OMRON コントローラ NJ シリーズ用 プロバイダ

Version 1.2.8

ユーザーズ ガイド

June 12, 2024

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2015-1-21	初版.
1.1.0	2016-12-9	・変数情報遅延取得機能追加 ・配列への直接書き込み機能追加 ・拡張 Ethernet/IP ユニット対応
1.1.1	2016-12-26	・エラーコード一覧追加
1.1.2	2017-1-7	・大きな配列へのアクセス対応
1.1.2	2018-5-21	・AddController メソッドの OpenMode オプション追加
1.1.2	2018-10-29	・タイムアウト設定オプション追加
1.2.0	2018-11-13	・AddVariable メソッドの OpenMode オプション追加 ・AddController メソッドの Model オプション追加, NX シリーズ対応
1.2.1	2019-3-4	・構造体一括取得で 2 次元以上の配列が正しく取得できるように修正 ・パフォーマンス向上
1.2.2	2019-7-30	・障害対応 ・構造体への直接アクセス方法を追記
1.2.3	2020-7-22	・GetVariableNames の不具合を修正
1.2.4	2020-11-9	・AddController メソッドの文言修正, データサイズに関する記述を追記 ・AddVariable メソッドの文言修正 ・タグデータリンクを用いた他局アクセス方法の追記 ・エラーコード一覧を更新 ・構造体の大きさが上限を超えた場合に構造体サイズエラーを出力するように修正
	2021-4-27	・DelayInit の説明修正 ・OpenMode の選択肢の文言修正 ・LBound の説明修正
1.2.5	2021-6-18	・一部のエラーコードを修正
1.2.6	2021-12-16	・AddController メソッドの BufferClearMode オプション追加 ・誤記修正
1.2.7	2022-3-25	・要求パケットと応答パケットの整合性チェックを追加
1.2.8	2024-06-12	・LargeForwardOpen 時に使用するシーケンス番号をコントローラ接続時にランダムな値で生成して使いまわすように修正. ・LargeForwardOpen 時に使用するシーケンス番号が重複してエラーになった場合にシーケンス番号を再作成する処理を追加. プロバイダが発行するエラーメッセージを変更.

【動作確認機種】

機種		注意事項
NJ301-1100		
NX701-1720		
NX1P2		

目次

1. はじめに

OMRON コントローラ NJ シリーズ用プロバイダは，OMRON コントローラ NJ シリーズで公開された変数に対して CIP 通信を用いてアクセスを行う ORiN2 CAO プロバイダです.

OMRON NJ シリーズ用プロバイダを利用することで，CIP プロトコルに詳しくない利用ユーザー様も，OMRON NJ シリーズの変数に簡単にアクセスすることができます.

本ドキュメントでは，OMRON NJ シリーズ用プロバイダの概要と，実装されている CAO インタフェース（関数仕様）について説明しています.

2. プロバイダの概要

2.1. インストール

OMRON NJ シリーズ用プロバイダモジュールは，下記の DLL で構成されています．ORiN2 SDK のインストーラでインストールした場合は，インストール作業は不要です．手動でインストールする場合は，表 2-1 のように実行してください．

表 2-1 NJ シリーズ用プロバイダ

ファイル名	CaoProvOmronNJ.dll
ProgID	CaoProv.OMRON.NJ
レジストリ登録	regsvr32 CaoProvOmronNJ.dll
レジストリ登録の抹消	regsvr32 /u CaoProvOmronNJ.dll

2.2. 概要

OMRON NJ シリーズ用プロバイダは，CIP プロトコルを用いて変数へのアクセスを行います．

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

CaoWorkspace に，コントローラオブジェクトを追加します．OMRON NJ シリーズ用プロバイダでは，AddController メソッド実行時に渡されたパラメータを参照し，該当する OMRON コントローラ NJ シリーズと接続を行います．以下に，AddController メソッドの仕様を示します．

書式

AddController

```
(
        ″<コントローラ名>″,            // コントローラ名(任意)
        ″CaoProv.OMRON.NJ″,            // プロバイダ名(固定)
        ″<マシン名>″,                  // プロバイダ実行マシン名
        ″<オプション>″                 // オプション文字列
)
```

オプション

以下にオプション文字列に指定するオプションを示します. オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります.

オプション	必須	説明	値範囲	デフォルト値
Conn	○	接続先を指定します. 詳細は 2.3.1.1.Conn オプションを参照してください. 例)“Conn=TCP:{接続先の IP アドレス}”	--	--
Timeout	--	通信がタイムアウトになるまでの時間をミリ秒単位で指定します.	1 - 2147483647	60000
DelayInit	--	変数情報の遅延取得機能の設定です. 以下のいずれかを指定してください. 0 : AddController(接続)時に全ての変数情報を取得します. 1 : AddVariable(変数追加)時に必要な変数情報を取得します.	0 - 1	0
Network_Type_Number	--	CPU ユニットのネットワーク識別番号を指定します.	1 - 255	1
Unit_Address	--	CPU ユニットの号機アドレスを指定します.	0 - 255	0
OpenMode	--	データ送受信時の通信方法を設定します. 詳細は 2.3.1.2.OpenMode オプションを参照してください.	0 - 2	0
Model	--	接続する CPU のモデルを指定します. 詳細は 2.3.1.3.Model オプションを参照してください.	0 - 1	0
BufferClearMode	--	リクエストを送信する前にバッファをクリアするかどうかを指定します. 以下のいずれかを指定してください. False : バッファをクリアしません. True : バッファをクリアします.	False, True	False

2.3.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します. ここで角括弧("[]")内は省略可能なことを, 各パラメータの解説中の下線部はオプションを指定しなかった時のデフォルト値をそれぞれ示します.

TCP

"Conn=TCP:<接続先 IP>[:<接続先ポート>]"

<接続先 IP> : 接続先 IP アドレスを***.***.***.***の形式で指定します.(必須)

<接続先ポート> : 接続先ポート番号を指定します. 44818

例 1)IP アドレス 192.168.1.10, ポート番号 44818, タイムアウト 3000ms, 変数情報の遅延取得機能 0, CPU ユニットのネットワーク識別番号 1, CPU ユニットの号機アドレス 0 で指定する場合

"Conn=TCP:192.168.1.10,Timeout=3000"

例 2)IP アドレス 192.168.1.10, ポート番号 44816, タイムアウト 3000ms, 変数情報の遅延取得機能 1, CPU ユニットのネットワーク識別番号 2, CPU ユニットの号機アドレス 1 で指定する場合

"Conn=TCP:192.168.1.10:44816,Timeout=3000,DelayInit=1,Network_Type_Number=2,Unit_Address=1"

2.3.1.2. OpenMode オプション

このオプションでは, データ送受信時の通信方法を指定します. 扱うデータのサイズが 240 バイトより小さい場合は UnConnected Send モード, 240 バイトより大きい場合は Large Forward Open モードを選択することで高速に通信できます[1].

また, 接続先のユニットによって接続可能な通信方法が異なります. 詳細は表 2-2 を参照してください.

設定値	モード	詳細	構造体サイズ上限(byte)	配列分割サイズ(byte)
0	Unconnected Send	扱うデータのサイズが 240 バイト以下の場合に選択してください. 240 バイト以上の配列を要求した場合は, 240 バイト単位で通信が分割されます.	240	240
1	Large Forward Open (直接接続)	Ethernet ケーブルが NJ または NX に直接接続されており, 扱うデータのサイズが 240 バイトより大きい場合に選択してください. 1200 バイト以上の配列を要求した場合は, 1200 バイト単位で通信が分割されます.	1200	1200

[1] UnConnected Send モード:データ部＋ヘッダー部のサイズが CIP 規約の 502 バイトを超えた場合, 通信に失敗します.
Large Forward Open モード:データ部＋ヘッダー部のサイズが CIP 規約の 1994 バイトを超えた場合, 通信に失敗します.

設定値	モード	詳細	構造体サイズ上限(byte)	配列分割サイズ(byte)
2	Large Forward Open (転送)	ハブ経由で NJ または NX に接続し，扱うデータのサイズが 240 バイトより大きい場合に選択してください．1200 バイト以上の配列を要求した場合は，1200 バイト単位で通信が分割されます．	1200	1200

表 2-2 接続可能な接続対象ユニット

設定値	モード	内蔵 EtherNet ユニット	拡張 EtherNet ユニット
0	Unconnected Send	○	○
1	Large Forward Open (直接接続)	○	×
2	Large Forward Open	○	○

例 1)IP アドレス 192.168.1.10, ポート番号 44818, 扱うデータのサイズが 240 バイト以下の場合

"Conn=TCP:192.168.1.10,Timeout=3000"

例 2)IP アドレス 192.168.1.10, ポート番号 44818, 扱うデータのサイズが 240 バイトより大きくハブ経由で NJ または NX に接続する場合

"Conn=TCP:192.168.1.10,OpenMode=2"

2.3.1.3. Model オプション

このオプションでは，接続する CPU のモデルを指定します．このオプションは 2.3.1.2.OpenMode オプションで Large Forward Open モードを指定した時のみ意味を持ちます．[2]

設定値	モデル	詳細	構造体サイズ上限(byte)	配列分割サイズ(byte)
0	全モデル	全てのモデルで選択できます．	1200	1200
1	NX701 CPU	NX701 の場合に選択してください．一度に転送可能なデータサイズが増え，通信が効率化されます．	7396	7396

[2] NX701 CPU モード:データ部＋ヘッダー部のサイズが CIP 規約の 8192 バイトを超えた場合，通信に失敗します．

例）IP アドレス 192.168.1.10, ポート番号 44818, CPU モデルが NX701 にハブ経由で接続する場合

"Conn=TCP:192.168.1.10,OpenMode=2,Model=1"

2.3.2. CaoController::GetVariableNames メソッド

接続した NJ シリーズで定義されている変数の一覧を取得します. 本メソッドで取得した変数名は, 後述する AddVariable メソッドの第一引数に使用することができます. 以下に, GetVariableNames の仕様を示します.

書式

GetVariableNames

```
(
        ″<オプション>″              // オプション文字列(未使用)
)
```

2.3.3. CaoController::AddVariable メソッド

変数へアクセスするための CaoVariable オブジェクトを追加します. 以下に, AddVariable の仕様を示します.

書式

AddVariable

```
(
        ″<変数名>″,                // 変数名
        ″<オプション>″              // オプション文字列
)
```

変数名

変数名には 2.3.2.CaoController::GetVariableNames メソッドで取得した変数名を指定できます. オプション文字列の Path オプションにアクセスパスを指定した場合は任意の変数名を指定可能です.

オプション

以下にオプション文字列に指定するオプションを示します. オプション文字列は下記に示す各オプションをカンマ(,)でつなげた文字列となります.

オプション	必須	説明	値範囲	デフォルト値
Path	--	アクセスパスを指定します（アクセスパスの指定方法参照）. Path オプションを指定した場合, 変数名は任意の文字列を指定可能です. 存在しないアクセスパスを指定した場合, 共用体を直接指定した場合はエ	--	--

オプション	必須	説明	値範囲	デフォルト値
		ラーになります. 省略時は<変数名>で指定した名前が指定されたことになります.		
LBound	−−	配列の先頭の添え字を指定します. また, Elem に-1 を指定した場合はこのオプションは無視されます.	0 - 65535	0
Elem	−−	配列長を指定します. 詳細は 2.3.3.1.Elem オプションを指定してください.	-1 - 2147483647	-1
OpenMode	−−	Variable ごとのデータ送受信時の通信方法を設定します. 値の詳細は, 2.3.1.2.OpenMode オプションと同様です. Variable で OpenMode を指定するとコントローラオブジェクト追加時に指定した OpenMode オプションを上書きした形で動作を行います.	0 - 2	AddController 時の OpenMode の値がデフォルトに設定されます.

2.3.3.1. Elem オプション

Elem オプションはアクセスする配列長を指定します. 1 を指定した場合は配列ではなく単体の値として扱われます. -1 を指定した場合は自動的に LBound オプションおよび Elem オプションの値を決定します. また, LBound オプションと Elem オプションを組み合わせることで配列の一部にアクセスできるようになります.

例)配列名 Arr[0..255]の Arr[100]～Arr[107]にアクセスしたい場合は以下のように指定します.

LBound=100,Elem=8

3. コマンドリファレンス

3.1. Controller クラス

表 3-1 CaoController::Execute コマンド一覧

3.1.1. CaoController::Execute(“GetVariableType”) コマンド

変数の型をプロバイダ内部で扱う型として取得します.

書式　GetVariableType (<VariableName>)

< VariableName > : [in]変数名 (VT_BSTR)
戻り値 : [out]変数型 (VT_UI2). 変数型参照. 存在しない変数を指定した場合はエラーになる.

使用例

```
Object obj = ctrl.Execute("GetVariableType", "Var_A");
```

3.1.2. CaoController::Execute(“GetStructName”) コマンド

構造体の名前を取得します.

書式　GetStructName (<VariableName>)

< VariableName > : [in]変数名 (VT_BSTR)
戻り値 : [out]構造体名 (VT_BSTR). 存在しない変数を指定した場合および変数型が構造体もしくは構造体の配列以外の場合はエラーになる.

使用例

```
string structName = (string)ctrl.Execute("GetStructName", "Var_A");
```

3.1.3. CaoController::Execute("GetStructMemberNames") コマンド

構造体/共用体メンバーの名前の一覧を取得します.

書式　GetStructMemberNames (<StructName>)

< StructName > : [in] 構造体/共用体名 (VT_BSTR)
戻り値 : [out] 構造体/共用体メンバー名の一覧 (VT_BSTR | VT_ARRAY). 存在しない構造体/共用体を指定した場合はエラーになる.

使用例

```
string[] varNames = (string[])ctrl.Execute("GetStructMemberNames", "STRUCT_A");
```

3.1.4. CaoController::Execute("GetStructMemberType") コマンド

構造体/共用体メンバーの変数型をプロバイダ内部で扱う型として取得します.

書式　GetStructMemberType (<MemberName>)

< MemberName > : [in] 構造体/共用体名および構造体/共用体メンバー名. (VT_BSTR | VT_ARRAY または VT_VARIANT | VT_ARRAY) 第一要素に構造体/共用体名, 第二要素に構造体/共用体メンバー名を指定する.
戻り値 : [out] 構造体/共用体メンバーの変数型 (VT_UI2). 変数型参照. 存在しない構造体/共用体または構造体内に存在しない構造体/共用体メンバー名を指定した場合はエラーになる.

使用例

```
Object obj = ctrl.Execute("GetStructMemberType",
 new string[]{"STRUCT_A", "MEMBER_A"})
```

3.1.5. CaoController::Execute(“GetStructMemberStructName”) コマンド

構造体/共用体メンバーの構造体名を取得します.

書式　GetStructMemberStructName (<MemberName>)

< MemberName >	:	[in] 構造体/共用体名および構造体/共用体メンバー名. (VT_BSTR | VT_ARRAY または VT_VARIANT | VT_ARRAY) 第一要素に構造体/共用体名, 第二要素に構造体/共用体メンバー名を指定する.
戻り値	:	[out] 構造体名 (VT_BSTR). 存在しない構造体/共用体または構造体内に存在しない構造体/共用体メンバー名を指定した場合はエラーになる.

使用例

```
string structName = (string)ctrl.Execute("GetStructMemberStructName",
new string[]{“STRUCT_A”, “MEMBER_A”})
```

3.1.6. CaoController::Execute(“GetArrayLength”) コマンド

変数の配列の要素数を取得します.

書式　GetArrayLength (<VariableName>)

< VariableName >	:	[in]変数名 (VT_BSTR)
戻り値	:	[out]配列の要素数(VT_UI4 | VT_ARRAY). 戻り値の要素数が配列の次元となる. 存在しない変数を指定した場合および変数が配列以外の場合はエラーになる.

使用例

```
uint[] arrLen = (uint[])ctrl.Execute("GetArrayLength", “Var_A”);
```

3.1.7. CaoController::Execute(“GetStructMemberArrayLength”) コマンド

構造体/共用体メンバーの要素数を取得します.

書式　GetStructMemberArrayLength (<MemberName>)

< MemberName > : [in] 構造体/共用体名および構造体/共用体メンバー名. (VT_BSTR | VT_ARRAY または VT_VARIANT | VT_ARRAY) 第一要素に構造体/共用体名, 第二要素に構造体/共用体メンバー名を指定する.

戻り値 : [out] 配列の要素数(VT_UI4 | VT_ARRAY). 戻り値の要素数が配列の次元となる. 存在しない構造体/共用体を指定した場合, 構造体内に存在しない構造体/共用体メンバー名を指定した場合, 構造体/共用体メンバーが配列以外の場合はエラーになる.

使用例

```
uint[] arrLen = (uint[])ctrl.Execute("GetStructMemberArrayLength",
new string[]{“STRUCT_A”, “MEMBER_A”})
```

3.1.8. CaoController::Execute(“GetArrayLBound”) コマンド

変数の配列の要素数を取得します.

書式　GetArrayLBound (<VariableName>)

< VariableName > : [in]変数名 (VT_BSTR)

戻り値 : [out]配列の添え字の下限(VT_UI4 | VT_ARRAY). 戻り値の要素数が配列の次元となる. 存在しない変数を指定した場合および変数が配列以外の場合はエラーになる.

使用例

```
uint[] arrLen = (uint[])ctrl.Execute("GetArrayLBound ", “Var_A”);
```

3.1.9. CaoController::Execute(“GetStructMemberArrayLBound”) コマンド

構造体/共用体メンバーの配列の添え字の下限を取得します.

書式　GetStructMemberArrayLBound (<MemberName>)

< MemberName > : [in] 構造体/共用体名および構造体/共用体メンバー名. (VT_BSTR | VT_ARRAY または VT_VARIANT | VT_ARRAY) 第一要素に構造体/共用体名, 第二要素に構造体/共用体メンバー名を指定する.

戻り値 : [out] 配列の添え字の下限(VT_UI4 | VT_ARRAY). 戻り値の要素数が配列の次元となる. 存在しない構造体/共用体を指定した場合, 構造体内に存在しない構造体/共用体メンバー名を指定した場合, 構造体/共用体メンバーが配列以外の場合はエラーになる.

使用例

```
uint[] arrLen = (uint[])ctrl.Execute("GetStructMemberArrayLBound",
new string[]{"STRUCT_A", "MEMBER_A"})
```

3.1.10. CaoController::Execute(“GetValue”) コマンド

CaoVariable を使用せずに直接変数の値を取得します.

書式　GetValue (<AccessPath>)

< AccessPath > : [in]アクセスパス (VT_BSTR). アクセスパスの指定方法参照.

戻り値 : [out]変数の値 (VT_VARIANT). 存在しない変数を指定した場合, 構造体/共用体のメンバーを直接指定した場合, 2 次元以上の配列を直接指定した場合はエラーになる.

使用例

```
Object obj = _ctrl.Execute("GetValue", "Var_A.MEMBER_A[0]")
```

3.1.11. CaoController::Execute(“PutValue”) コマンド

CaoVariable を使用せずに直接変数の値を設定します.

書式　PutValue (<Data>)

< Data>	:	[in] アクセスパスおよび設定する値. (VT_VARIANT | VT_ARRAY) 第一要素にアクセスパス(アクセスパスの指定方法参照), 第二要素に書き込む値を指定する.
戻り値	:	[out] なし. 存在しない変数を指定した場合, 構造体/共用体のメンバーを直接指定した場合, 指定した値が変数型に変換できない場合はエラーになる.

使用例

```
_ctrl.Execute("PutValue",
new object[]{ “Var_A.MEMBER_A[0]”, “Value” });
```

3.1.12. CaoController::Execute(“GetVariableCIPType”) コマンド

変数の変数型を NJ シリーズ内部の変数型として取得します.

書式　GetVariableCIPType (<VariableName>)

< VariableName >	:	[in]変数名 (VT_BSTR)
戻り値	:	[out]NJ シリーズ内部の変数型 (VT_UI1). 変数型参照. 配列の場合は配列の要素の変数型になる. 存在しない変数を指定した場合はエラーになる.

使用例

```
Object obj = ctrl.Execute("GetVariableCIPType", “Var_A”);
```

3.1.13. CaoController::Execute(“GetStructMemberCIPType”) コマンド

構造体/共用体メンバーの変数型を NJ シリーズ内部の変数型として取得します.

書式　GetStructMemberType (<MemberName>)

< MemberName > : [in] 構造体/共用体名および構造体/共用体メンバー名. (VT_BSTR | VT_ARRAY または VT_VARIANT | VT_ARRAY)
第一要素に構造体/共用体名, 第二要素に構造体/共用体メンバー名を指定する.

戻り値 : [out] 構造体/共用体メンバーの NJ シリーズ内部の変数型 (VT_UI1). 変数型参照.
配列の場合は配列の要素の変数型になる. 存在しない変数を指定した場合はエラーになる.

使用例

```
Object obj = ctrl.Execute("GetStructMemberCIPType",
 new string[]{“STRUCT_A”, “MEMBER_A”})
```

4. エラーコード一覧

OMRON コントローラ NJ シリーズ用プロバイダでは以下の固有エラーコードが定義されています.

表 4-1 固有エラーコード

エラー名	エラー番号	説明
型変換失敗	0x80101001	CAO のデータ型から NJ のデータ型への変換に失敗しました.
サポート外の型	0x80101002	NJ の変数の型がプロバイダがサポートしている型以外です.
不正ソケット長	0x80101003	ソケットの長さが不正です.
変数読み込み失敗	0x80101004	NJ の変数の読み込みに失敗しました.
セッション登録失敗	0x80101006	CIP セッションの登録に失敗しました.
変数書き込み失敗	0x80101009	NJ の変数の書き込みに失敗しました.
変数指定エラー	0x8010100b	NJ に指定した変数が存在しません.
構造体指定エラー	0x8010100c	構造体操作用のコマンドを実行する際に指定した変数が構造体ではありません.
コマンド実行エラー	0x8010100d	CIP コマンドの実行に失敗しました. AddController 時のオプション文字列で Model に 1 を指定し, NX701CPU 以外に接続しようとした場合もこのエラーが発生します. .Model に 0 を指定し再接続してください.
配列指定エラー	0x8010100f	配列操作用のコマンドを実行する際に指定した変数が配列ではありません.
通信切断エラー	0x80101010	CIP コマンドを実行しようとしましたが, 通信が切断されています.
構造体名取得エラー	0x80101011	構造体名を取得しようとしましたが, 指定した変数は構造体ではありません.
共用体直接アクセスエラー	0x80101012	共用体に直接アクセスしようとしました.
配列次元数指定エラー	0x80101013	指定された配列の次元数が不正です.
配列インデック範囲外エラー(下方)	0x80101014	指定されたインデックスが最小インデックス未満です.
配列インデック範囲外エラー(上方)	0x80101015	指定されたインデックスが最大インデックスを超えています.

構造体サイズエラー	0x80101016	構造体の大きさが上限を超えています.
内部エラー	0x80101fff	想定外のエラーが発生しました. サポートに連絡してください.

5. 構造体に直接アクセスした場合に取得できるデータについて

5.1. 構造体への直接アクセスの方法

構造体に直接アクセスするには構造体または構造体の配列のタグ名を指定してください.

5.2. 構造体に直接アクセスした場合のデータ型

構造体に直接アクセスした場合, 単体へのアクセスであれば VT_UI1 | VT_ARRAY, 配列であれば VT_VARIANT | VT_ARRAY の各要素に構造体 1 個分の VT_UI1 | VT_ARRAY が格納されたデータが取得できます.

5.3. 構造体のメンバーのオフセットの計算方法

NJ シリーズの構造体は CJ タイプの構造体と NJ タイプの構造体が指定できます.
CJ タイプの構造体はバイトオフセットが Sysmac Studio 上で確認できますが, NJ タイプの構造体の場合はオフセット値を自分で計算する必要があります.

NJ タイプの構造体のオフセットの計算のルールは以下の通りとなります.

- メンバーのオフセットは型の大きさの倍数でなければならない
    - ただし, メンバーが構造体である場合は, 構造体に含まれる最も大きな型の倍数になる
- 構造体のサイズは構造体内部の最も大きなデータの大きさの倍数でなければならない

小さいサイズのメンバーの後に大きなサイズのメンバーを配置すると計算が煩雑になりますので, NJ タイプの構造体を使用する場合はメンバーを大きい順に配置することをお勧めします.

例)
以下のような構造体があった場合, オフセットは以下のようになります.

メンバー名	大きさ	計算上のオフセット	アラインメントを考慮した実際のオフセット
boolData	2	0	0
byteData	1	2	2
wordData	2	3	4(3 以上で最も小さい 2 の倍数)
dwordData	4	6	8(6 以上で最も小さい 4 の倍数)
lwordData	8	12	16(12 以上で最も小さい 8 の倍数)

6. アクセスパスの指定方法

6.1. 変数名の指定

構造体以外の型を持つ変数は CaoController::GetVariableNames で取得した値でアクセスパスを指定可能です.

変数名で指定したアクセスパスで作成した CaoVariable::Value は読み取りおよび書き込みが可能です.

ただし, 配列への書き込みを行う場合は要素数が変数の要素数と同じ配列を渡す必要があります.

6.2. 構造体メンバー名の指定

構造体の型を持つ変数は 変数名 + "." + 構造体メンバー名 でアクセスパスを指定可能です.

構造体のメンバーが構造体の場合は, 上記のアクセスパスに加えて "." + 入れ子になっている構造体のメンバー名を指定してください.

6.3. 配列の要素の指定

配列の要素は大かっこ([])を用いて配列の添え字を指定します.

指定できる添え字の下限は GetArrayLBound または GetStructMemberArrayLBound で取得できます.

指定できる添え字の上限は添え字の下限 + (GetArrayLength もしくは GetStructMemberArrayLength で取得した要素数) – 1 です.

2 次元以上の配列を指定する場合は次元数の分だけ添え字を指定するか, 添え字を"."(ピリオド)区切りで指定してください.

例)

以下のような構造を持つ変数 Var の 2 番目の要素の MEMBER_D の(1,2)の要素にアクセスする場合は "Var[3].MEMBER_B.MEMBER_D[1][2]" または" Var[3].MEMBER_B.MEMBER_D[1.2]"と指定する.

```
Struct STRUCT_A {
        INT                     MEMBER_A;
        STRUCT_B                MEMBER_B
};
Struct STRUCT_B {
        INT                     MEMBER_C;
        ARRAY[0..7,0..7] OF INT MEMBER_D;
};
ARRAY[2..9] OF STRUCT_A         Var;
```

7. 変数型

7.1. GetVariableType/GetStructMemberVariableType で取得できる変数型

GetVariableType/GetStructMemberVariableType で取得できる変数の型は C++の VARTYPE の値で表現されます.
また, 変数が配列の場合, 変数型に 8192(VT_ARRAY)が加算された値になります.

表 7-1 変数型一覧

VARTYPE	値(10 進数)	意味
VT_I1	16	符号付き 1 バイト整数.
VT_I2	2	符号付き 2 バイト整数.
VT_I4	3	符号付き 4 バイト整数.
VT_I8	20	符号付き 8 バイト整数.
VT_R4	4	4 バイト実数.
VT_R8	5	8 バイト実数
VT_BSTR	8	文字列.
VT_UI1	17	符号なし 1 バイト整数.
VT_UI2	18	符号なし 2 バイト整数.
VT_UI4	19	符号なし 4 バイト整数.
VT_UI8	21	符号なし 8 バイト整数.
VT_USERDEFINED	29	構造体/共用体

7.2. GetVariableCIPType/GetStructMemberVariableCIPType で取得できる変数型

GetVariableCIPType/GetStructMemberVariableCIPType で取得できる変数の型は以下のように表現されます.
また, 変数が配列の場合, 取得できる値は配列の要素の変数型になります.

表 7-2 変数型一覧

NJ シリーズ内部の型	値(16 進数)	意味
BOOL	0xc1	論理値.
SINT	0xc2	符号付き 1 バイト整数.
INT	0xc3	符号付き 2 バイト整数.
DINT	0xc4	符号付き 4 バイト整数.
LINT	0xc5	符号付き 8 バイト整数.
USINT	0xc6	符号なし 1 バイト整数.
UINT	0xc7	符号なし 2 バイト整数.
UDINT	0xc8	符号なし 4 バイト整数.
ULINT	0xc9	符号なし 8 バイト整数.
REAL	0xca	4 バイト実数.
LREAL	0xcb	8 バイト実数
STRING	0xd0	文字列.
BYTE	0xd1	符号なし 1 バイト整数.
WORD	0xd2	符号なし 2 バイト整数.
DWORD	0xd3	符号なし 4 バイト整数.
LWORD	0xd4	符号なし 8 バイト整数.
STRUCT	0xa2	構造体.
UNION	0x0c	共用体.
ENUM	0x07	列挙体.
DATE_NSEC	0x08	符号なし 8 バイト整数で 1970-01-01 からの経過時間をナノ秒単位で表す.
TIME_NSEC	0x09	符号なし 8 バイト整数でナノ秒単位の経過時間を表す.
DATE_AND_TIME_NSEC	0x0a	符号なし 8 バイト整数で 1970-01-01 00:00:00 からの経過時間をナノ秒単位で表す.
TIME_OF_DAY_NSEC	0x0b	符号なし 8 バイト整数で 00:00:00 からの経過時間をナノ秒単位で表す.

8. サンプルプログラム

OMRON NJ シリーズ用プロバイダと Visual C# を使った簡単なサンプルプログラムを紹介します. サンプルプログラムの全容は下記のフォルダにありますので参考にしてください.

" <ORiN2 SDK インストールフォルダ>¥CAO¥ProviderLib¥OMRON¥NJ¥SAMPLE¥NJSample "

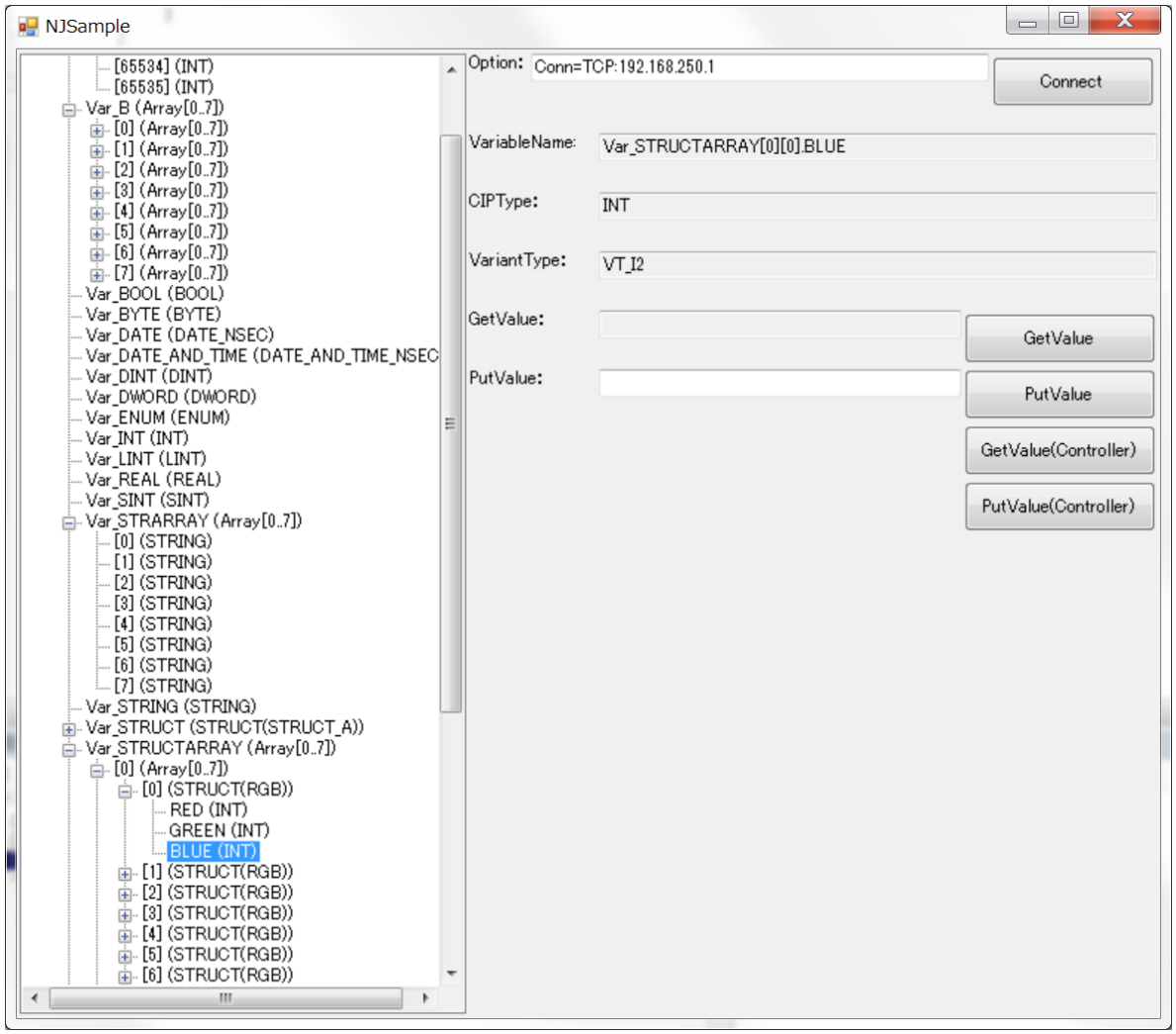


図 8-1 NJSample 画面

9. タグデータリンクを用いた他局アクセス方法

本章では，OMRON の NJ シリーズのタグデータリンク機能を用い，OMRON NJ シリーズ用プロバイダにて親 PLC から子 PLC の値を取得するための設定手順を記します． **他局アクセスの場合，データの読み込みは可能ですが，書き込みはできませんのでご注意ください．**

9.1. 事前準備

本章で紹介する方法では，親 PLC と子 PLC 両方に CPU ユニットのほかに拡張 EtherNet/IP ユニットを装着する必要があります．NJ シリーズの設定には OMRON 社製のアプリケーション「Sysmac Studio」を使用します．

親 PLC となる NJ シリーズに拡張 EtherNet/IP ユニットを装着し，子 PLC となる NJ シリーズにも拡張 EtherNet/IP ユニットを装着させます．PC と親 PLC の CPU ユニットにある内蔵 EtherNet/IP ポートを EtherNet ケーブルで接続し，親 PLC の拡張 EtherNet/IP ユニットと子 PLC の拡張 EtherNet/IP ユニットを EtherNet ケーブルで接続しておきます[3]．

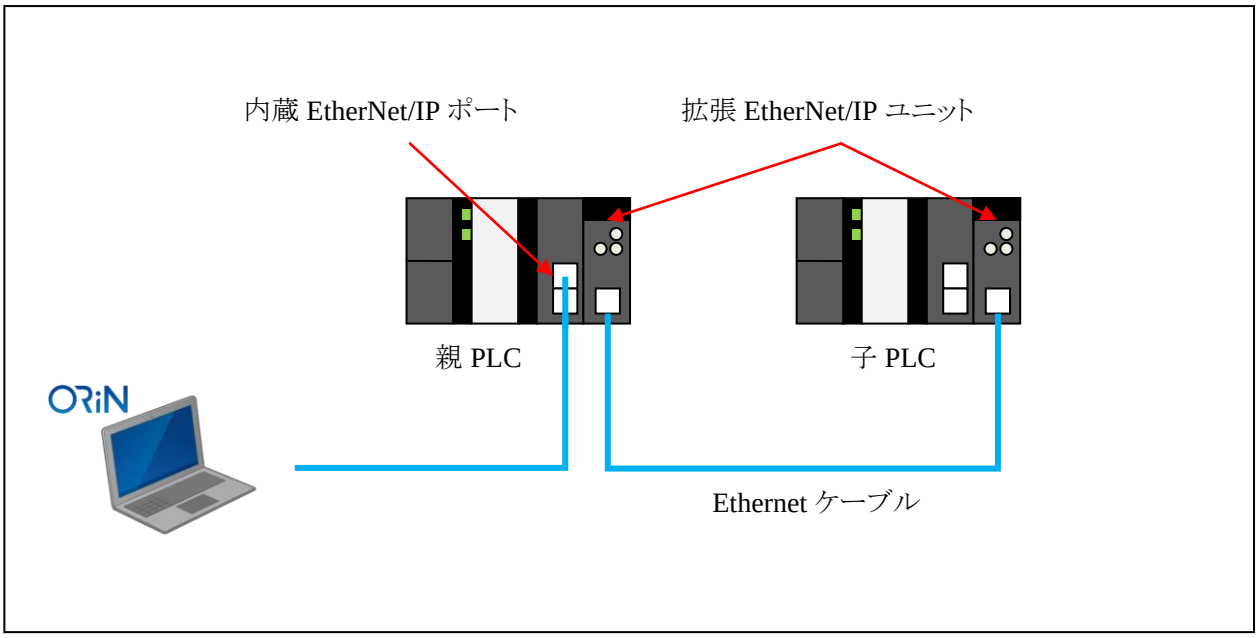


図 9-1 他局アクセス方法の概略図

[3] CJ1W-EIP21 ユニットを使用する場合，ユニットバージョンを Ver2.1 以上にする必要があります．

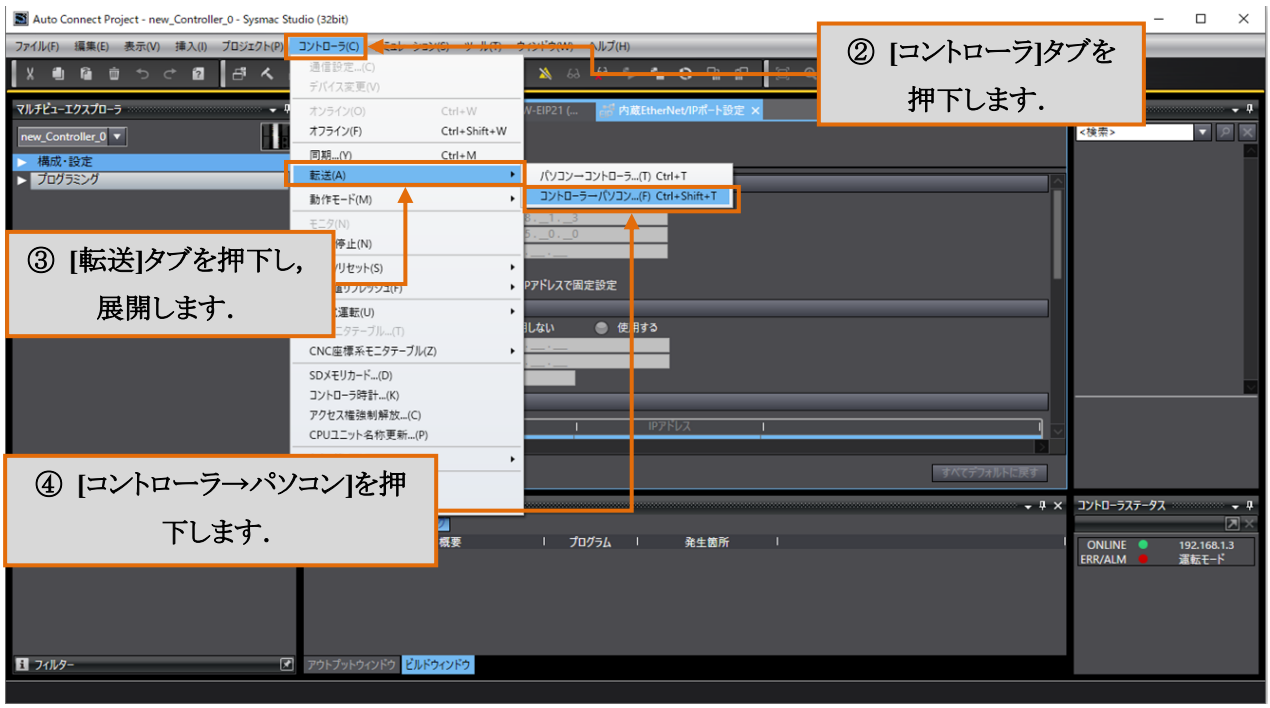
9.2. 親 PLC の設定方法

本節では，Sysmac Studio を用いて親 PLC 側の設定を行います．

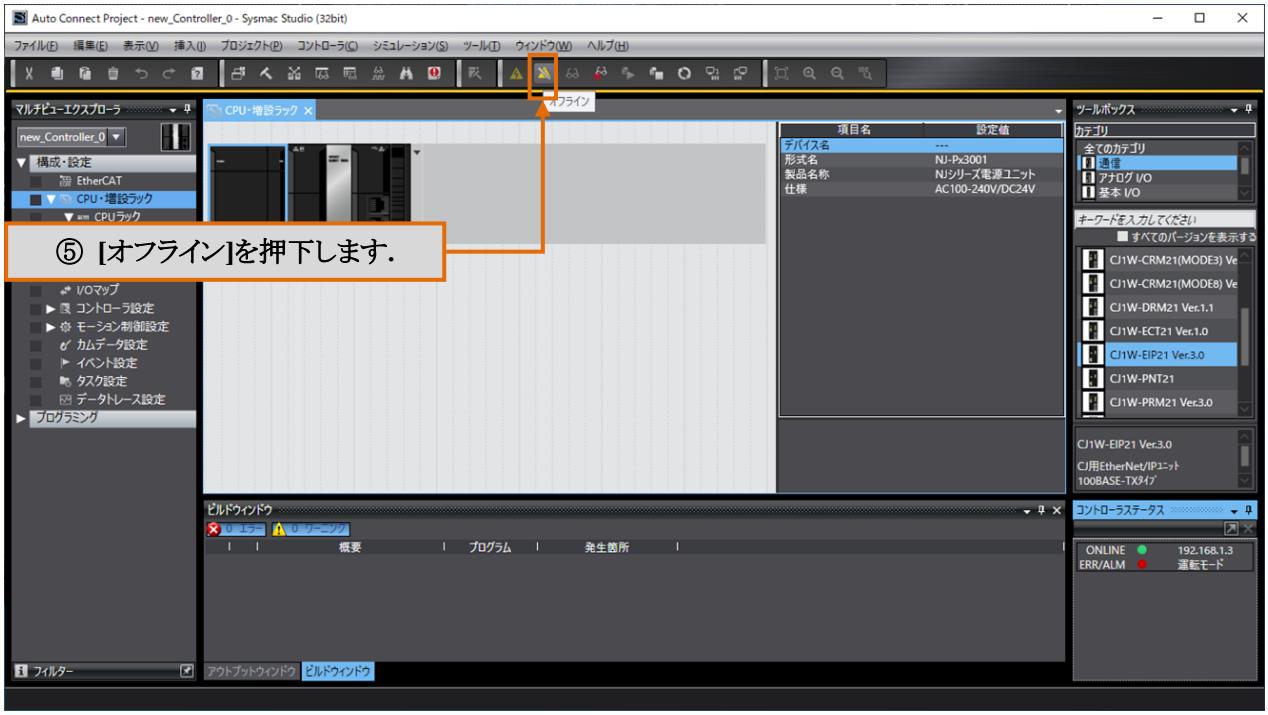
9.2.1. 構成設定

Sysmac Studio を起動し，デバイスに接続を行います．

PLC のデータに設定されている内容を取得します．

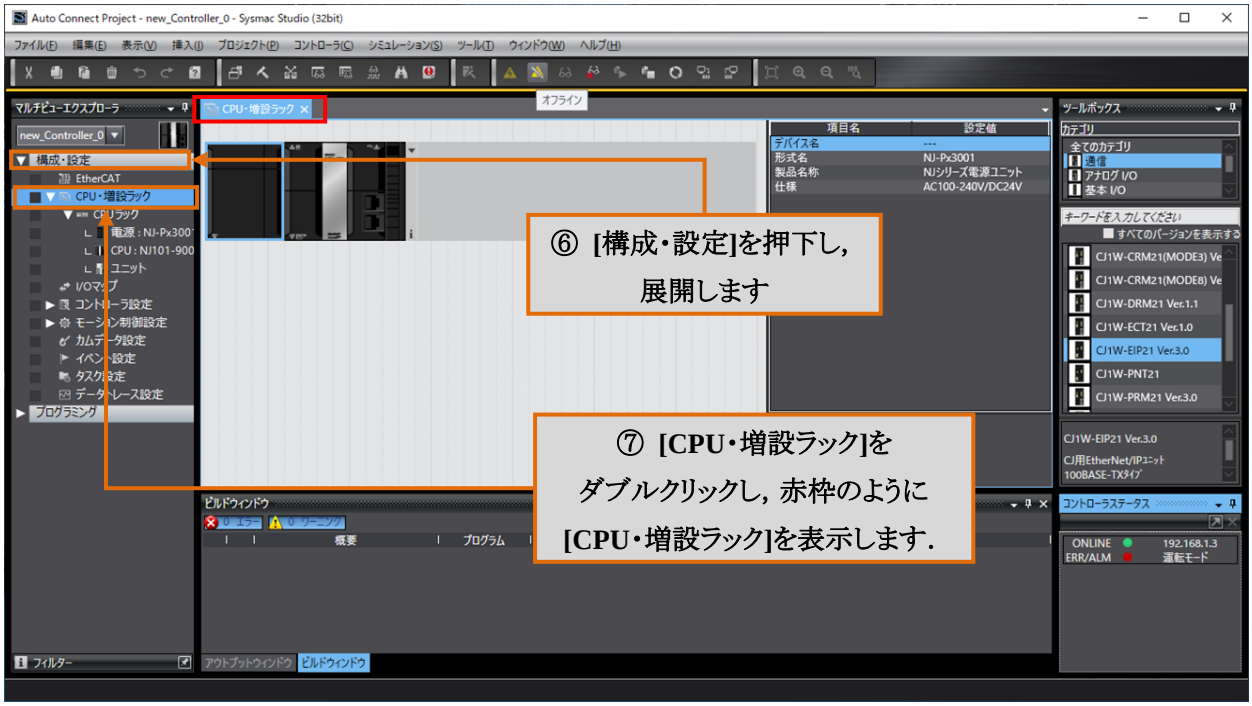


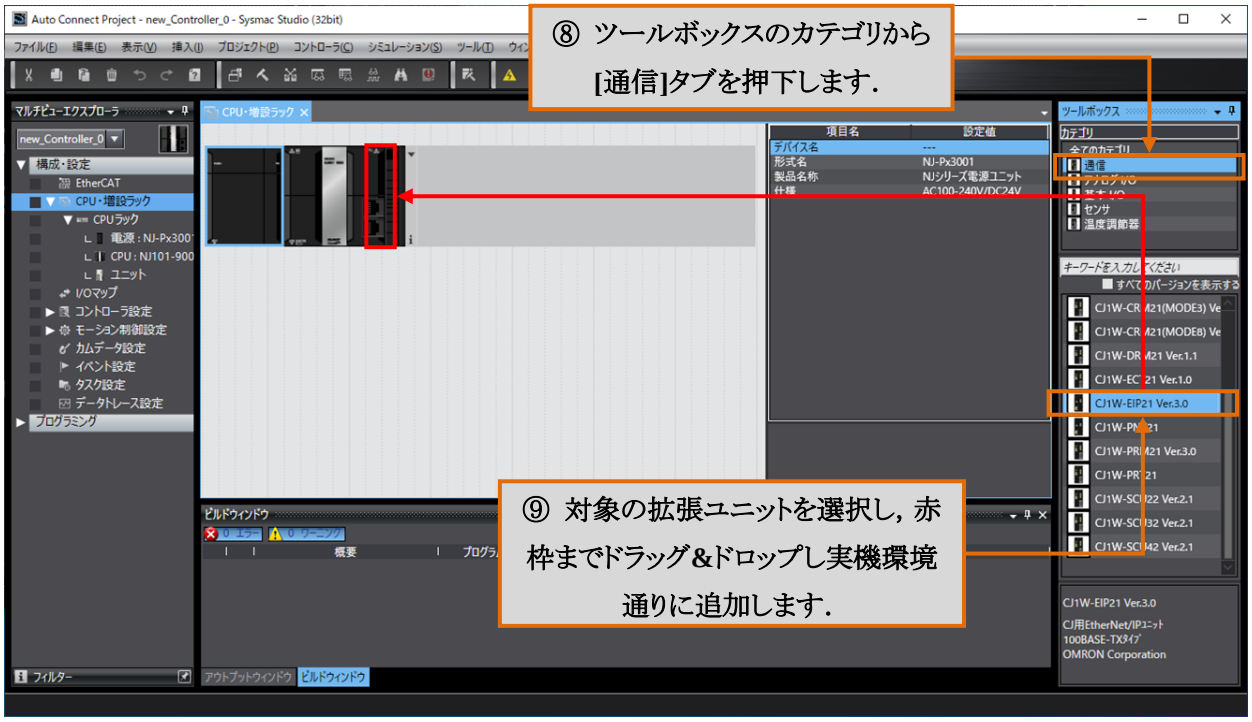
転送が完了したら設定の編集を行うため，一度オフラインにします．

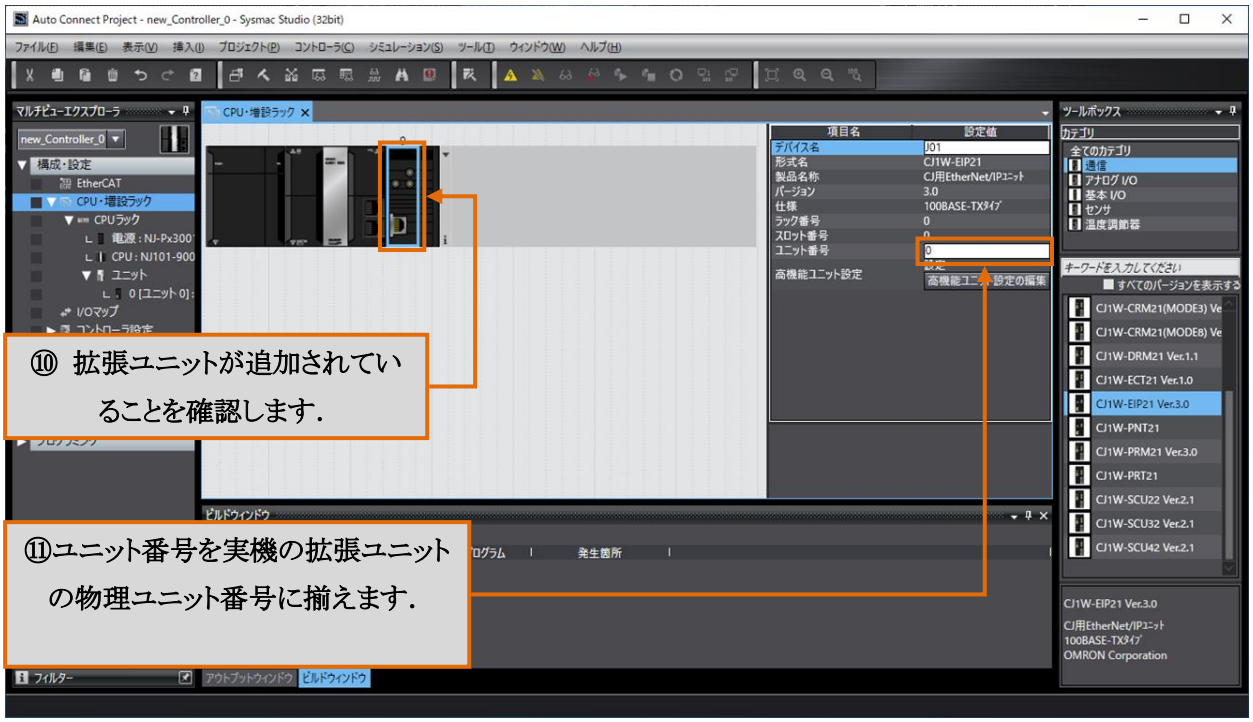


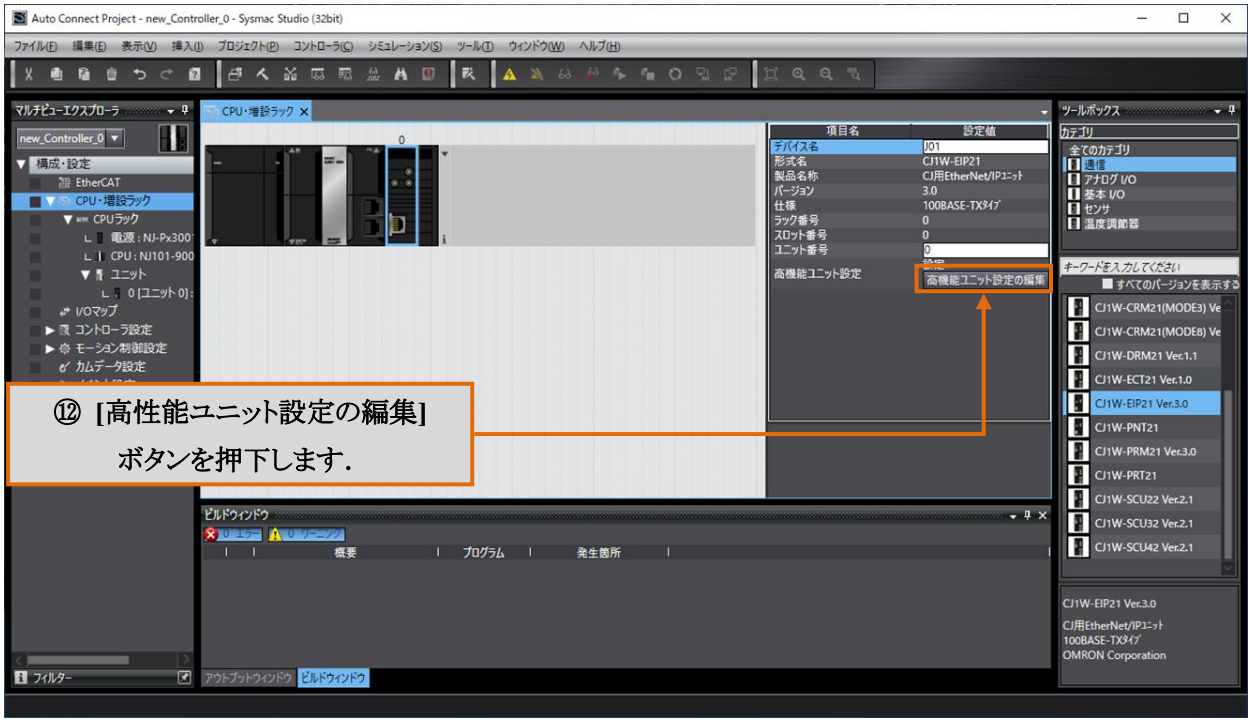
9.2.1.1. 拡張 EtherNet/IP ユニットの設定方法

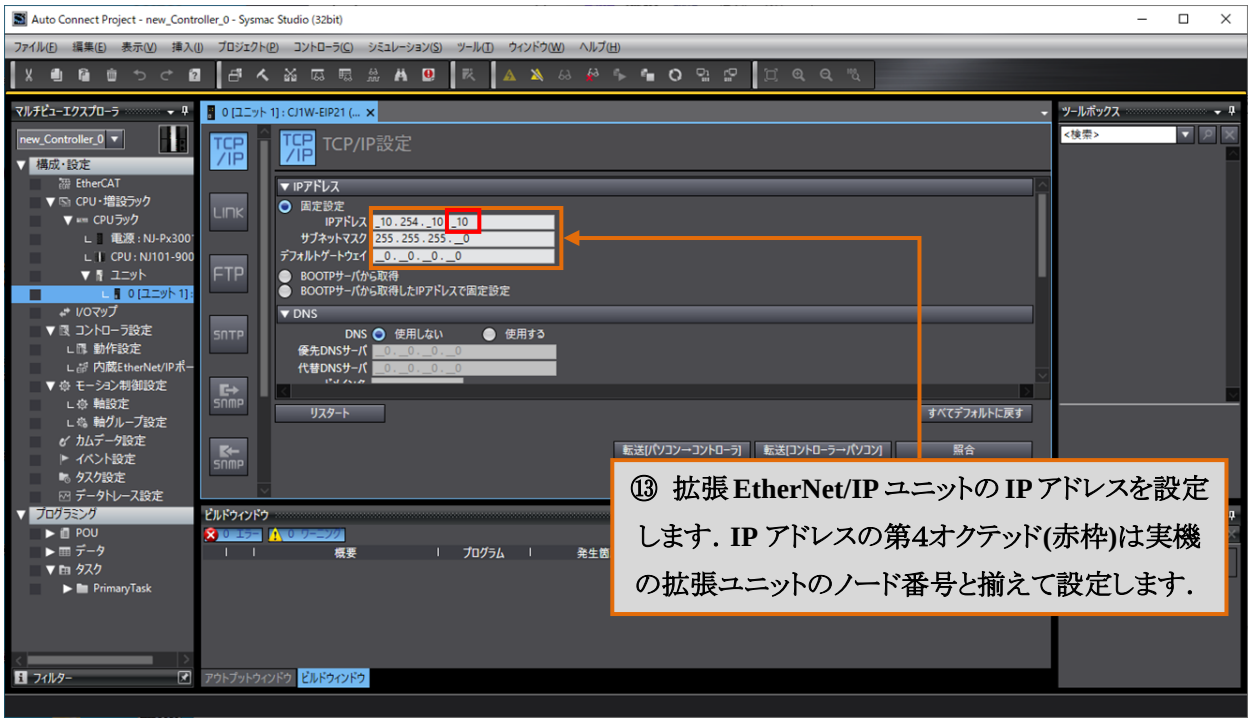
拡張 EtherNet/IP ユニットが認識されていない場合，構成・設定タブから設定を行います．認識されている場合は，手順の番号⑬から設定を行っていください．




⑧ ツールボックスのカテゴリから[通信]タブを押下します.
⑨ 対象の拡張ユニットを選択し，赤枠までドラッグ&ドロップし実機環境通りに追加します.

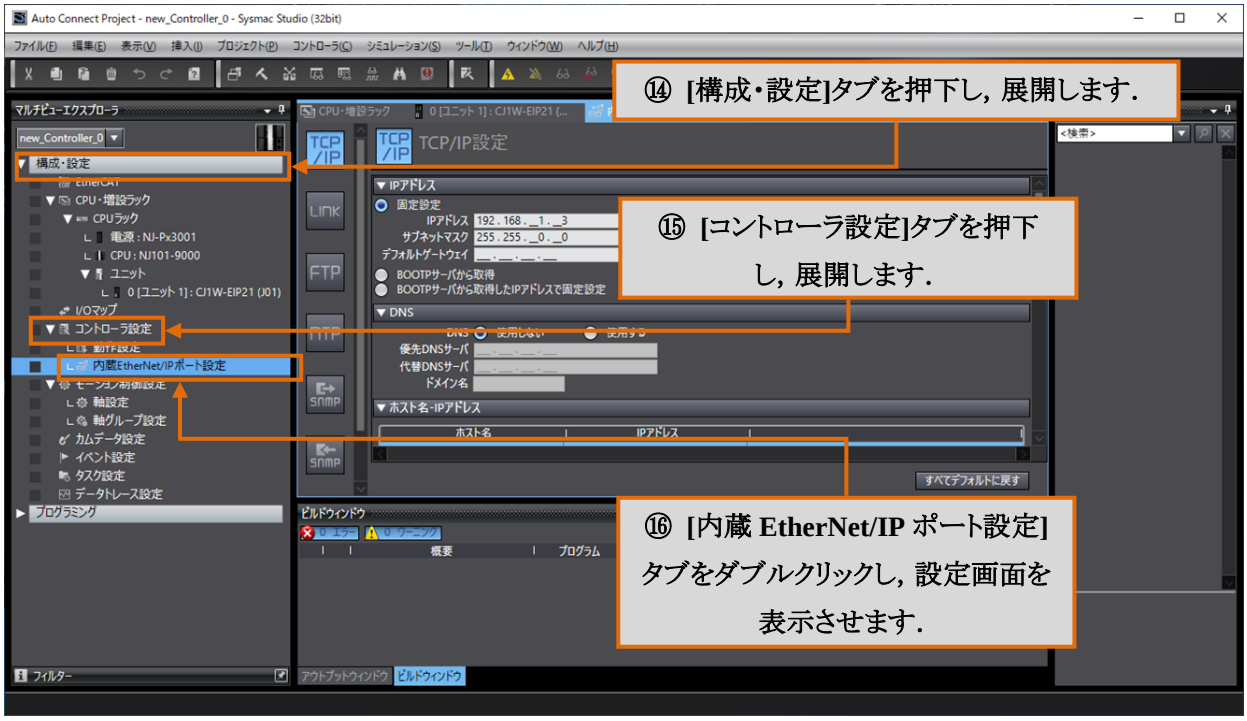

⑩ 拡張ユニットが追加されていることを確認します.
⑪ユニット番号を実機の拡張ユニットの物理ユニット番号に揃えます.

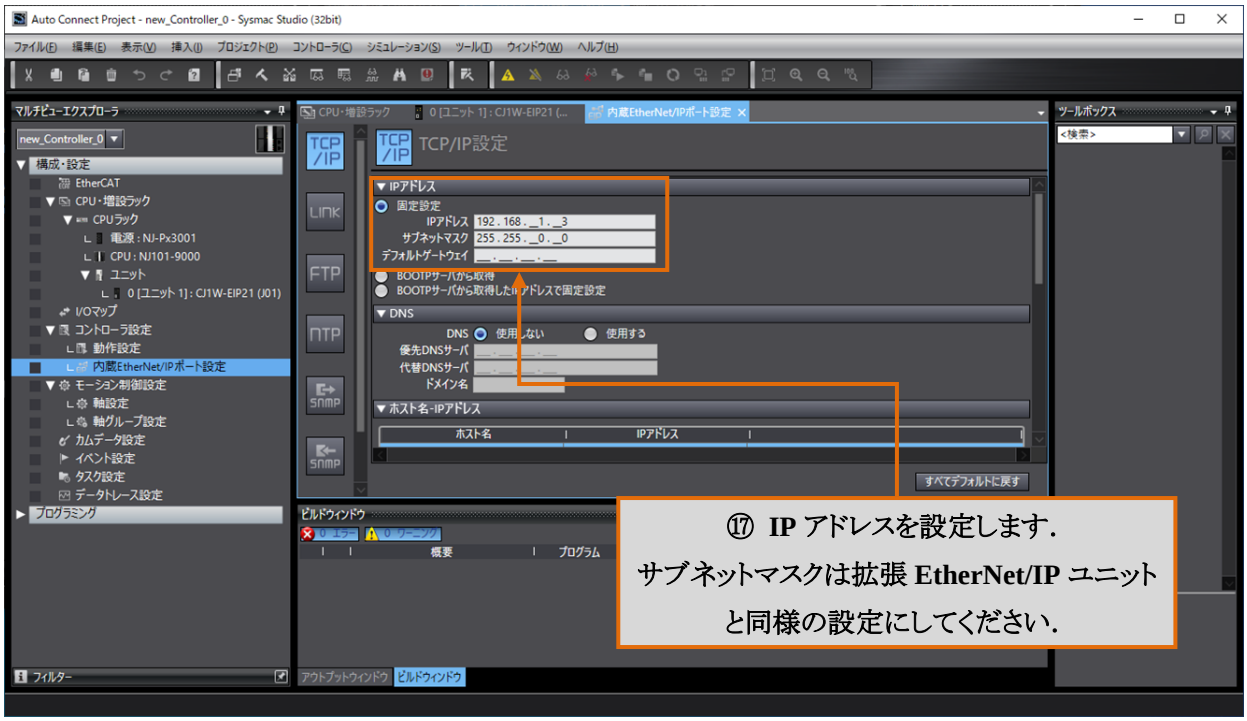

⑫ [高性能ユニット設定の編集]
ボタンを押下します.


⑬ 拡張 EtherNet/IP ユニットの IP アドレスを設定します. IP アドレスの第4オクテッド(赤枠)は実機の拡張ユニットのノード番号と揃えて設定します.

9.2.1.2. CPU ユニットの設定方法

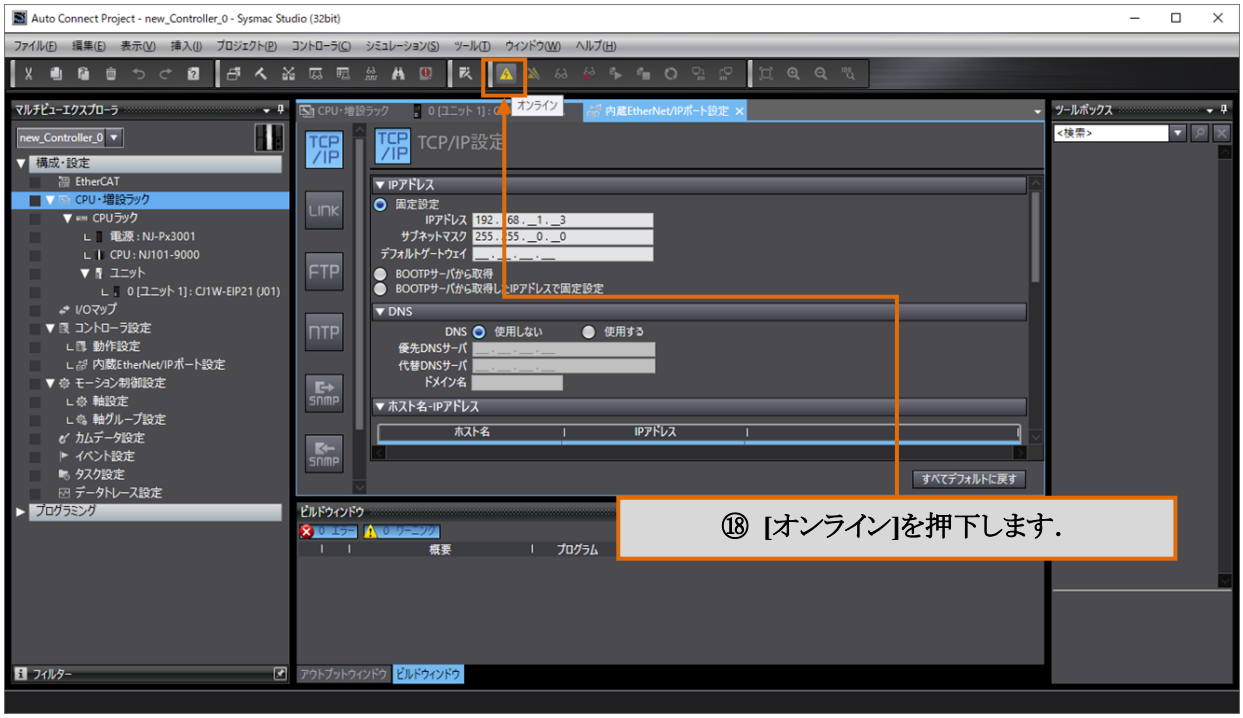
CPU ユニットの内蔵 EtherNet/IP ポートの設定を行います.

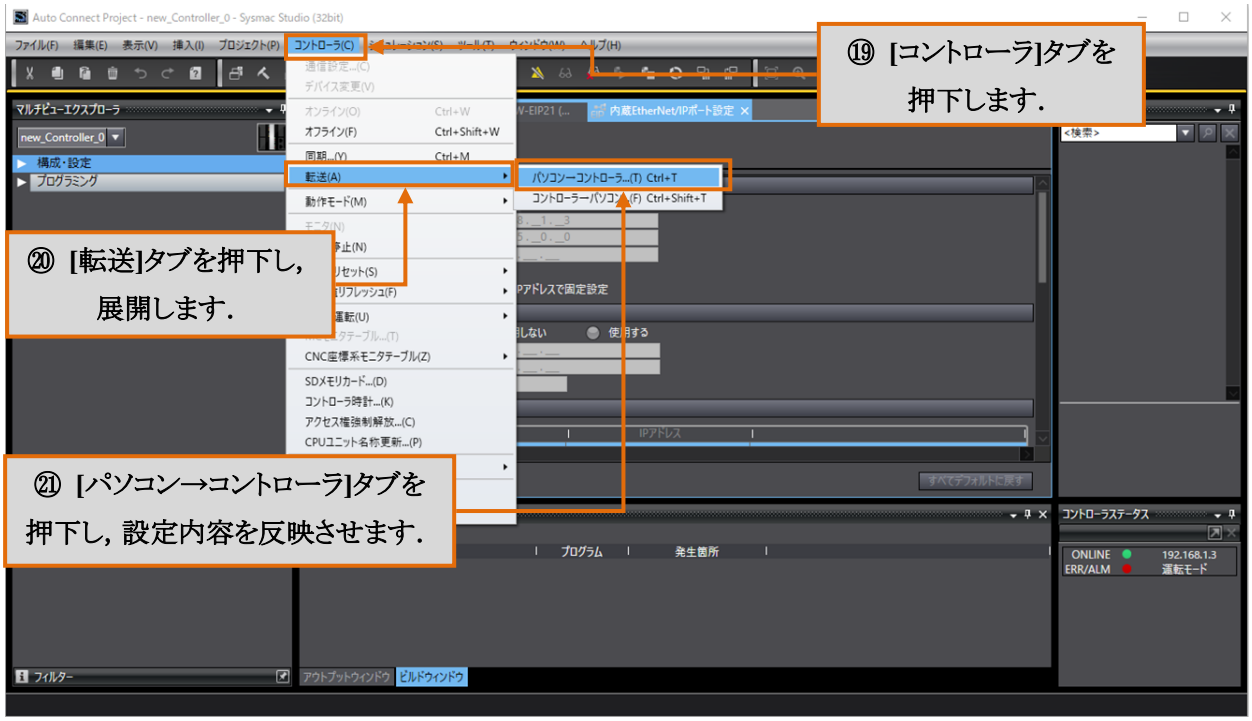




9.2.1.3. 設定内容を機器に反映

各ユニットの設定が完了したら，PLC に設定内容を反映させます．



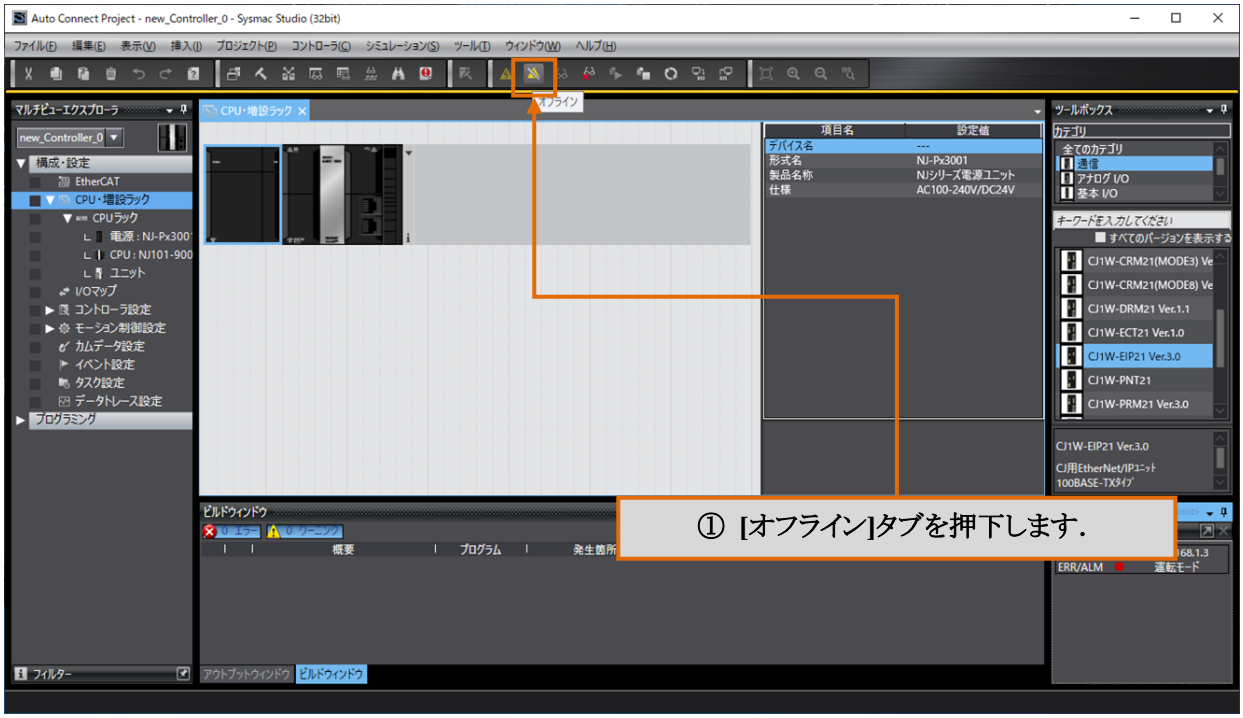


PLC へ転送が完了したら，PLC の電源を再起動させ，PLC にエラーが出ていなければ構成設定は完了です．

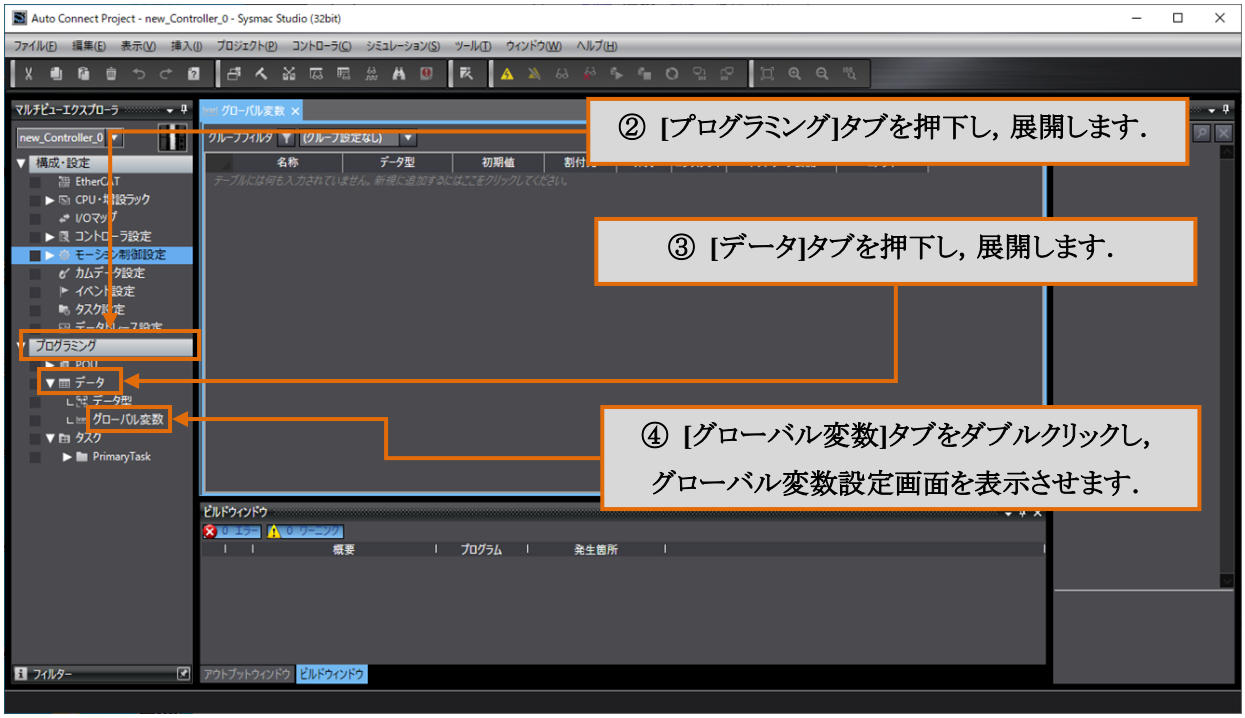
9.2.2. 変数登録

タグデータリンクを行うためのグローバル変数を親 PLC に登録します.

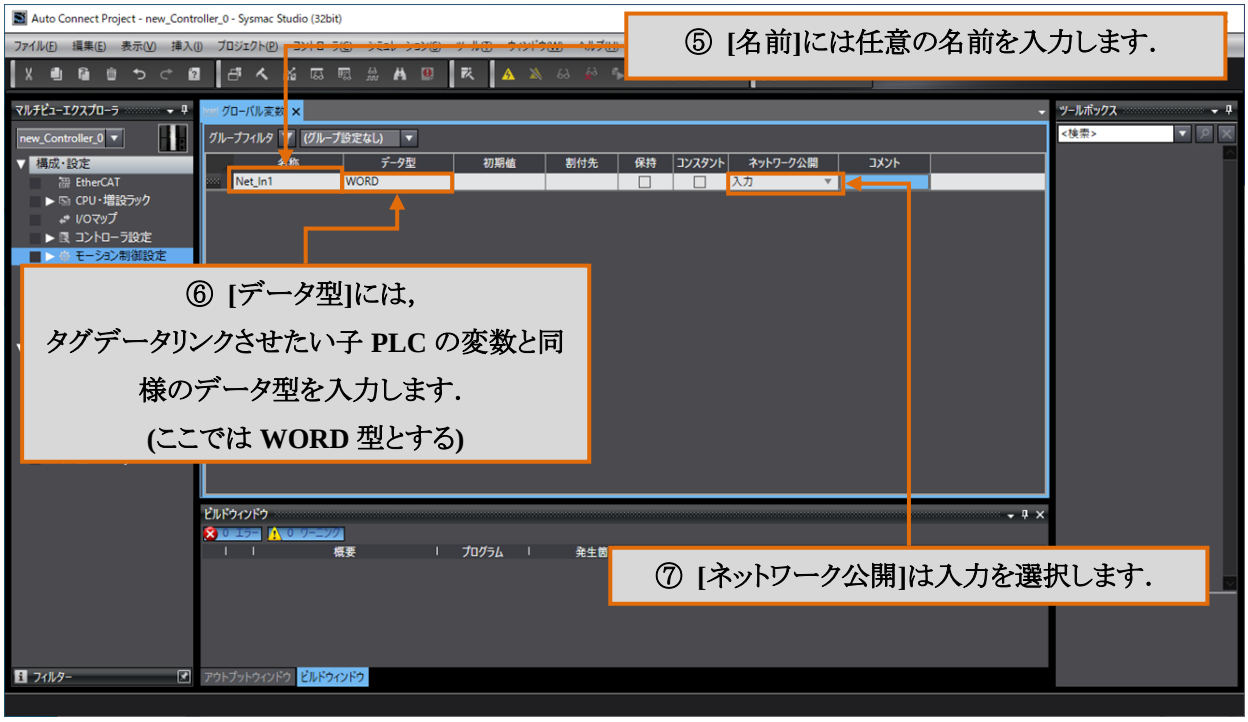
設定の編集を行うため「9.2.1.構成設定」同様に, オンラインであればオフラインにします.



次に, グローバル変数を登録します.



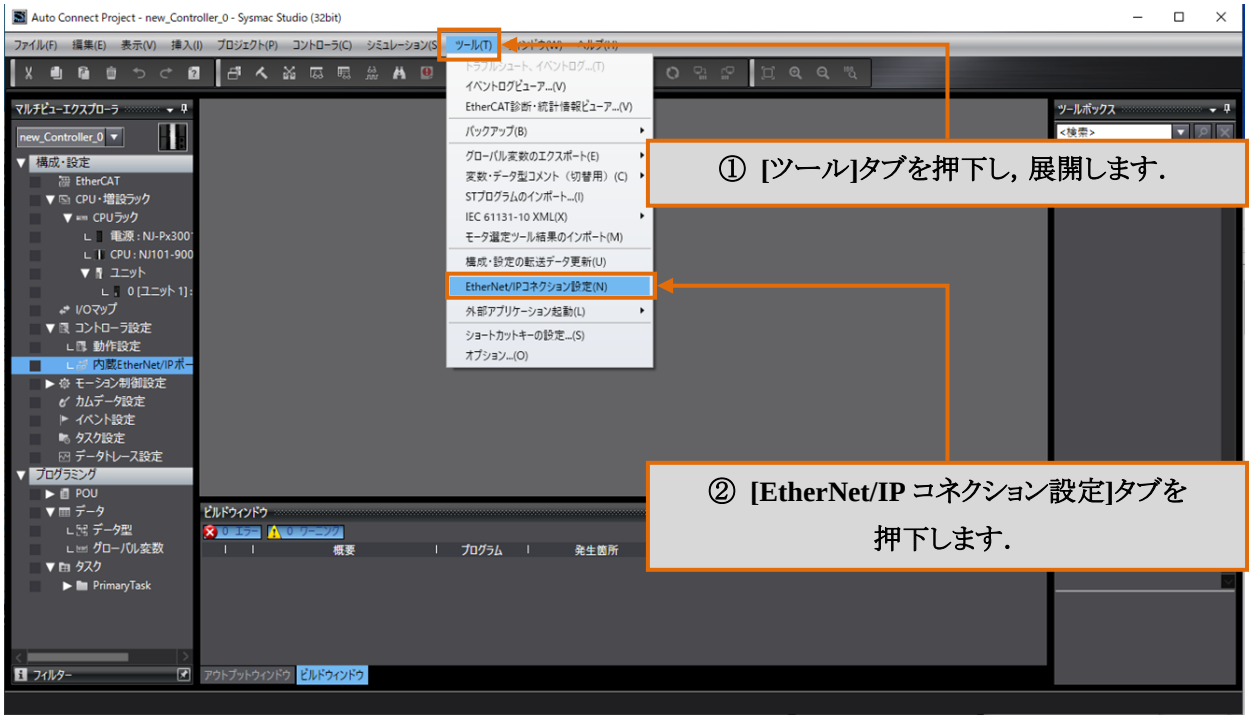
タグデータリンクさせたいグローバル変数を登録し，下記のように設定します．下記の項目以外は特に設定する必要はありません．

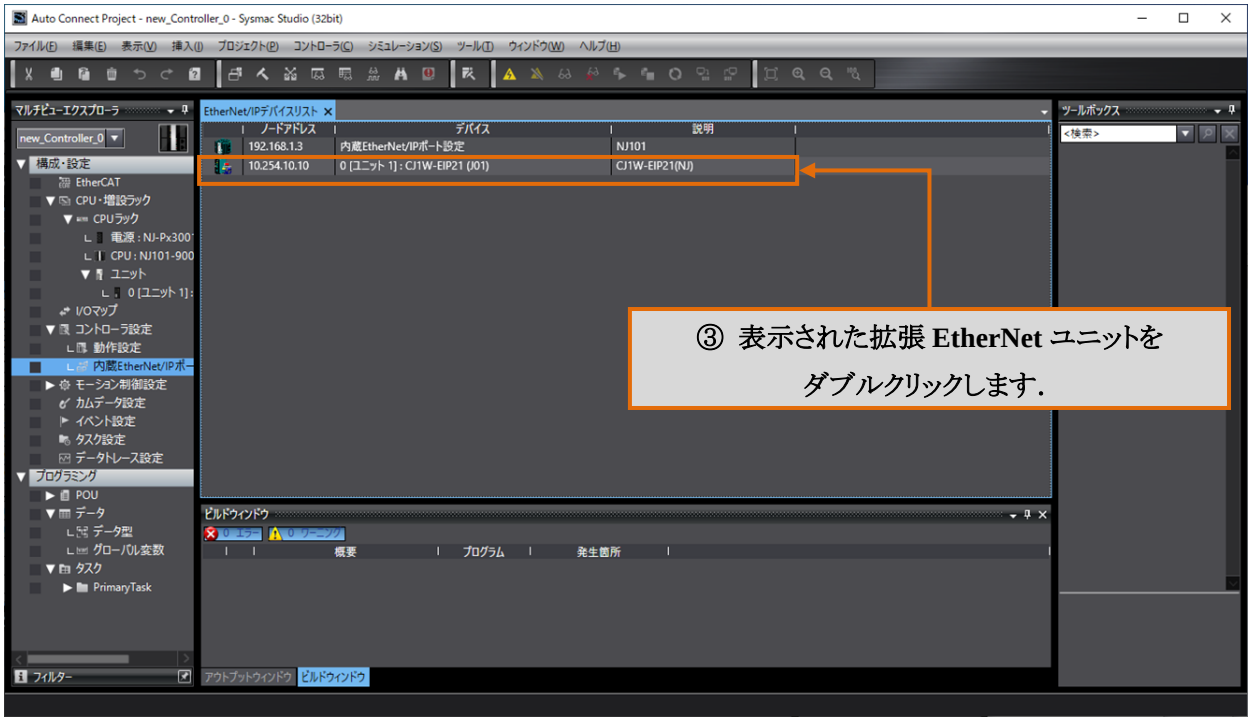


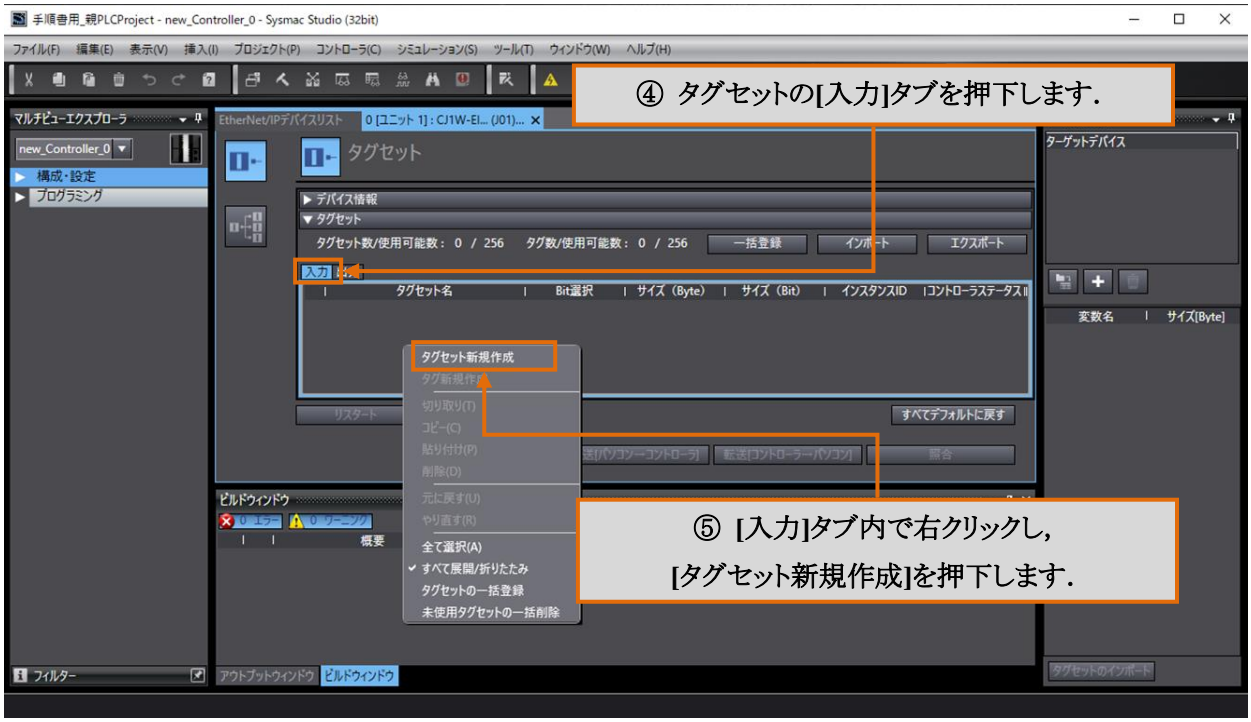
「9.2.1.3.設定内容を機器に反映」と同様に設定内容を親 PLC に反映させます．完了したら変数の登録は以上となります．

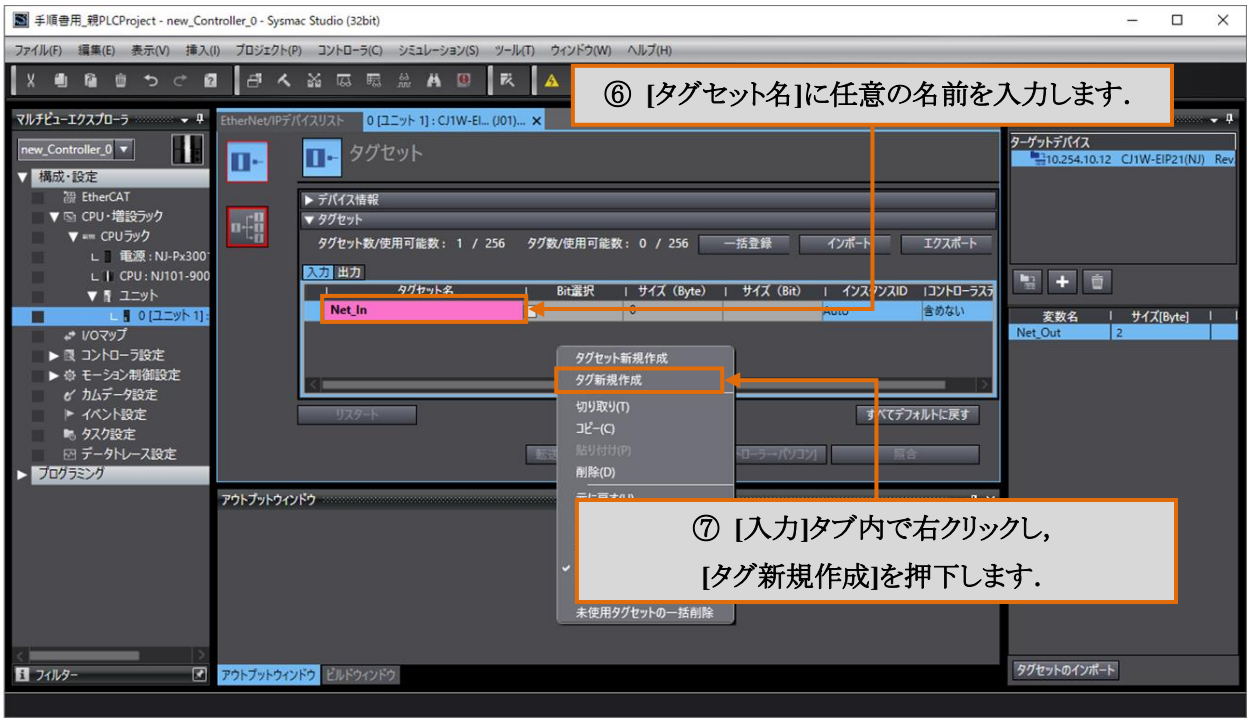
9.2.3. EtherNet/IP コネクション設定

EtherNet/IP コネクション設定から，タグデータリンクを行うためのタグセットの設定を行います．この節では，子 PLC の設定をした Sysmac Studio のプロジェクトを使用しますので，「9.3.子 PLC の設定方法」を完了させ，設定時に使用した Sysmac Studio プロジェクトを保存しておいて下さい．

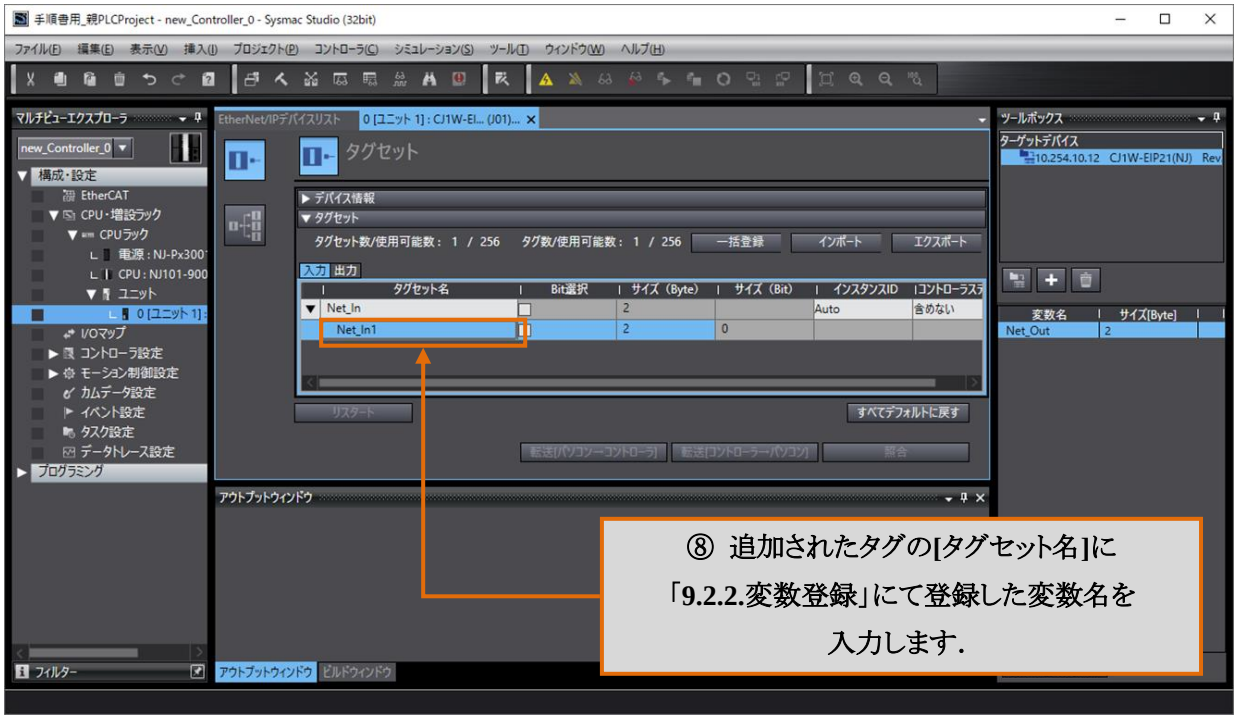




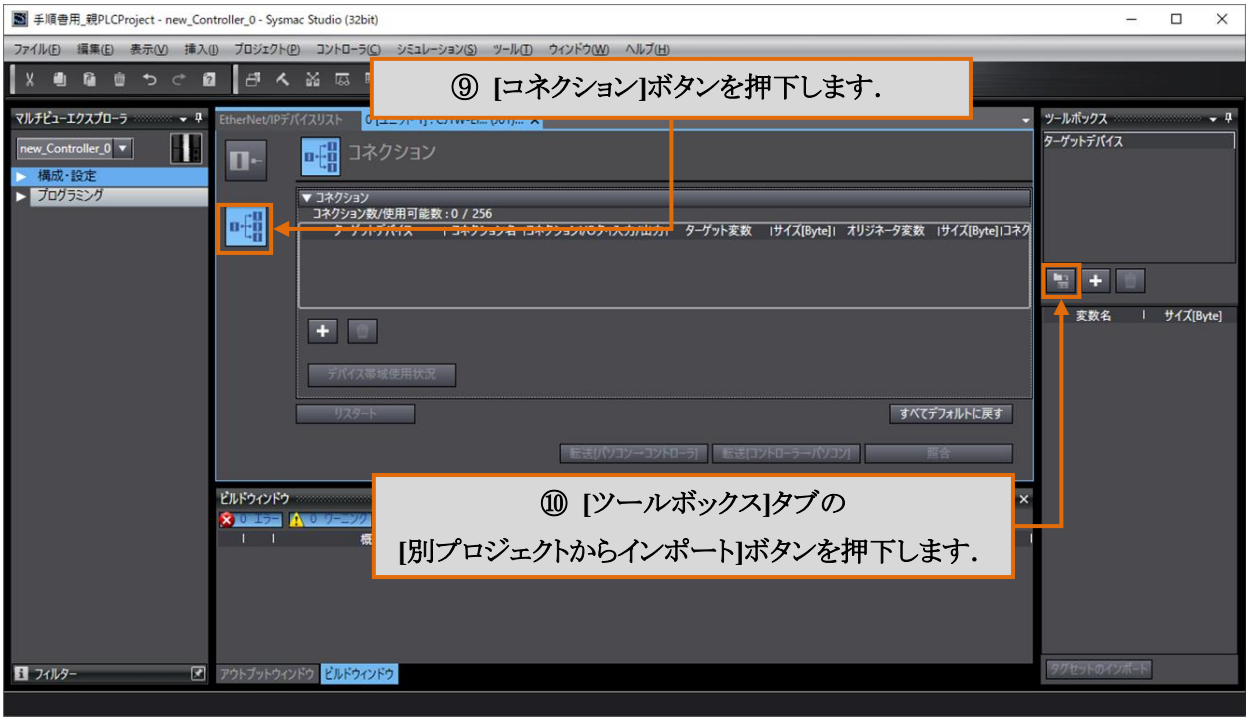

④ タグセットの[入力]タブを押下します.
⑤ [入力]タブ内で右クリックし,
[タグセット新規作成]を押下します.
タグセット新規作成
タグ新規作成
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
元に戻す(U)
やり直す(R)
全て選択(A)
すべて展開/折りたたみ
タグセットの一括登録
未使用タグセットの一括削除


⑥ [タグセット名]に任意の名前を入力します.
⑦ [入力]タブ内で右クリックし,
[タグ新規作成]を押下します.
Net_In
タグセット新規作成
タグ新規作成
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
未使用タグセットの一括削除

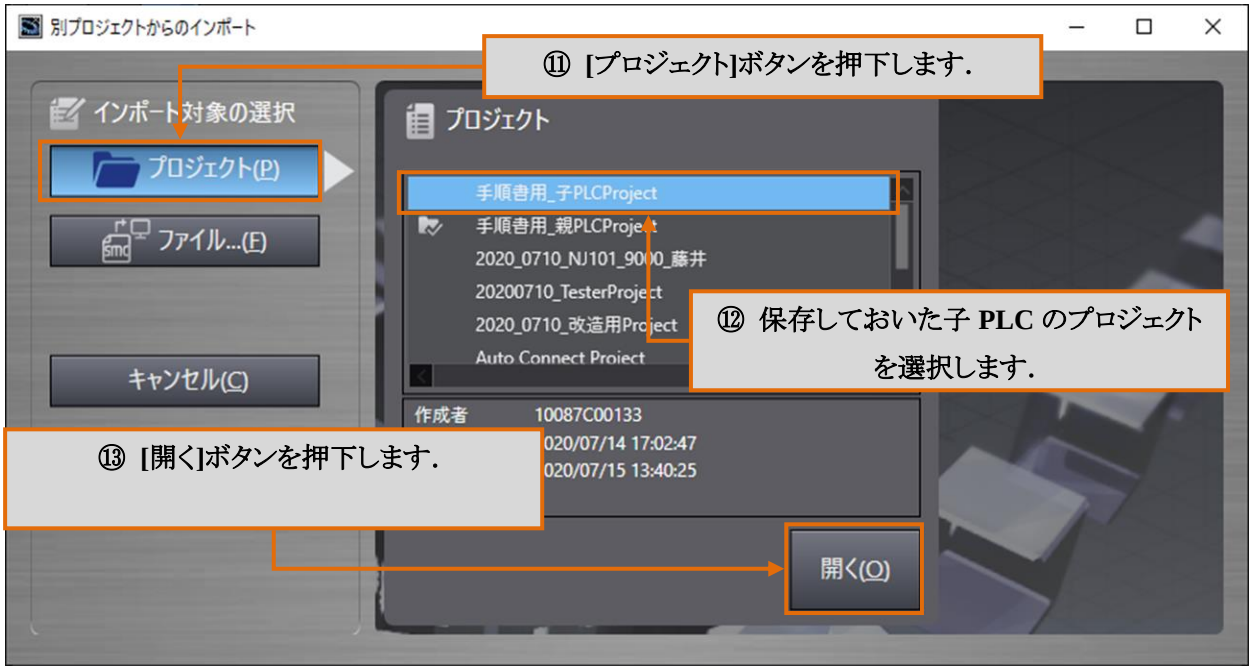
タグセットにタグデータリンクさせる変数を設定します．タグセットに複数個の変数を登録することも可能ですが親 PLC と子 PLC で同じサイズ，データ型，タグ数を合わせる必要があります．

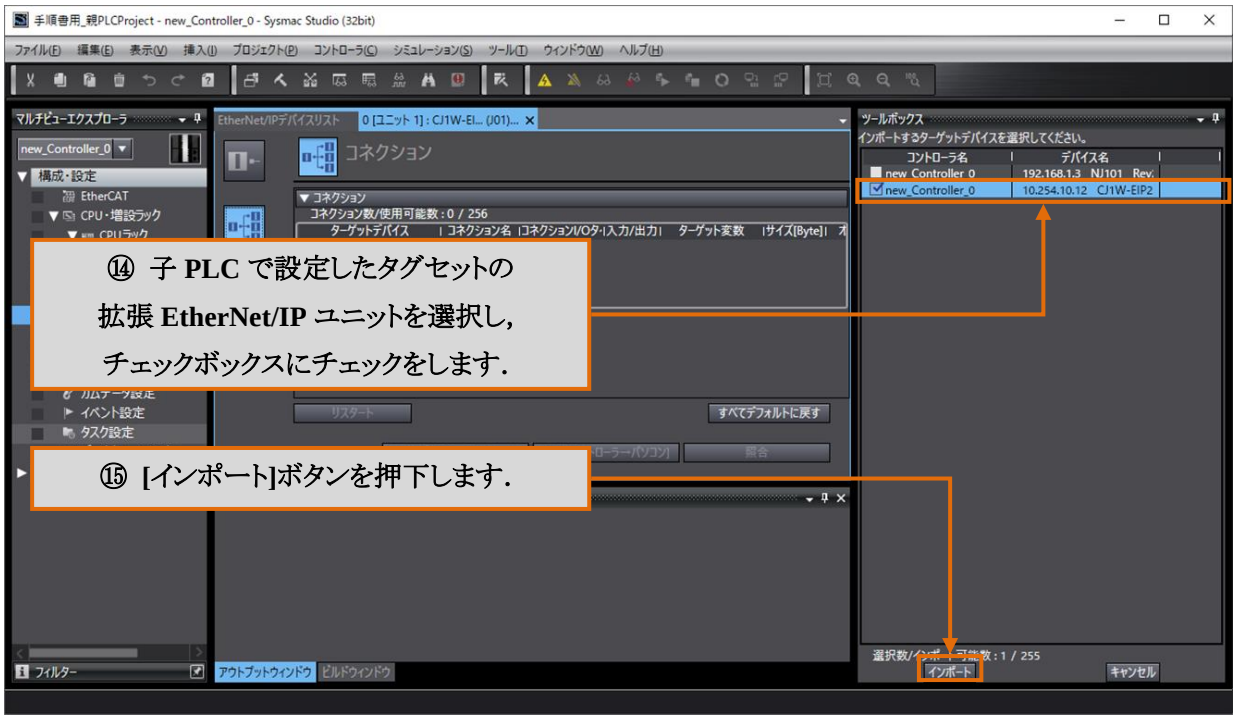


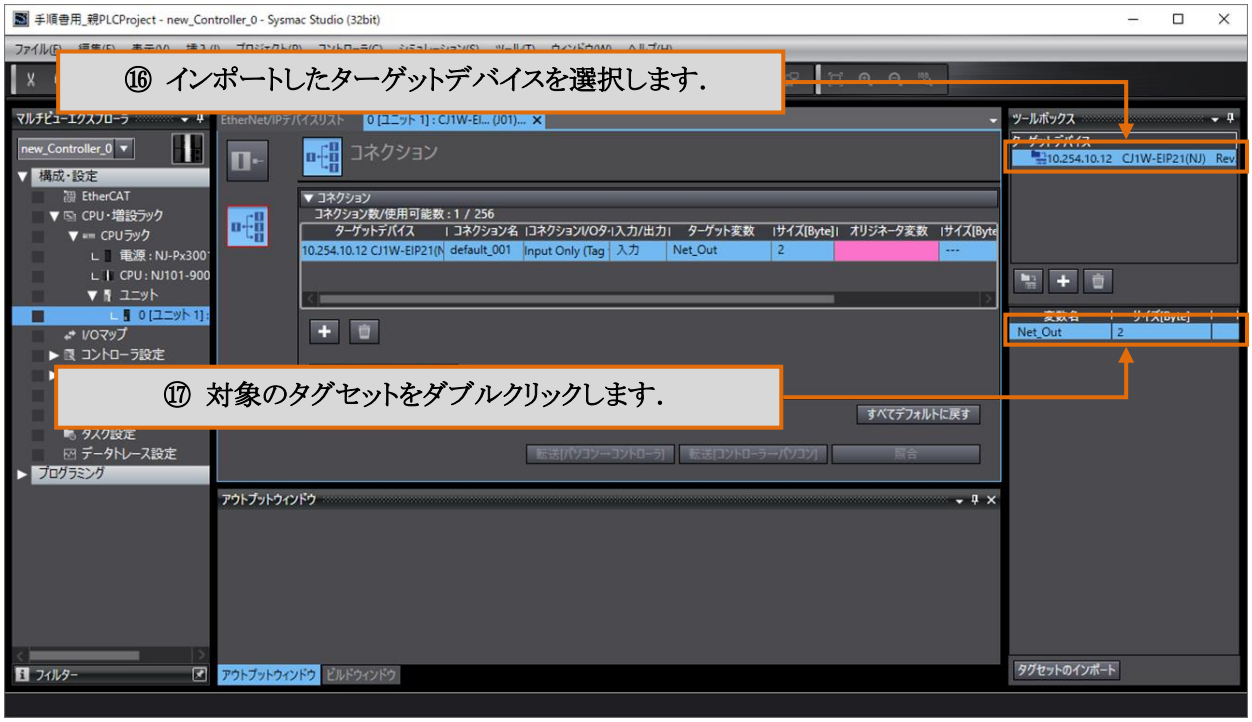
次に，設定したタグセットを対象の子 PLC に設定したタグセットとコネクションさせます．

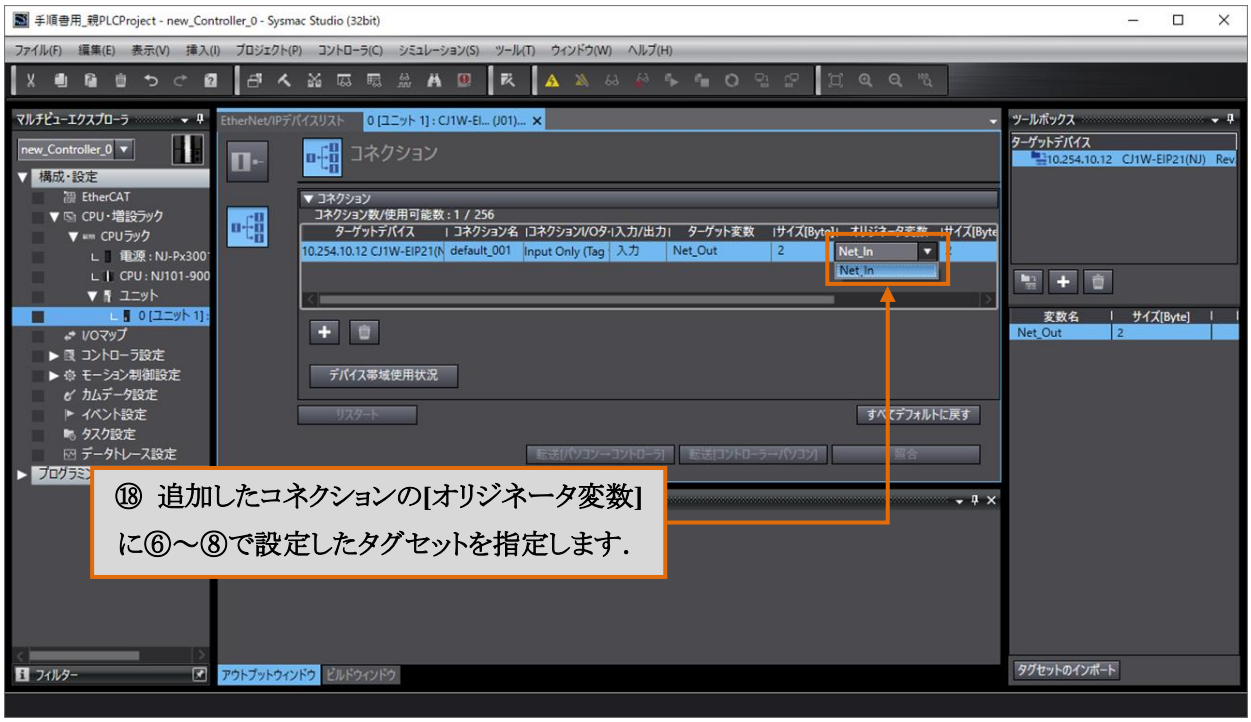


設定した子 PLC のプロジェクトからタグセット情報を読み込みます.

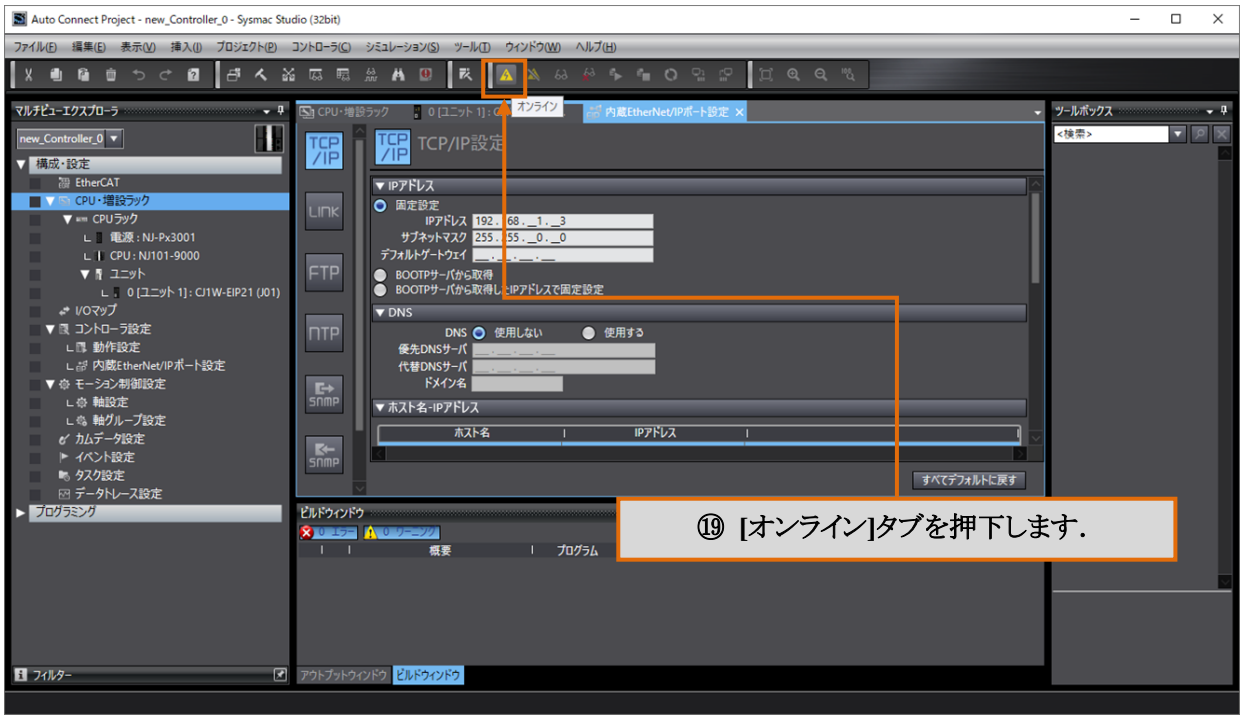


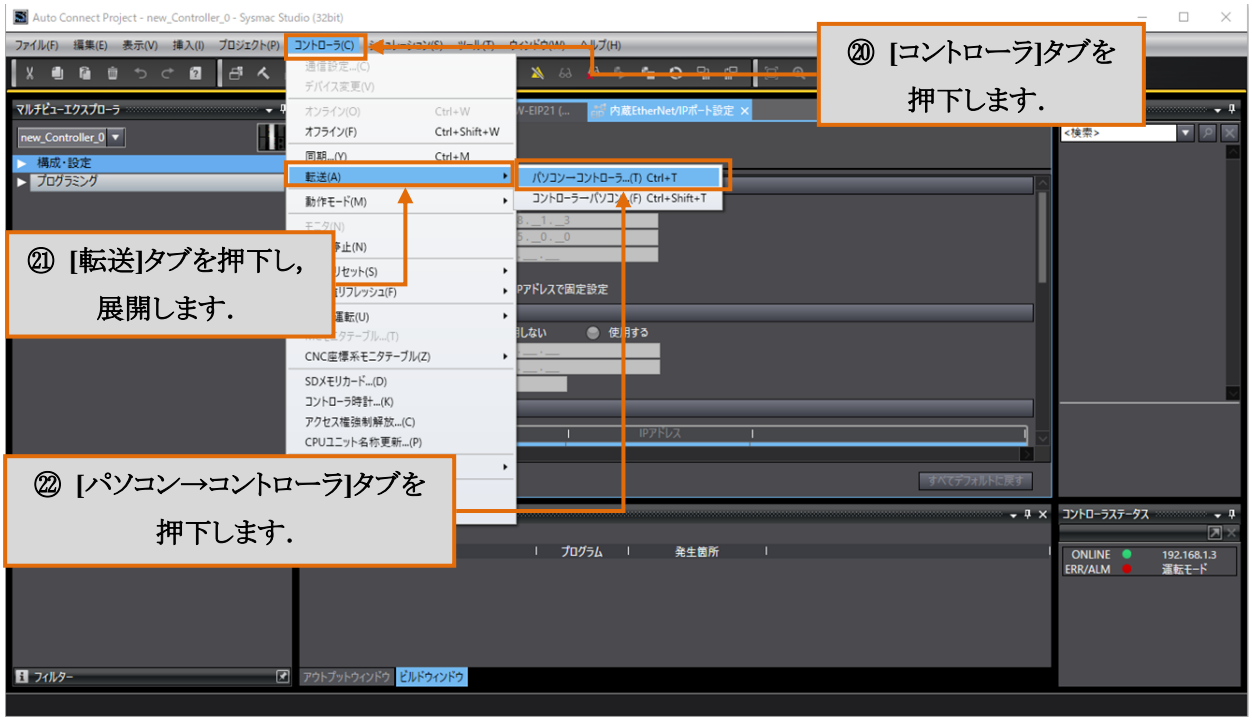



⑯ インポートしたターゲットデバイスを選択します.
⑰ 対象のタグセットをダブルクリックします.

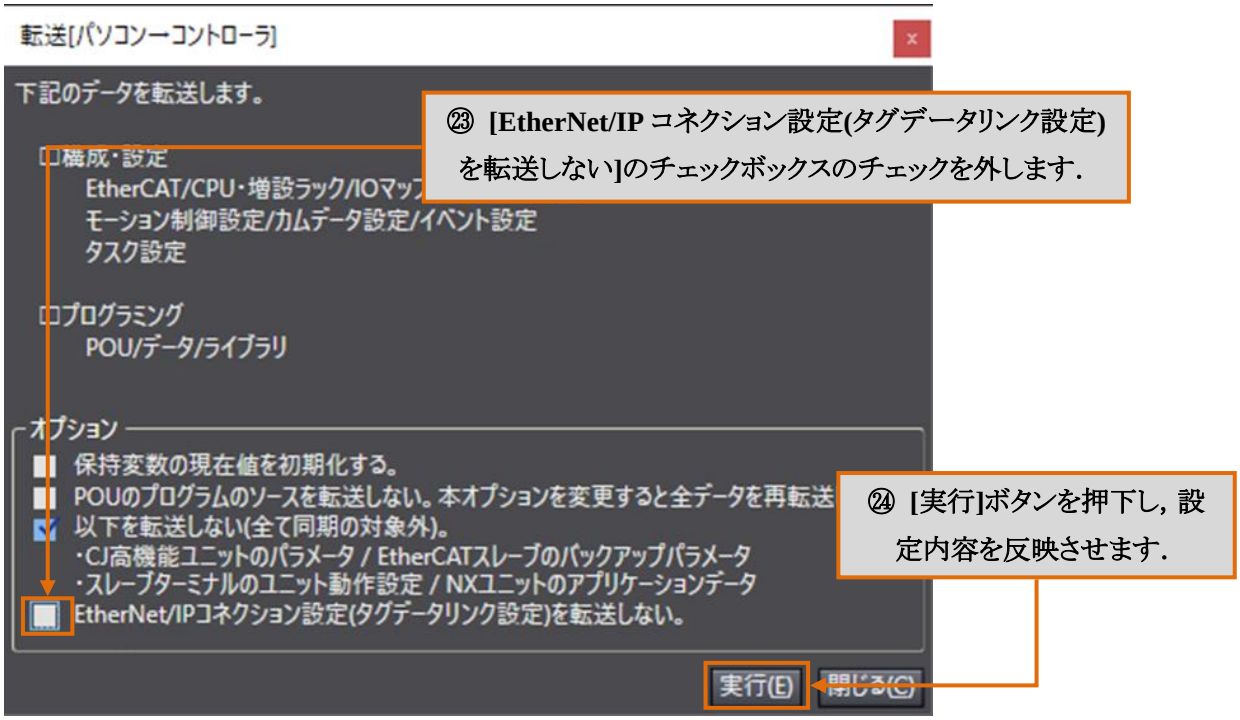

⑱ 追加したコネクションの[オリジネータ変数]
に⑥～⑧で設定したタグセットを指定します.

タグセットの設定が完了したら，親 PLC に設定内容を反映させます．





パソコンから転送をする際に下記のようなウィンドウが表示されるので，該当のチェックボックスを外し転送してください．



以上で親 PLC の設定は完了となります．

9.3. 子 PLC の設定方法

本節では，Sysmac Studio を用いて子 PLC 側の設定を行います．

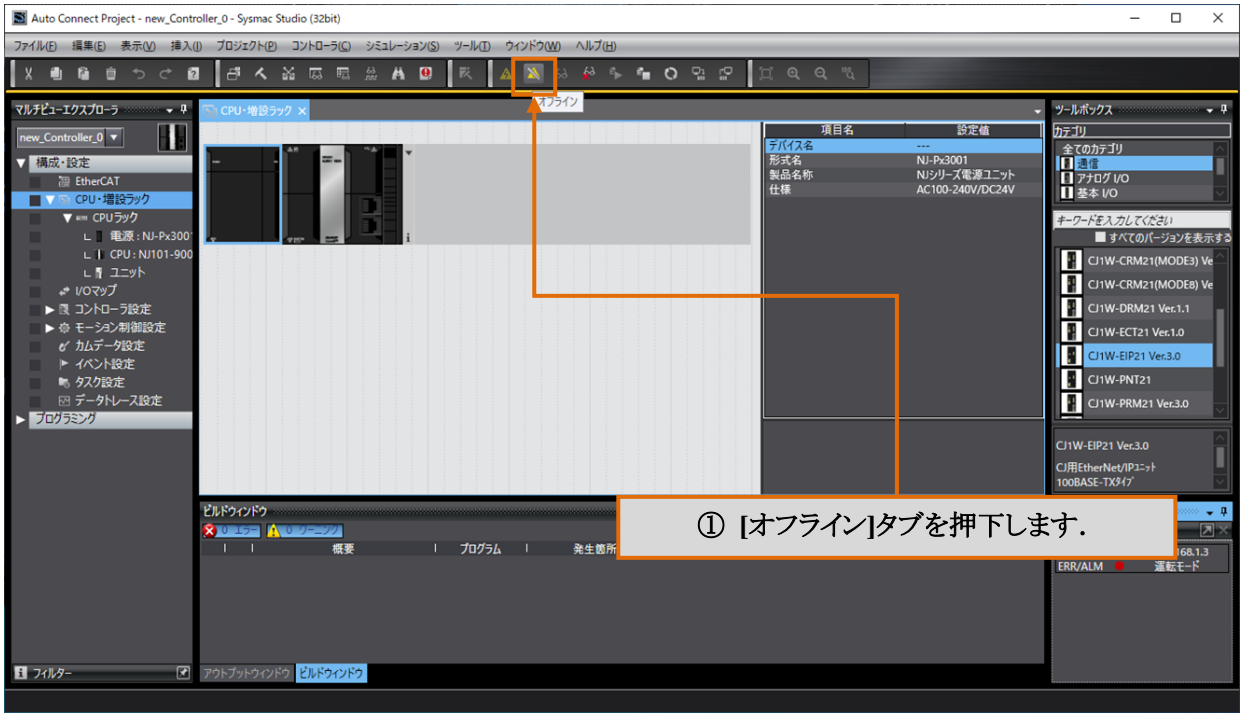
9.3.1. 構成設定

子 PLC の構成設定方法は，親 PLC の構成設定と同様です．詳細は「9.2.1.構成設定」を参照してください．ただし IP アドレスなどは親 PLC の設定と被らないように設定する必要があります．

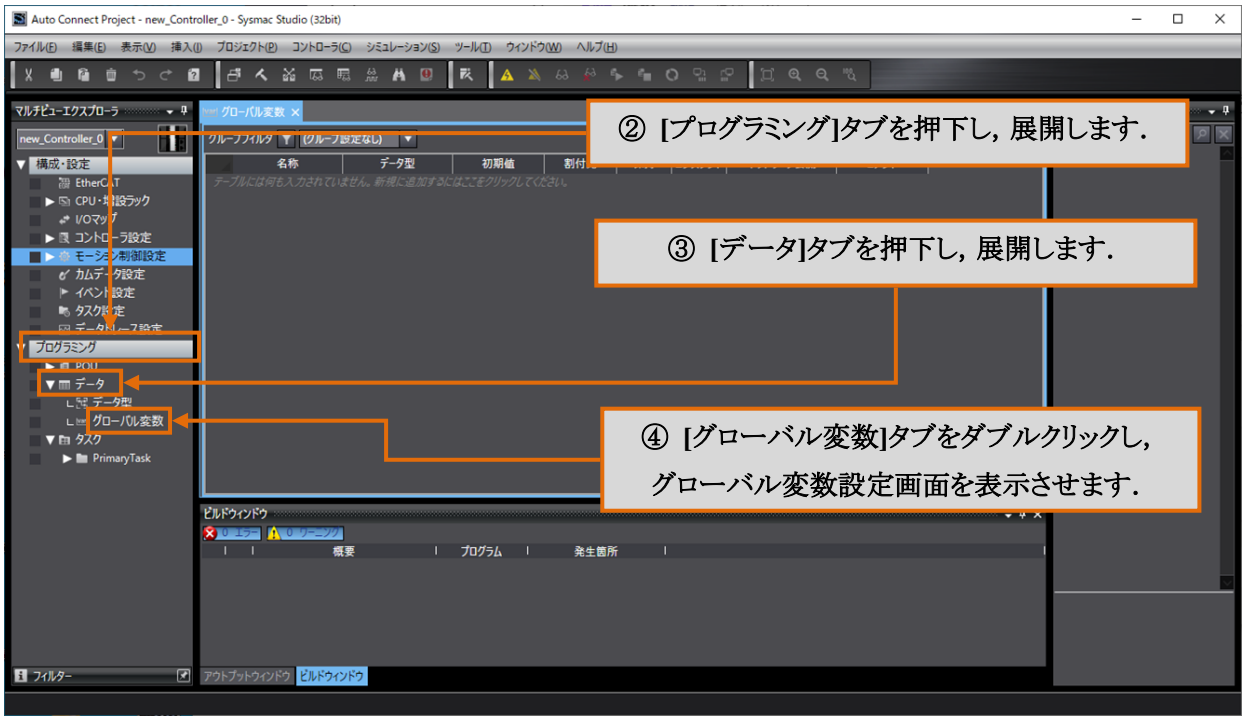
9.3.2. 変数登録

タグデータリンクを行うためのグローバル変数を子 PLC に登録し，設定します．

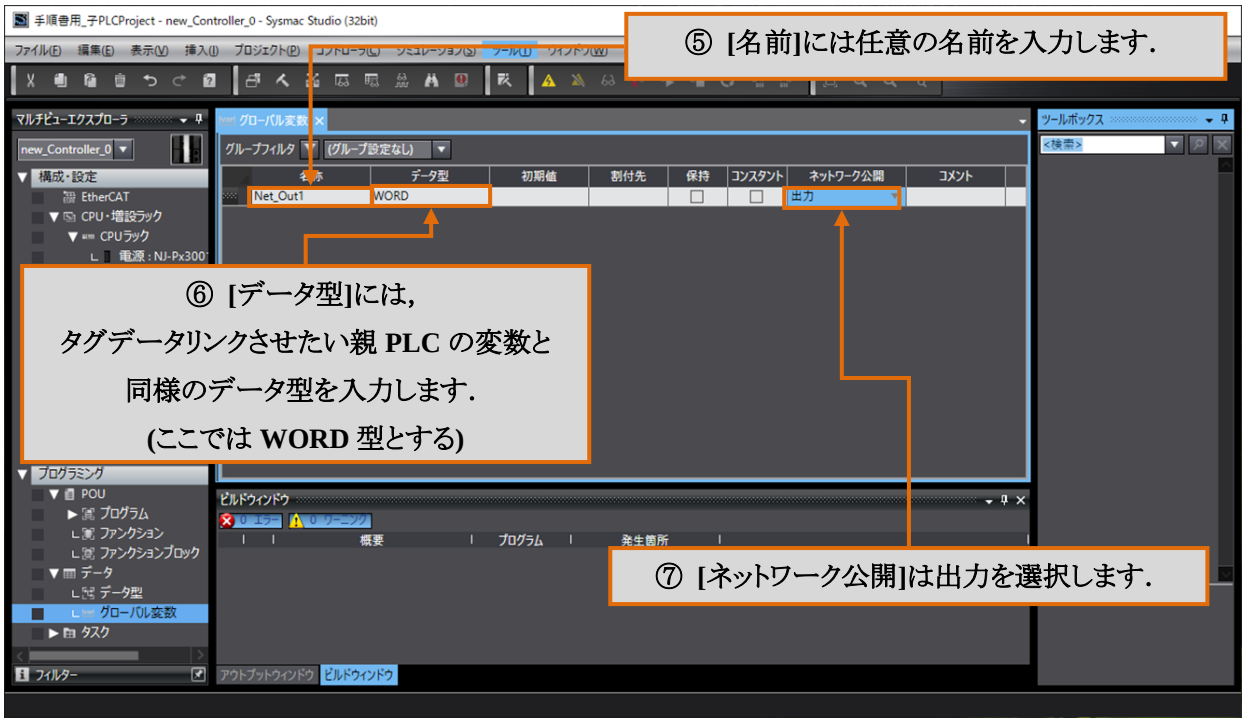
設定の編集を行うため「9.2.1.構成設定」同様に，オンラインであればオフラインにします．



次に，グローバル変数を登録します．



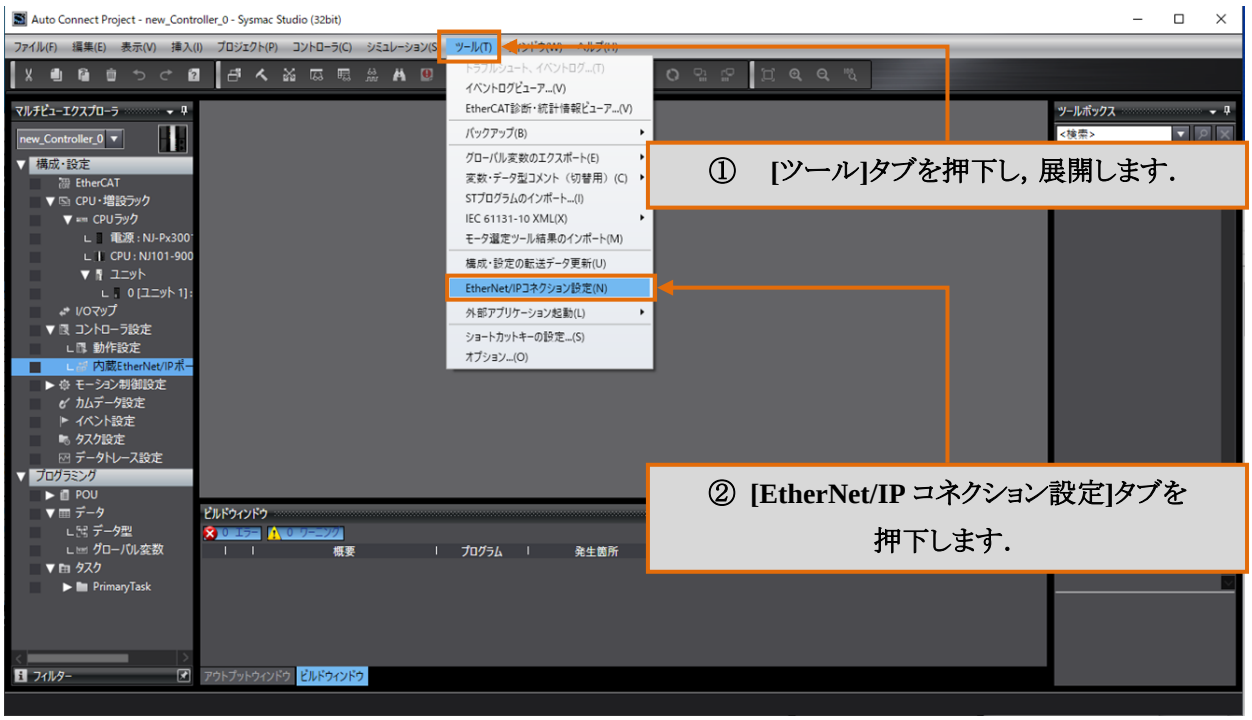
タグデータリンクさせたいグローバル変数を登録し，下記のように設定します．下記の項目以外は特に設定する必要はありません．

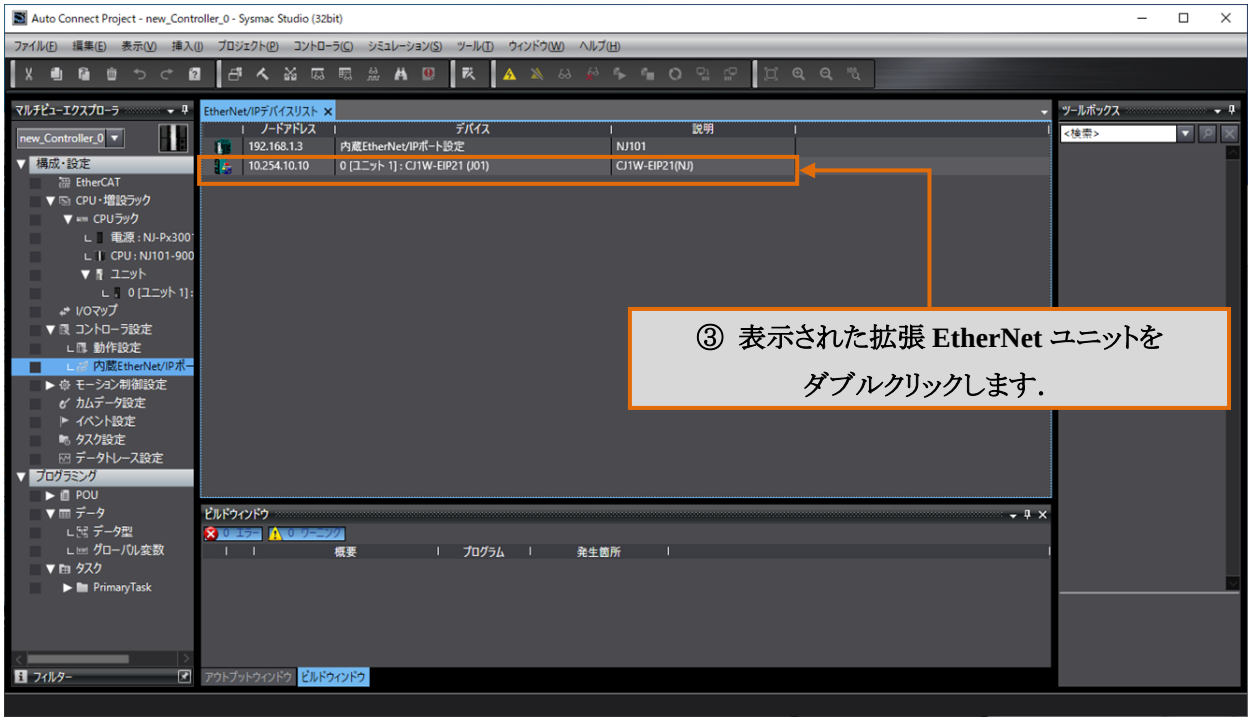


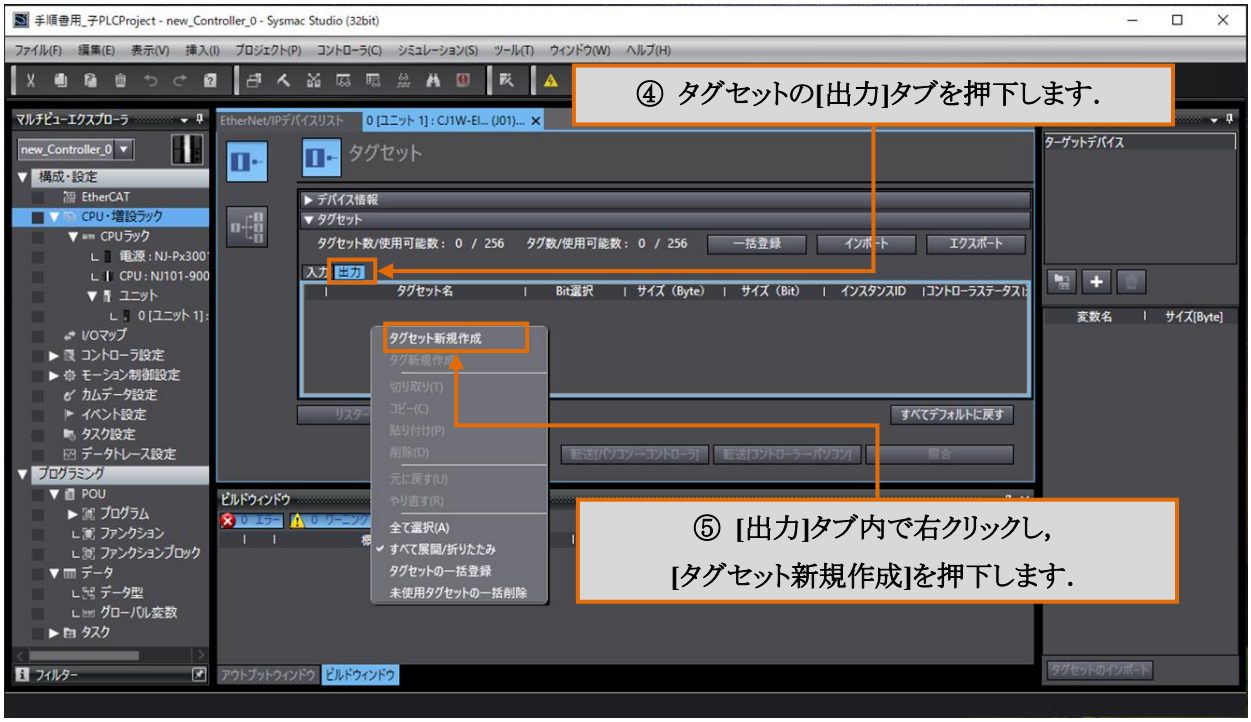
変数の登録は以上です．「9.2.1.3.設定内容を機器に反映」と同様に設定内容を子 PLC に反映させます．

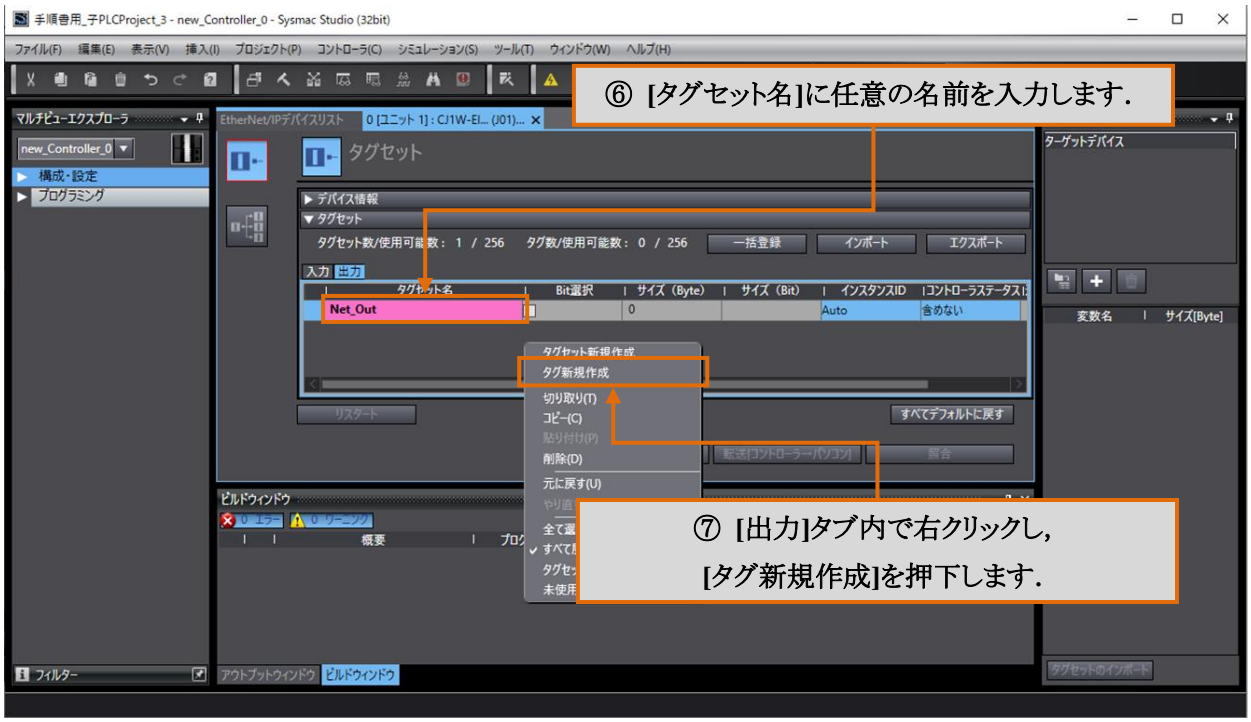
9.3.3. EtherNet/IP コネクション設定

タグデータリンクを行うためのタグセットの設定を行います.

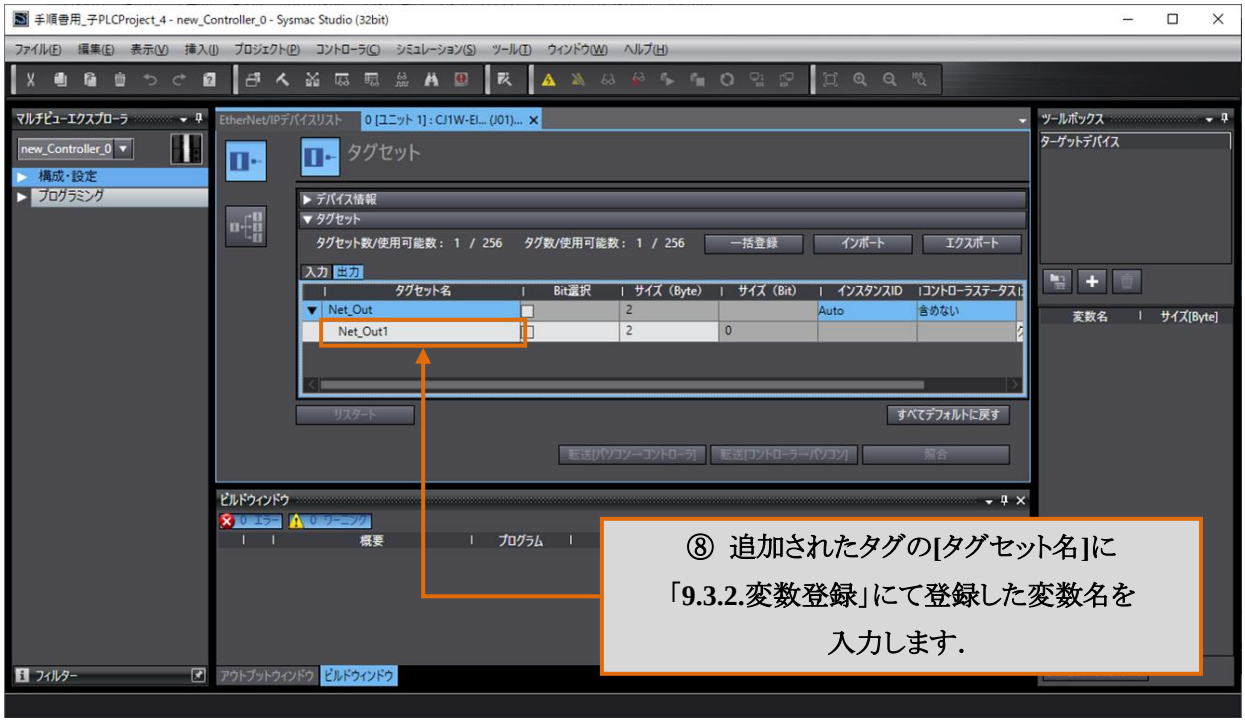




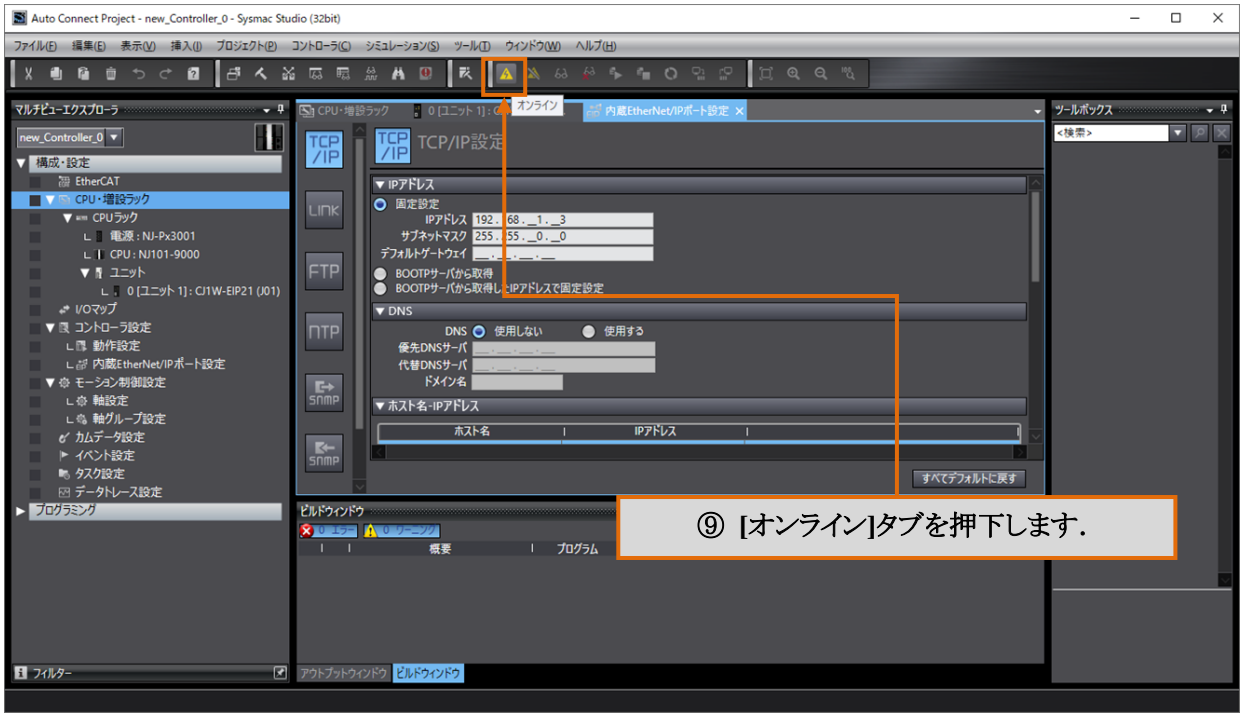

④ タグセットの[出力]タブを押下します.
⑤ [出力]タブ内で右クリックし,
[タグセット新規作成]を押下します.
タグセット新規作成
タグ新規作成
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
元に戻す(U)
やり直す(R)
全て選択(A)
すべて展開/折りたたみ
タグセットの一括登録
未使用タグセットの一括削除

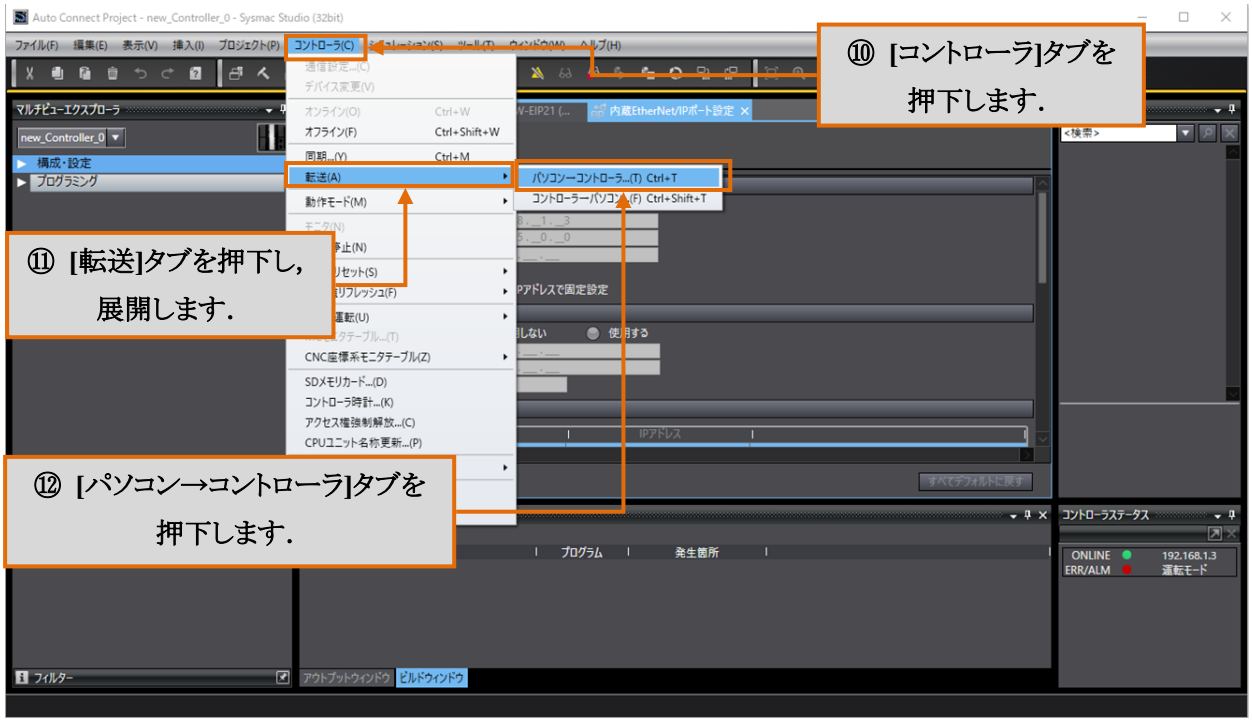

⑥ [タグセット名]に任意の名前を入力します.
⑦ [出力]タブ内で右クリックし,
[タグ新規作成]を押下します.
Net_Out
タグ新規作成
切り取り(T)
コピー(C)
貼り付け(P)
削除(D)
元に戻す(U)

タグセットにタグデータリンクさせる変数を設定します．タグセットに複数個の変数を登録することも可能ですが親 PLC と子 PLC で同じサイズ，データ型，タグ数を合わせる必要があります．

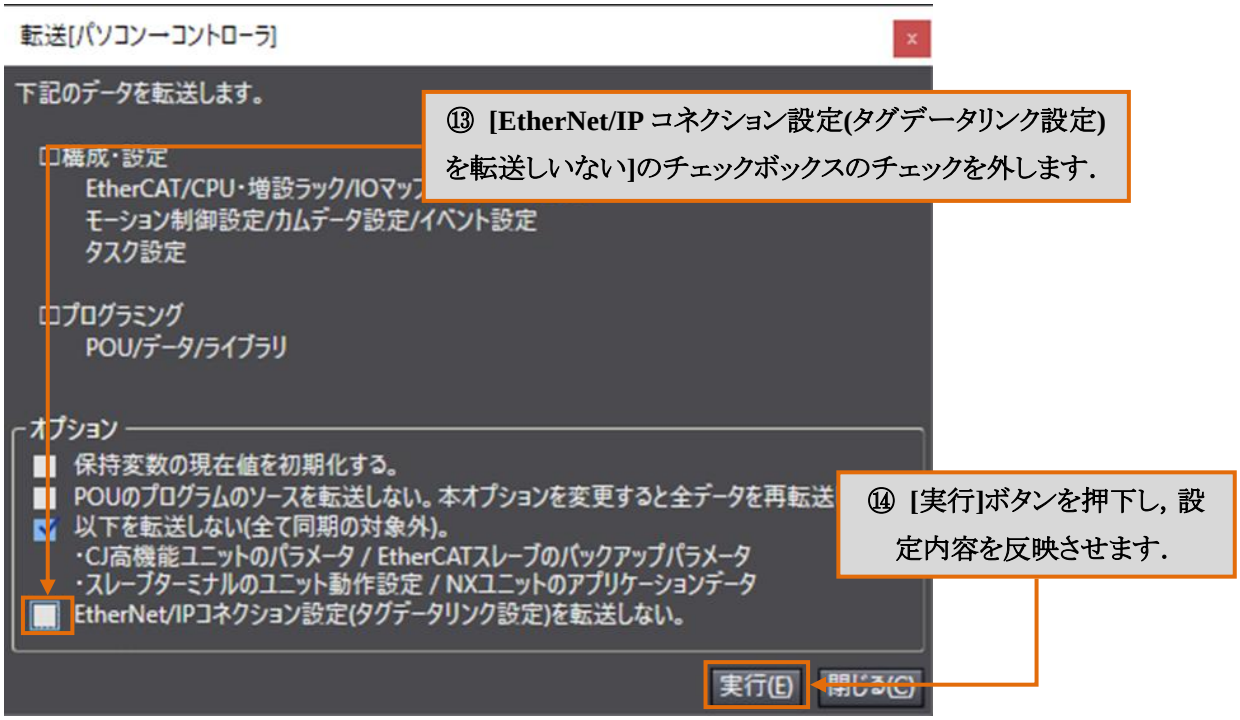


タグセットの設定が完了したら，子 PLC に設定内容を反映させます．





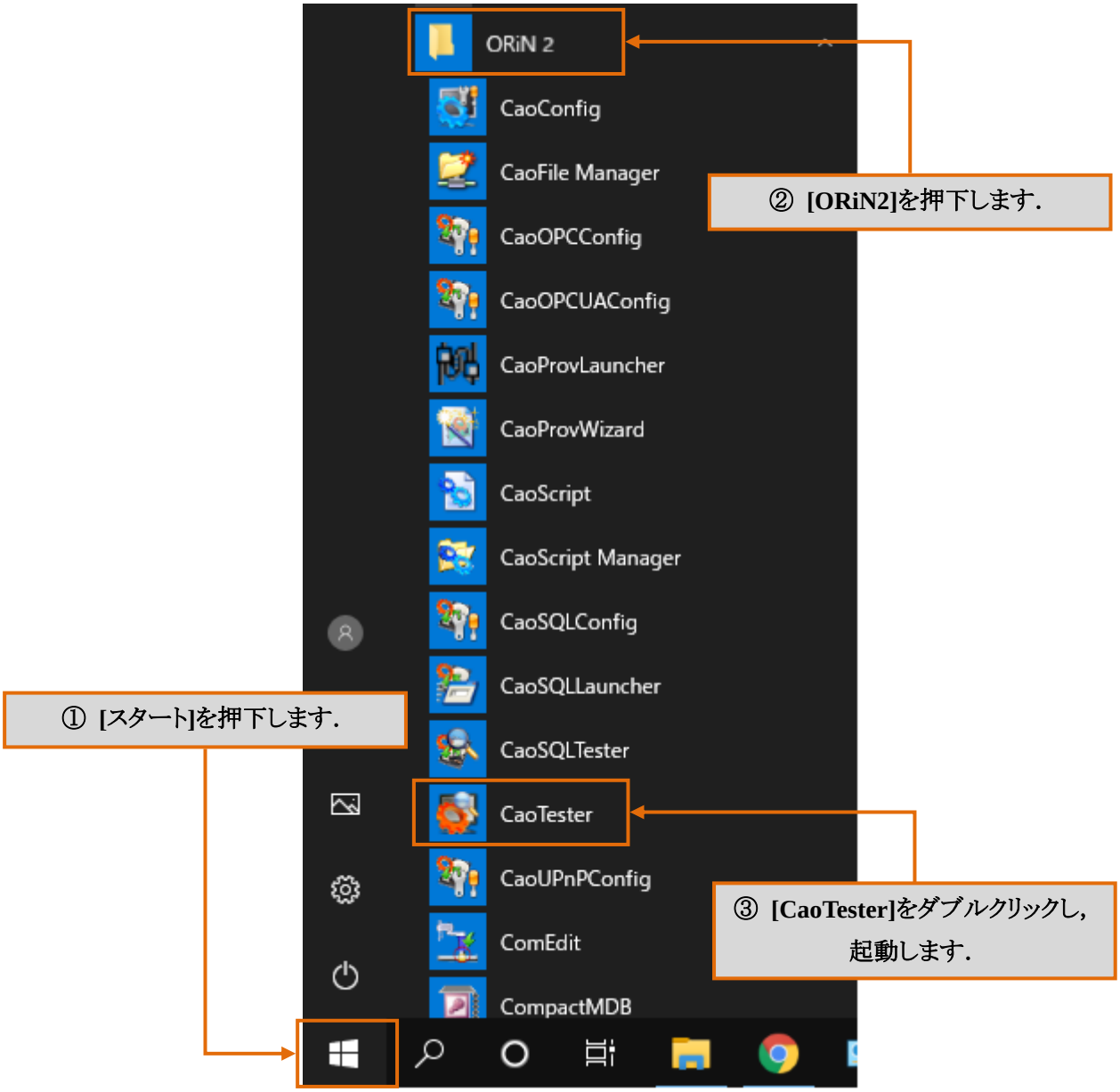
パソコンから転送をする際に下記のようなウィンドウが表示されるので，該当のチェックボックスを外し転送してください.



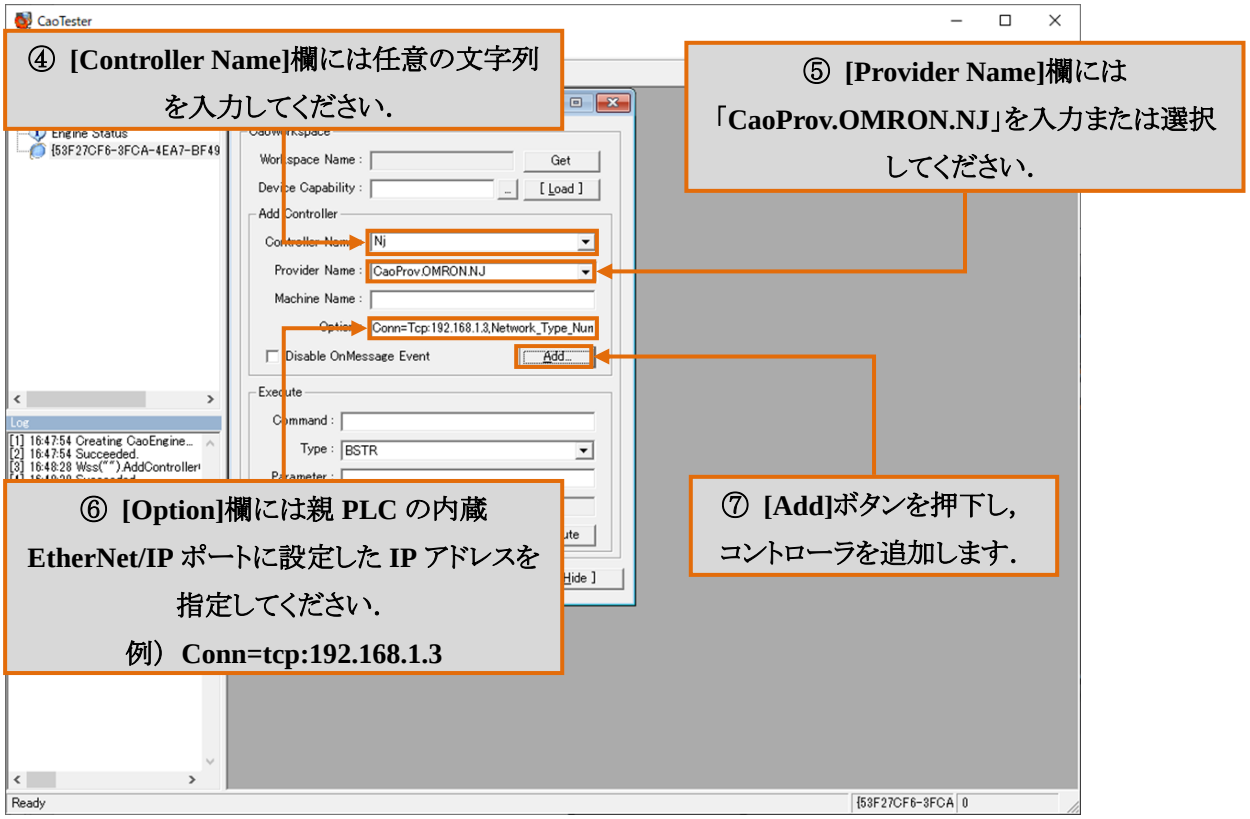
以上で子 PLC の設定は完了となります.

9.4. CaoTester にて値を取得

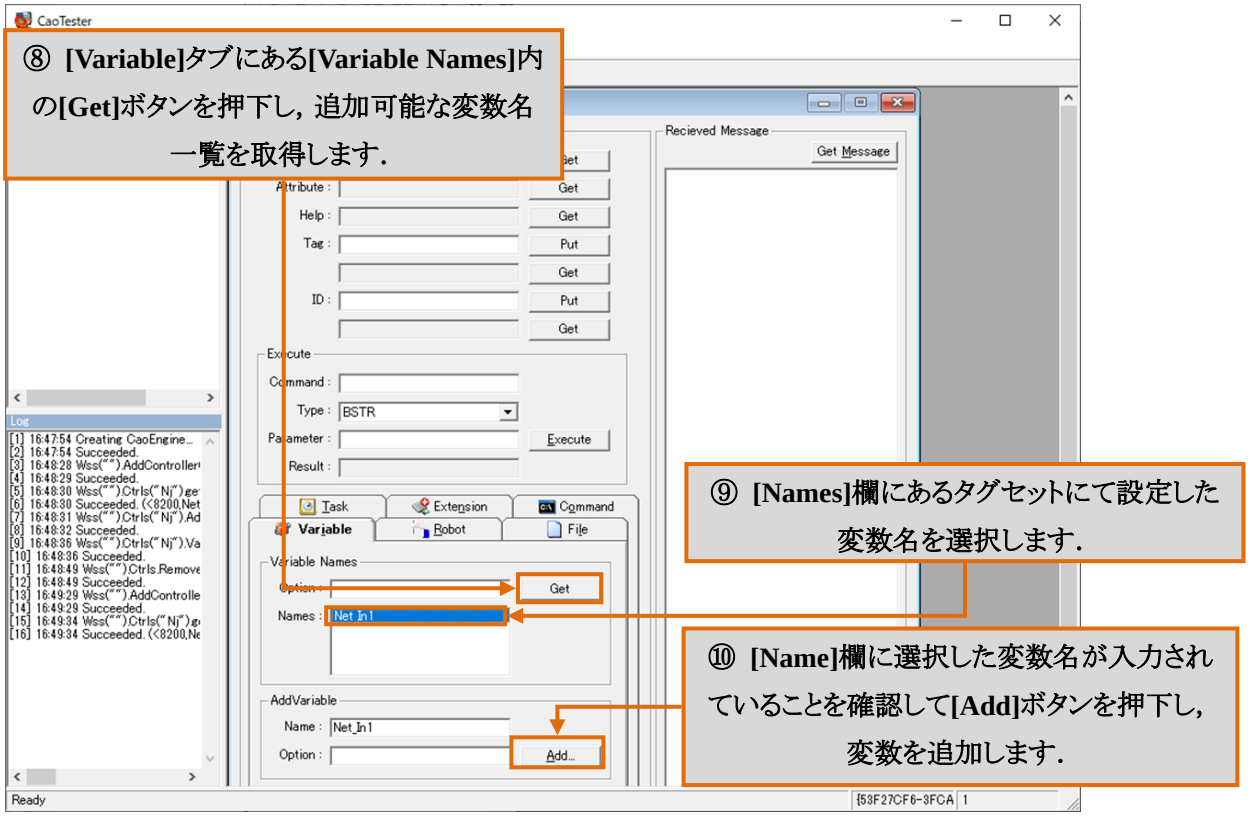
「9.2.親 PLC の設定方法」と「9.3.子 PLC の設定方法」で設定したタグデータを，実際に CaoTester にて OMRON.NJ シリーズプロバイダを使用して，値を取得します．



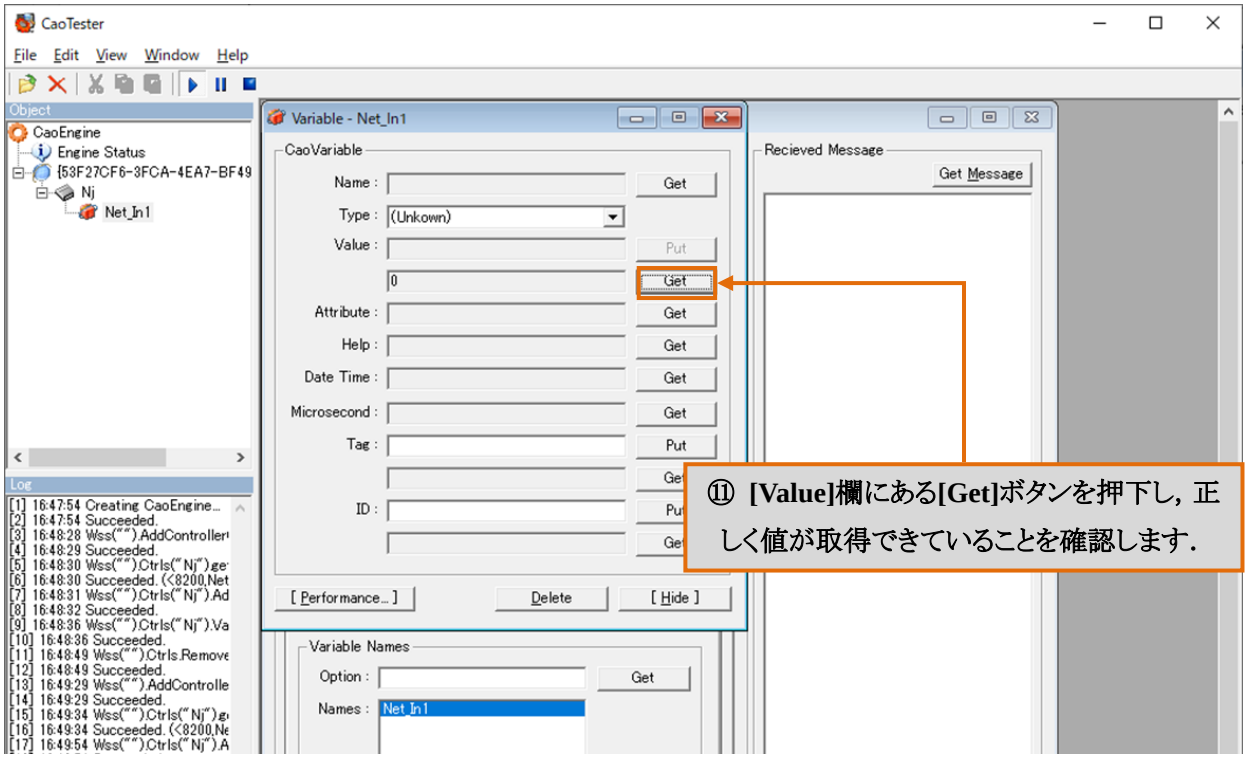
CaoTester を起動させた後，AddController を行います．



AddController 後に，AddVariable を行います．



最後に GetValue を実行して値を取得します.



タグデータリンクを用いての他局アクセス方法の手順は以上となります.