraytag {2} {2}	Array ohne Offset	Double	Wert = [0, 1.16767E+15][9.4666E+14, 9.46746E+14]
lintarraytag {4}	Array ohne Offset	Date	Ungültig: Date-Array wird nicht unterstützt.
lintarraytag {1}	Array ohne Offset	Double	Wert = 0
lintarraytag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
lintarraytag [0,1] {2}	Array mit Offset	Double	Wert = [1.16767E+15, 9.4666E+14]
lintarraytag . 3	Bit	Boolean	Ungültig: Syntax-/Datentyp nicht unterstützt.
lintarraytag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

*Datumswerte liegen in UTC-Zeit und nicht in lokalisierter Zeit vor.

REAL

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Byte, Char** Word, Short, BCD*** DWord, Long, LBCD Float****	Das native Tag muss ein Array sein.
Array mit Offset	Byte-Array, Char-Array** Word-Array, Short-Array, BCD-Array*** DWord-Array, Long-Array, LBCD-Array Float-Array****	Das native Tag muss ein Array sein.
Array ohne Offset	Boolean-Array Byte-Array, Char-Array**, Word-Array, Short-Array, BCD-Array***, DWord-Array, Long-Array, LBCD-Array, Float-Array****	1. Verwenden Sie diesen Fall, damit die Bit innerhalb von REAL in Array-Form vorliegen. Dies ist kein Array von REALs in boolescher Schreibweise. 2. Gilt nur für Bit innerhalb REAL. Beispiel: tag_1.0 {32}. 3. Die Summe aus .bit und Array-Größe darf 32 Bit nicht überschreiten. Beispiel: tag_1.1{32} überschreitet REAL, tag_1.0{32} nicht. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein.

Format	Unterstützte Datentypen	Hinweise
Bit	Boolean	1. Der Bereich ist von 0 bis 31 begrenzt. 2. Wenn das native Tag ein Array ist, muss der Bit-Klassenreferenz eine Array-Element-Klassenreferenz vorangestellt werden. Beispiel: tag_1 [2,2,3].0. **Hinweis:** Float ist auf DWord festgelegt, um das Referenzieren von Bit zu ermöglichen.
Standard	Boolean*, Byte, Char**, Word, Short, BCD***, DWord, Long, LBCD, Float****	Keine.
Zeichenfolge	String	1. Wenn auf ein einzelnes Element zugegriffen wird, muss das native Tag kein Array sein. **Hinweis:** Der Wert der Zeichenfolge ist das ASCII-Äquivalent des REAL-Werts (auf 255 fixiert). Beispiel: SINT = 65 (dezimal) = "A". 2. Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein. Der Wert der Zeichenfolge ist das mit Null beendete ASCII-Äquivalent aller REALs (auf 255 fixiert) in der Zeichenfolge. 1 Zeichen in Zeichenfolge = 1 REAL, auf 255 fixiert. REAL-Zeichenfolgen sind nicht gepackt. Verwenden Sie für eine größere Effizienz stattdessen SINT-Zeichenfolgen oder die STRING-Struktur.

*Werte ungleich Null sind auf "Wahr" fixiert.
**Werte, die 255 überschreiten, sind auf 255 fixiert.
***Werte, die 65535 überschreiten, sind auf 65535 fixiert.
****Der Float-Wert ist eine gültige IEEE-Gleitkommazahl mit einfacher Präzision.

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle dar.

Unteilbares REAL-Tag - realtag = 512.5 (dezimal)

Server-Tag-Adresse	Format	Datentyp	Hinweise
realtag	Standard	Boolean	Wert = Wahr
realtag	Standard	Byte	Wert = 255
realtag	Standard	Word	Wert = 512
realtag	Standard	DWord	Wert = 512
realtag	Standard	Float	Wert = 512.5
realtag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array. Boolean ist auch ungültig.
realtag [3]	Array-Element	DWord	Ungültig: Tag ist kein Array.
realtag {3}	Array ohne Offset	DWord	Ungültig: Tag ist kein Array.
realtag {1}	Array ohne Offset	Float	Wert = [512.5]

Server-Tag-Adresse	Format	Datentyp	Hinweise
realtag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
realtag [3] {1}	Array mit Offset	Float	Ungültig: Tag ist kein Array.
realtag . 3	Bit	Boolean	Wert = Wahr
realtag . 0 {32}	Array ohne Offset	Boolean	Wert = [0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,...0] Bit-Wert für 512
realtag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
realtag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

REAL-Array-Tag - 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]]

Server-Tag-Adresse	Format	Datentyp	Hinweise
realarraytag	Standard	Boolean	Ungültig: Tag kann kein Array sein.
realarraytag	Standard	Byte	Ungültig: Tag kann kein Array sein.
realarraytag	Standard	Word	Ungültig: Tag kann kein Array sein.
realarraytag	Standard	DWord	Ungültig: Tag kann kein Array sein.
realarraytag	Standard	Float	Ungültig: Tag kann kein Array sein.
realarraytag [3]	Array-Element	Float	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.
realarraytag [1,3]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
realarraytag [1,3]	Array-Element	Float	Wert = 259,8
realarraytag {10}	Array ohne Offset	Byte	Wert = [82,69,65,76,255,255,255,255,9,10]
realarraytag {2} {5}	Array ohne Offset	Float	Wert = [82.1,69.2,65.3,76.4,256.5] [257.6,258.7,259.8,9,10]
realarraytag {1}	Array ohne Offset	Float	Wert = 82.1
realarraytag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
realarraytag [1,3] {4}	Array mit Offset	Float	Wert = [259.8,9.0,10.0,11.0]
realarraytag . 3	Bit	Boolean	Ungültig: Tag muss unteilbare Position referenzieren.
realarraytag [1,3] . 3	Bit	Boolean	Wert = 0
realarraytag [1,3] . 0 {32}	Array ohne Offset	Boolean	Wert = [1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0] Bit-Wert für 259
realarraytag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
realarraytag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

LREAL

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Double*	Das native Tag muss ein Array sein.
Array mit Offset	Double-Array	Das native Tag muss ein Array sein.

Format	Unterstützte Datentypen	Hinweise
Array ohne Offset	Double-Array	Wenn auf mehr als ein einzelnes Element zugegriffen wird, muss das native Tag ein Array sein.
Bit	Boolean	Ungültig: Syntax-/Datentyp nicht unterstützt.
Standard	Double*	Keine.
Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

*Der Double-Wert ist eine gültige IEEE-Gleitkommazahl mit doppelter Präzision.

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle dar.

Unteilbares LREAL-Tag - lrealtag = 512.5 (dezimal)

Server-Tag-Adresse	Format	Datentyp	Hinweise
lrealtag	Standard	Boolean	Ungültig: Datentyp wird nicht unterstützt.
lrealtag	Standard	Byte	Ungültig: Datentyp wird nicht unterstützt.
lrealtag	Standard	Word	Ungültig: Datentyp wird nicht unterstützt.
lrealtag	Standard	DWord	Ungültig: Datentyp wird nicht unterstützt.
lrealtag	Standard	Double	Wert = 512.5
lrealtag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array und Boolean ist ungültig.
lrealtag [3]	Array-Element	DWord	Ungültig: Tag ist kein Array.
lrealtag {3}	Array ohne Offset	DWord	Ungültig: Tag ist kein Array.
lrealtag {1}	Array ohne Offset	Double	Wert = [512.5]
lrealtag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
lrealtag [3] {1}	Array mit Offset	Float	Ungültig: Tag ist kein Array.
lrealtag . 3	Bit	Boolean	Ungültig: Datentyp wird nicht unterstützt.
lrealtag . 0 {32}	Array ohne Offset	Boolean	Ungültig: Datentyp wird nicht unterstützt.
lrealtag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
lrealtag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

LREAL-Array-Tag - 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]]

Server-Tag-Adresse	Format	Datentyp	Hinweise
lrealarraytag	Standard	Boolean	Ungültig: Tag kann kein Array sein.
lrealarraytag	Standard	Byte	Ungültig: Tag kann kein Array sein.
lrealarraytag	Standard	Word	Ungültig: Tag kann kein Array sein.
lrealarraytag	Standard	DWord	Ungültig: Tag kann kein Array sein.
lrealarraytag	Standard	Double	Ungültig: Tag kann kein Array sein.
lrealarraytag [3]	Array-Element	Double	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.
lrealarraytag [1,3]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
lrealarraytag [1,3]	Array-Element	Double	Wert = 259,8
lrealarraytag {10}	Array ohne Off-	Byte	Ungültig: Datentyp wird nicht unterstützt.

Server-Tag-Adresse	Format	Datentyp	Hinweise
	set		
lrealarraytag {2} {5}	Array ohne Offset	Double	Wert = [82.1,69.2,65.3,76.4,256.5] [257.6,258.7,259.8,9,10]
lrealarraytag {1}	Array ohne Offset	Double	Wert = 82.1
lrealarraytag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
lrealarraytag [1,3] {4}	Array mit Offset	Double	Wert = [259.8,9.0,10.0,11.0]
lrealarraytag . 3	Bit	Boolean	Ungültig: Tag muss unteilbare Position referenzieren.
lrealarraytag [1,3] . 3	Bit	Boolean	Wert = 0
lrealarraytag [1,3] . 0 {32}	Array ohne Offset	Boolean	Ungültig: Syntax-/Datentyp nicht unterstützt.
lrealarraytag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.
lrealarraytag	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

SHORT_STRING

Weitere Informationen zum Format finden Sie unter Adressformate.

Format	Unterstützte Datentypen	Hinweise
Array-Element	Zeichenfolge	Das native Tag muss ein Array sein.
Array mit Offset	k.A.	k.A.
Array ohne Offset	k.A.	k.A.
Bit	k.A.	k.A.
Standard	Zeichenfolge	Die Länge der Zeichenfolge basiert auf der Längenkodierung innerhalb des nativen Tags. Wenn die Zeichenfolge nicht druckbare Zeichen enthält, sind diese in der Zeichenfolge enthalten.
Zeichenfolge	k.A.	Die Länge der Zeichenfolge muss in der Tag-Adresse angegeben werden.

Beispiele

Die hervorgehobenen Beispiele stellen allgemeine Anwendungsfälle dar.

Unteilbares SHORT_STRING-Tag - stringtag = "mystring"

Server-Tag-Adresse	Format	Datentyp	Hinweise
stringtag	Standard	Zeichenfolge	Wert = mystring.
stringtag	Standard	Byte	Ungültig: Byte wird nicht unterstützt.
stringtag	Standard	Word	Ungültig: Word wird nicht unterstützt.
stringtag [3]	Array-Element	Boolean	Ungültig: Tag ist kein Array und Boolean ist ungültig.
stringtag [3]	Array-Element	Double	Ungültig: Tag ist kein Array.
stringtag {3}	Array ohne Offset	Double	Ungültig: Tag ist kein Array.
stringtag {1}	Array ohne Offset	Double	Wert = [1.16767E+15].

Server-Tag-Adresse	Format	Datentyp	Hinweise
stringtag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
lintag [3] {1}	Array mit Offset	Double	Ungültig: Tag ist kein Array.
stringtag . 3	Bit	Boolean	Ungültig: Syntax-/Datentyp nicht unterstützt.
stringtag / 1	Zeichenfolge	String	Ungültig: Syntax-/Datentyp nicht unterstützt.

SHORT_STRING-Array-Tag - stringarraytag[2,2] = [eins,zwei].[drei,vier]

Server-Tag-Adresse	Format	Datentyp	Hinweise
stringarraytag	Standard	Boolean	Ungültig: Boolean nicht unterstützt.
stringarraytag	Standard	Byte	Ungültig: Byte wird nicht unterstützt.
stringarraytag	Standard	Word	Ungültig: Word wird nicht unterstützt.
stringarraytag	Standard	Double	Ungültig: Tag kann kein Array sein.
stringarraytag	Standard	Date	Ungültig: Tag kann kein Array sein.
stringarraytag [1]	Array-Element	Double	Ungültig: Server-Tag weist fehlende Adresse für Dimension 2 auf.
stringarraytag [1,1]	Array-Element	Boolean	Ungültig: Boolean nicht für Array-Elemente zulässig.
stringarraytag [1,1]	Array-Element	Zeichenfolge	Wert: "vier"
stringarraytag {4}	Array ohne Offset	Zeichenfolge	Ungültig: String-Array wird nicht unterstützt.
stringarraytag {2} {2}	Array ohne Offset	Zeichenfolge	Ungültig: String-Array wird nicht unterstützt.
stringarraytag {1}	Array ohne Offset	Boolean	Ungültig: Fehlerhafter Datentyp.
stringarraytag [0, 1] {2}	Array mit Offset	Zeichenfolge	Wert: "drei"
stringarraytag . 3	Bit	Boolean	Ungültig: Syntax-/Datentyp nicht unterstützt.
stringarraytag / 1	Zeichenfolge	String	Ungültig: Syntax wird nicht unterstützt.

Fehlercodes

In den folgenden Abschnitten sind Fehlercodes definiert, die im Ereignisprotokoll des Servers möglicherweise erfasst wurden. Weitere Informationen zu einem bestimmten Fehlercode erhalten Sie, wenn Sie eine Verknüpfung in der Liste unten auswählen.

Kapselungsprotokoll-Fehlercodes
CIP-Fehlercodes

Kapselungsprotokoll-Fehlercodes

Die folgenden Fehlercodes liegen in hexadezimaler Schreibweise vor.

Fehlercode	Description
0001	Befehl nicht verarbeitet.
0002	Speicher nicht für Befehl verfügbar.
0003	Schlecht gebildete oder unvollständige Daten.
0064	Ungültige Sitzungs-ID.
0065	Ungültige Länge des Headers.
0069	Angeforderte Protokollversion nicht unterstützt.
0070	Ungültige Ziel-ID.

CIP-Fehlercodes

Die folgenden Fehlercodes liegen in hexadezimaler Schreibweise vor.

Fehlercode	Description
0001	Verbindungsfehler.*
0002	Nicht genügend Ressourcen.
0003	Wert ungültig.
0004	IOI konnte nicht entziffert werden oder Tag ist nicht vorhanden.
0005	Unbekanntes Ziel.
0006	Angeforderte Daten würden nicht in Antwortpaket passen.
0007	Unterbrochene Verbindung.
0008	Nicht unterstützter Dienst.
0009	Fehler in Datensegment oder ungültiger Attributwert.
000A	Attributlistenfehler.
000B	Status bereits vorhanden.
000C	Objektmodellkonflikt.
000D	Objekt bereits vorhanden.
000E	Attribut nicht konfigurierbar.
000F	Berechtigung verweigert.
0010	Gerätestatuskonflikt.
0011	Antwort passt nicht.
0012	Fragment einfach.
0013	Nicht ausreichende Befehlsdaten/-parameter zur Ausführung des Diensts angegeben.
0014	Attribut nicht unterstützt.
0015	Zu viele Daten angegeben.
001A	Bridge-Anforderung zu groß.
001B	Bridge-Antwort zu groß.
001C	Attributliste nicht ausreichend.

Fehlercode	Description
001D	Ungültige Attributliste.
001E	Fehler bei eingebettetem Dienst.
001F	Fehler während Verbindung.**
0022	Ungültige Antwort erhalten.
0025	Schlüsselsegmentfehler.
0026	Die Anzahl der angegebenen IOI-Wörter entspricht nicht der IOI-Wortanzahl.
0027	Unerwartetes Attribut in Liste.

* **Siehe auch:** 0x0001 Erweiterte Fehlercodes
** **Siehe auch:** 0x001F Erweiterte Fehlercodes

Bestimmte Fehlercodes von Allen-Bradley

Fehlercode (hex)	Beschreibung
00FF	Allgemeiner Fehler*

* **Siehe auch:** 0x00FF Erweiterte Fehlercodes

Nicht aufgelistete Fehlercodes finden Sie in der Dokumentation von Rockwell Automation.

0x0001 Erweiterte Fehlercodes

Die folgenden Fehlercodes liegen in hexadezimaler Schreibweise vor.

Fehlercode	Beschreibung
0100	Verbindung wird verwendet.
0103	Transport wird nicht unterstützt.
0106	Besitzerkonflikt.
0107	Verbindung nicht gefunden.
0108	Ungültiger Verbindungstyp.
0109	Ungültige Verbindungsgröße.
0110	Modul nicht konfiguriert.
0111	EPR nicht unterstützt.
0114	Falsches Modul.
0115	Falscher Gerätetyp.
0116	Falsche Revision.
0118	Ungültiges Konfigurationsformat.
011A	Keine Verbindungen für Anwendung.
0203	Verbindungs-Timeout
0204	Nicht verbundenes Meldungs-Timeout.
0205	Parameterfehler für nicht verbundenes Senden.
0206	Nachricht zu groß.
0301	Kein Pufferspeicher.
0302	Bandbreite nicht verfügbar.
0303	Keine Screener verfügbar.
0305	Signaturübereinstimmung.
0311	Port nicht verfügbar.
0312	Verknüpfungsadresse nicht verfügbar.
0315	Ungültiger Segmenttyp.
0317	Verbindung nicht geplant.

Fehlercode	Beschreibung
0318	Eigene Verknüpfungsadresse ist ungültig.

- *Nicht aufgelistete Fehlercodes finden Sie in der Dokumentation von Rockwell Automation.*

0x001F Erweiterte Fehlercodes

Die folgenden Fehlercodes liegen in hexadezimaler Schreibweise vor.

Fehlercode	Beschreibung
0203	Timeout der Verbindung.

- *Nicht aufgelistete Fehlercodes finden Sie in der Dokumentation von Rockwell Automation.*

0x00FF Erweiterte Fehlercodes

Die folgenden Fehlercodes liegen in hexadezimaler Schreibweise vor.

Fehlercode	Beschreibung
2104	Adresse außerhalb des zulässigen Bereichs.
2105	Versuch, über das Ende des Datenobjekts hinaus zuzugreifen.
2106	Daten werden verwendet.
2107	Datentyp ist ungültig oder wird nicht unterstützt.

- *Nicht aufgelistete Fehlercodes finden Sie in der Dokumentation von Rockwell Automation.*

Ereignisprotokollmeldungen

Die folgenden Informationen betreffen Meldungen, die im Fensterbereich Ereignisprotokoll in der Hauptbenutzeroberfläche angezeigt werden. Informationen zum Filtern und Sortieren der Detailansicht Ereignisprotokoll finden Sie in der Serverhilfe. In der Serverhilfe sind viele allgemeine Meldungen enthalten, die also auch gesucht werden sollten. Im Allgemeinen werden die Art der Meldung (Information, Warnung) sowie Fehlerbehebungsinformationen bereitgestellt (sofern möglich).

Controller wird nicht unterstützt. | Händler-ID = <Händler>, Produkttyp = <Typ>, Produktcode = <Code>, Produktname = '<Produkt>'.

Fehlertyp:
Warnung

Der vom Gerät empfangene Frame enthält Fehler.

Fehlertyp:
Warnung

Mögliche Ursache:

1. Die Pakete sind falsch ausgerichtet (aufgrund der Verbindung/Trennung zwischen dem PC und dem Gerät).
2. Fehlerhafte Verbindungskabel zwischen den Geräten verursachen Störungen.
3. Eine falsche Frame-Größe wurde empfangen.
4. Keine TNS-Übereinstimmung.
5. Vom Gerät wurde ein ungültiger Antwortbefehl zurückgegeben.

Mögliche Lösung:
Zwar kann der Treiber ohne weitere Maßnahmen nach diesem Fehler wiederhergestellt werden, es besteht jedoch möglicherweise ein Problem bei der Verkabelung oder dem Gerät selbst, das behoben werden sollte.

Schreibanforderung für Tag ist aufgrund eines Framing-Fehlers fehlgeschlagen. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Mögliche Ursache:

1. Eine Schreibanforderung für das angegebene Tag ist aufgrund eines falschen Dienstcodes nach einer bestimmten Anzahl von Wiederholungen fehlgeschlagen.
2. Eine Schreibanforderung für das angegebene Tag ist nach einer bestimmten Anzahl von Wiederholungen fehlgeschlagen, da die empfangene Byte-Anzahl größer oder kleiner war als erwartet.

Mögliche Lösung:
Möglicherweise besteht ein Problem bei der Verkabelung oder dem Gerät selbst. Erhöhen Sie den Wert für 'Erneute Versuche', damit der Treiber mehr Möglichkeiten erhält, nach diesem Fehler wiederhergestellt zu werden.

Leseanforderung für Tag ist aufgrund eines Framing-Fehlers fehlgeschlagen. | Tag-Adresse = '<Adresse>'.

Fehlertyp:

Warnung

Mögliche Ursache:

1. Eine Leseanforderung für das angegebene Tag ist aufgrund eines falschen Dienstcodes nach einer bestimmten Anzahl von Wiederholungen fehlgeschlagen.
2. Eine Leseanforderung für das angegebene Tag ist nach einer bestimmten Anzahl von Wiederholungen fehlgeschlagen, da die empfangene Byte-Anzahl größer oder kleiner war als erwartet.

Mögliche Lösung:
Möglicherweise besteht ein Problem bei der Verkabelung oder dem Gerät selbst. Erhöhen Sie den Wert für 'Erneute Versuche', damit der Treiber mehr Möglichkeiten erhält, nach diesem Fehler wiederhergestellt zu werden.

Block-Leseanforderung ist aufgrund eines Framing-Fehlers fehlgeschlagen. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl> (Elemente).

Fehlertyp:
Warnung

Mögliche Ursache:

1. Eine Leseanforderung für das angegebene Tag ist aufgrund eines falschen Dienstcodes nach einer bestimmten Anzahl von Wiederholungen fehlgeschlagen.
2. Eine Leseanforderung für das angegebene Tag ist nach einer bestimmten Anzahl von Wiederholungen fehlgeschlagen, da die empfangene Byte-Anzahl größer oder kleiner war als erwartet.

Mögliche Lösung:
Möglicherweise besteht ein Problem bei der Verkabelung oder dem Gerät selbst. Erhöhen Sie den Wert für 'Erneute Versuche', damit der Treiber mehr Möglichkeiten erhält, nach diesem Fehler wiederhergestellt zu werden.

In Tag auf Gerät kann nicht geschrieben werden. | Tag-Adresse = '<Adresse>', CIP-Fehler = <Code>, erweiterter Fehler = <Code>.

Fehlertyp:
Warnung

Mögliche Ursache:
Während einer Schreibanforderung für das angegebene Tag wurde vom Gerät ein Fehler im CIP-Teil des Pakets zurückgegeben.

Mögliche Lösung:
Die Lösung hängt davon ab, welche Fehlercodes zurückgegeben wurden. Weitere Informationen finden Sie in den Definitionen der CIP- und erweiterten Codes.

Tag von Gerät kann nicht gelesen werden. | Tag-Adresse = '<Adresse>', CIP-Fehler = <Code>, erweiterter Fehler = <Code>.

Fehlertyp:
Warnung

Mögliche Ursache:
Während einer Leseanforderung für das angegebene Tag wurde vom Gerät ein Fehler im CIP-Teil des Pakets zurückgegeben.

Mögliche Lösung:

Die Lösung hängt davon ab, welche Fehlercodes zurückgegeben wurden. Weitere Informationen finden Sie in den Definitionen der CIP- und erweiterten Codes.

Block von Gerät kann nicht gelesen werden. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl>, CIP-Fehler = <Code>, erweiterter Fehler = <Code>.

Fehlertyp:
Warnung

Mögliche Ursache:
Während einer Block-Leseanforderung für das angegebene Tag wurde vom Gerät ein Fehler im CIP-Teil des Pakets zurückgegeben.

Mögliche Lösung:
Die Lösung hängt davon ab, welche Fehlercodes zurückgegeben wurden. Weitere Informationen finden Sie in den Definitionen der CIP- und erweiterten Codes.

In Tag auf Gerät kann nicht geschrieben werden. Controller-Tag-Datentyp ist unbekannt. | Tag-Adresse = '<Adresse>', unbekannter Datentyp = <Typ>.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Anforderung für das angegebene Tag ist fehlgeschlagen, da der Tag-Datentyp nicht unterstützt wird.

Mögliche Lösung:

1. Ändern Sie den Tag-Datentyp auf einen unterstützten Typ. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.
2. Wenden Sie sich an den technischen Support.

Tag kann nicht von Gerät gelesen werden. Controller-Tag-Datentyp ist unbekannt. Tag deaktiviert. | Tag-Adresse = '<Adresse>', unbekannter Datentyp = <Typ>.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Anforderung für das angegebene Tag ist fehlgeschlagen, da der Tag-Datentyp nicht unterstützt wird.

Mögliche Lösung:

1. Ändern Sie den Tag-Datentyp auf einen unterstützten Typ. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.
2. Wenden Sie sich an den technischen Support.

Block von Gerät kann nicht gelesen werden. Controller-Tag-Datentyp ist unbekannt. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl>, unbekannter Datentyp = <Typ>.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Anforderung für das angegebene Tag ist fehlgeschlagen, da der Tag-Datentyp nicht unterstützt wird.

Mögliche Lösung:

1. Ändern Sie den Tag-Datentyp auf einen unterstützten Typ. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.
2. Wenden Sie sich an den technischen Support.

In Tag auf Gerät kann nicht geschrieben werden. Datentyp wird nicht unterstützt. | Tag-Adresse = '<Adresse>', nicht unterstützter Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Anforderung für das angegebene Tag ist fehlgeschlagen, da der Tag-Datentyp nicht unterstützt wird.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp in einen unterstützten Typ. Beispiel: Der Datentyp 'Short' ist für ein natives Tag in einem BOOL-Array nicht zulässig. Ändern Sie den Datentyp in 'Boolean', um das Problem zu beheben.

Siehe auch:
Adressieren unteilbarer Datentypen

Tag kann nicht von Gerät gelesen werden. Datentyp wird nicht unterstützt. Tag deaktiviert. | Tag-Adresse = '<Adresse>', nicht unterstützter Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Anforderung für das angegebene Tag ist fehlgeschlagen, da der Tag-Datentyp nicht unterstützt wird.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp auf einen unterstützten Typ. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

Block von Gerät kann nicht gelesen werden. Datentyp wird nicht unterstützt. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl> (Elemente), nicht unterstützter Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Anforderung für das angegebene Tag ist fehlgeschlagen, da der Tag-Datentyp nicht unterstützt wird.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp auf einen unterstützten Typ. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

In Tag kann nicht geschrieben werden. Datentyp für Tag ist unzulässig. | Tag-Adresse = '<Adresse>', unzulässiger Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Die Schreibanforderung für das angegebene Tag ist fehlgeschlagen, da der Client-Tag-Datentyp für das angegebene native Tag unzulässig ist.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp in einen unterstützten Typ. Beispiel: Der Datentyp 'Short' ist für ein natives Tag in einem BOOL-Array nicht zulässig. Ändern Sie den Datentyp in 'Boolean', um dieses Problem zu beheben.

Siehe auch:
Adressieren unteilbarer Datentypen

Tag von Gerät kann nicht gelesen werden. Datentyp für dieses Tag ist unzulässig. Tag deaktiviert. | Tag-Adresse = '<Adresse>', unzulässiger Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Die Leseanforderung für das angegebene Tag ist fehlgeschlagen, da der Client-Tag-Datentyp für das angegebene native Tag unzulässig ist.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp in einen unterstützten Typ. Beispiel: Der Datentyp 'Short' ist für ein natives Tag in einem BOOL-Array nicht zulässig. Ändern Sie den Datentyp in 'Boolean', um dieses Problem zu beheben. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

Block von Gerät kann nicht gelesen werden. Datentyp für Block ist unzulässig. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl> (Elemente), unzulässiger Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Leseanforderung für das angegebene Tag ist fehlgeschlagen, da der Client-Tag-Datentyp für das angegebene native Tag unzulässig ist.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp in einen unterstützten Typ. Beispiel: Der Datentyp 'Short' ist für ein natives Tag in einem BOOL-Array nicht zulässig. Ändern Sie den Datentyp in 'Boolean', um dieses Problem zu beheben. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

In Tag auf Gerät kann nicht geschrieben werden. Tag unterstützt keine Arrays mit mehreren Elementen. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Leseanforderung für das angegebene Tag ist fehlgeschlagen, da der Treiber für den angegebenen nativen Tag keinen Zugriff durch Arrays mit mehreren Elementen unterstützt.

Mögliche Lösung:
Ändern Sie Datentyp oder Adresse des Tags in unterstützte Einstellungen. Als Reaktion auf diesen Fehler wird das Tag deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

Tag kann nicht von Gerät gelesen werden. Tag unterstützt keine Arrays mit mehreren Elementen. Tag deaktiviert. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Leseanforderung für das angegebene Tag ist fehlgeschlagen, da der Treiber für den angegebenen nativen Tag keinen Zugriff durch Arrays mit mehreren Elementen unterstützt.

Mögliche Lösung:
Ändern Sie Datentyp oder Adresse des Tags in unterstützte Einstellungen. Als Reaktion auf diesen Fehler wird das Tag deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

Block von Gerät kann nicht gelesen werden. Block unterstützt keine Arrays mit mehreren Elementen. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl> (Elemente).

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Leseanforderung für das angegebene Tag ist fehlgeschlagen, da der Treiber für das angegebene Tag keinen Zugriff durch Arrays mit mehreren Elementen unterstützt.

Mögliche Lösung:
Ändern Sie den Datentyp oder die Adresse für Tags in diesem Block in unterstützte Einstellungen. Als Reaktion auf diesen Fehler werden die Blockelemente deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

In Tag auf Gerät kann nicht geschrieben werden. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Mögliche Ursache:

1. Die Verbindung zwischen dem Gerät und dem Host-PC ist unterbrochen.
2. Die Kommunikationsparameter für die Verbindung sind falsch.
3. Möglicherweise wurde dem benannten Gerät eine falsche Adresse zugewiesen.

Mögliche Lösung:

1. Überprüfen Sie die Verkabelung zwischen dem PC und dem Gerät.
2. Vergewissern Sie sich, dass für das benannte Gerät der richtige Port angegeben wurde.
3. Vergewissern Sie sich, dass die Adresse des benannten Geräts mit der des eigentlichen Geräts übereinstimmt.

Tag kann nicht von Gerät gelesen werden. Tag deaktiviert. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Mögliche Ursache:

1. Die Verbindung zwischen dem Gerät und dem Host-PC ist unterbrochen.
2. Die Kommunikationsparameter für die Verbindung sind falsch.
3. Möglicherweise wurde dem benannten Gerät eine falsche Adresse zugewiesen.

Mögliche Lösung:

1. Überprüfen Sie die Verkabelung zwischen dem PC und dem Gerät.
2. Vergewissern Sie sich, dass für das benannte Gerät der richtige Port angegeben wurde.
3. Vergewissern Sie sich, dass die Adresse des benannten Geräts mit der des eigentlichen Geräts übereinstimmt.

Hinweis:
Als Reaktion auf diesen Fehler wird das Tag deaktiviert und nicht erneut verarbeitet.

Block von Gerät kann nicht gelesen werden. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl>.

Fehlertyp:
Warnung

Mögliche Ursache:

1. Die Verbindung zwischen dem Gerät und dem Host-PC ist unterbrochen.
2. Die Kommunikationsparameter für die Verbindung sind falsch.
3. Möglicherweise wurde dem benannten Gerät eine falsche Adresse zugewiesen.

Mögliche Lösung:

1. Überprüfen Sie die Verkabelung zwischen dem PC und dem Gerät.
2. Vergewissern Sie sich, dass für das benannte Gerät der richtige Port angegeben wurde.
3. Vergewissern Sie sich, dass die Adresse des benannten Geräts mit der des eigentlichen Geräts übereinstimmt.

Hinweis:
Als Reaktion auf diesen Fehler wird der Block deaktiviert und nicht erneut verarbeitet.

Gerät hat mit CIP-Fehler geantwortet. | Statuscode = <Code>, erweiterter Statuscode = <Code>.

Fehlertyp:
Warnung

Mögliche Ursache:
Während einer Anforderung wurde vom Gerät ein Fehler im CIP-Teil des Pakets zurückgegeben. Alle Lese- und Schreibvorgänge innerhalb der Anforderung sind fehlgeschlagen.

Mögliche Lösung:
Die Lösung hängt davon ab, welche Fehlercodes zurückgegeben wurden. Weitere Informationen finden Sie in den CIP-Codes.

Siehe auch:
CIP-Fehlercodes

Speicherplatz für Tag konnte nicht zugeordnet werden. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Die zum Erstellen eines Tags erforderlichen Ressourcen konnten nicht zugeordnet werden. Das Tag wird nicht zum Projekt hinzugefügt.

Mögliche Lösung:
Schließen Sie nicht verwendete Anwendungen, und/oder erhöhen Sie den virtuellen Arbeitsspeicher. Versuchen Sie es anschließend erneut.

Gerät hat mit Kapselungsfehler geantwortet.

Fehlertyp:
Warnung

Mögliche Ursache:
Während einer Anforderung wurde vom Gerät ein Fehler im Kapselungsteil des Pakets zurückgegeben. Alle Lese- und Schreibvorgänge innerhalb der Anforderung sind fehlgeschlagen.

Mögliche Lösung:

1. Wiederherstellung des Treibers nach Fehler wird versucht.
2. Die Lösung hängt davon ab, welche Fehlercodes vom Gerät zurückgegeben wurden.

Siehe auch:

1. Kapselungsprotokoll-Fehlercodes
2. Fehlermatrix

Tag kann nicht von Gerät gelesen werden. Interner Speicher ist ungültig. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Tag kann nicht von Gerät gelesen werden. Datentyp für Tag ist unzulässig. | Tag-Adresse = '<Adresse>', unzulässiger Datentyp = '<Typ>'.

Fehlertyp:
Warnung

Mögliche Ursache:
Eine Leseanforderung für das angegebene Tag ist fehlgeschlagen, da der Client-Tag-Datentyp für das angegebene native Tag unzulässig ist.

Mögliche Lösung:
Ändern Sie den Tag-Datentyp in einen unterstützten Typ. Beispiel: Der Datentyp 'Short' ist für ein natives Tag in einem BOOL-Array nicht zulässig. Ändern Sie den Datentyp in 'Boolean', um dieses Problem zu beheben. Als Reaktion auf diesen Fehler wird das Tag deaktiviert und nicht erneut verarbeitet.

Siehe auch:
Adressieren unteilbarer Datentypen

Tag kann nicht von Gerät gelesen werden. Interner Speicher ist ungültig. Tag deaktiviert. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Block von Gerät kann nicht gelesen werden. Interner Speicher ist ungültig. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl> (Elemente).

Fehlertyp:
Warnung

In Adresse auf Gerät kann nicht geschrieben werden. Interner Speicher ist ungültig. | Tag-Adresse = '<Adresse>'.

Fehlertyp:
Warnung

Block von Gerät kann nicht gelesen werden. Block deaktiviert. | Blockanfang = '<Adresse>', Blockgröße = <Anzahl>, CIP-Fehler = <Code>, erweiterter Fehler = <Code>.

Fehlertyp:
Warnung

Mögliche Ursache:
Während einer Leseanforderung für das angegebene Tag wurde vom Gerät ein Fehler im CIP-Teil des Pakets zurückgegeben.

Mögliche Lösung:
Die Lösung hängt davon ab, welche Fehlercodes zurückgegeben wurden.

Siehe auch:
CIP-Fehlercodes

Geräte-ID-Details. | IP = '<Adresse>', Händler-ID = <Händler>, Produkttyp = <Typ>, Produktcode = <Code>, Revision = '<Revision>', Produktname = '<Produkt>', Produkt-Seriennummer = <serial Anzahl>.

Fehlertyp:

Informationen

Gerät unterstützt keine fragmentierten Lese-/Schreibdienste. Es wird automatisch auf nicht fragmentierte Dienste ausgewichen.

Fehlertyp:
Informationen

Glossar

Auf native Tags basierte Adressierung

Begriff	Definition
Array-Element	Element innerhalb eines nativen Array-Tags. Für Client/Server-Zugriff muss das Element unteilbar sein. Beispiel: ARRAYTAG [0].
Array mit Offset	Client/Server-Array-Tag, dessen Adresse ein angegebenes natives Array-Element aufweist. Beispiel: ARRAYTAG [0] {5}.
Array ohne Offset	Client/Server-Array-Tag, dessen Adresse kein angegebenes natives Array-Element aufweist. Beispiel: ARRAYTAG {5}.
Unteilbarer Datentyp	Ein vordefinierter, nicht strukturierter nativer Datentyp. Beispiel: SINT, DINT.
Unteilbares Tag	Ein mit einem unteilbaren Datentyp definiertes natives Tag.
Client	Ein HMI-, SCADA- oder Daten-Bridging-Softwarepaket, das OPC, DDE oder ein proprietäres Client/Server-Protokoll für die Schnittstelle mit dem Server verwendet.
Datentyp für Client/Server	Datentyp für Tags, die statisch auf dem Server oder dynamisch im Client definiert wurden. Die im Client unterstützten Datentypen sind vom verwendeten Client abhängig.*
Client/Server-Tag	Tag, das statisch auf dem Server oder dynamisch in einem Client definiert wurde. Diese Tags weisen andere Einheiten als native Tags auf. Beim Referenzieren eines solchen nativen Tags wird ein nativer Tag-Name zu einer Client/Server-Tag-Adresse.
Client/Server-Array	Das vom Server und einigen Clients unterstützte Datenpräsentationsformat (Zeile x Spalte). Nicht alle Clients unterstützen Arrays.
CCW	Connected Components Workbench.
Nativer Datentyp	Ein in CCW definierter Datentyp für Micro800-Controller.
Natives Tag	Ein in CCW definiertes Tag für Micro800-Controller.
Nativer Array-Datentyp	Ein in CCW unterstütztes mehrdimensionales Array (1, 2 oder 3 Dimensionen möglich) für Micro800-Controller. Alle unteilbaren Datentypen unterstützen native Arrays. Nicht alle strukturierten Datentypen unterstützen native Arrays.
Array-Tag	Ein mit einem nativen Array-Datentyp definiertes natives Tag.
Vordefinierter Datentyp	Ein von CCW unterstützter und vordefinierter nativer Datentyp für Micro800-Controller.*
Benutzerdefiniert Datentyp	Ein von CCW unterstützter und vom Benutzer definierter nativer Datentyp für Micro800-Controller.*
Server	Der OPC-, DDE- sowie der proprietäre Server verwenden diesen Micro800-Ethernet-Treiber von Allen-Bradley.
Strukturierter Datentyp	Ein vordefinierter oder benutzerdefinierter Datentyp, der aus Mitgliedern besteht, deren Datentypen unteilbar oder strukturiert sind.
Struktur-Tag	Ein mit einem strukturierten Datentyp definiertes natives Tag.

*Die auf dem Server unterstützten Datentypen sind unter Datentypbeschreibung aufgelistet.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z