Welcome

PLCnext Controllerを使ったTCP/IPメッセージ(PLCnext Controller Setup)

注意事項

- 本内容は2025年2月時点での資料となります。

  本資料並びに本アプリケーションのサポートを将来的に約束するものではございません。

  また本内容における動作保証に関して当社は一切の責任を負いません。

使用しているバージョン

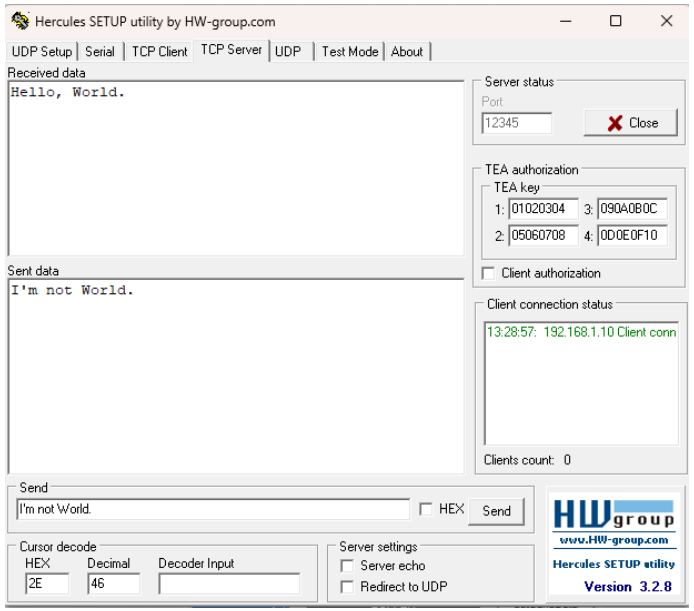
PLCnext Engineer 2024.6

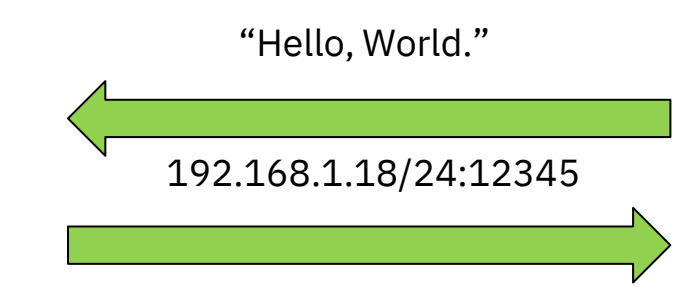
AXC F 2152 FW 2024.6.0

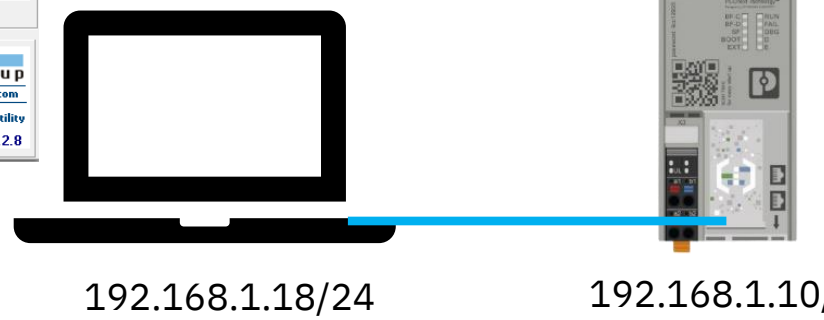
Hercules SETUP utility 3.2.8 (TCP/IP通信を確認するツール、フリーソフト)

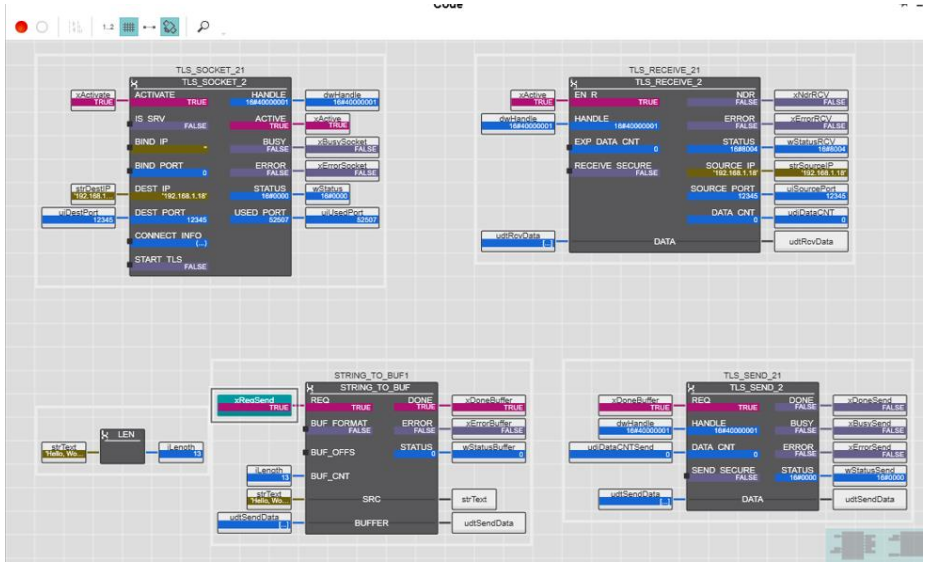
デモ構成

- AXC F 2152とPCをLANケーブルで接続し、AXC F2152⇔PC間でTCP/IP通信にてテストメッセージを送受信します。



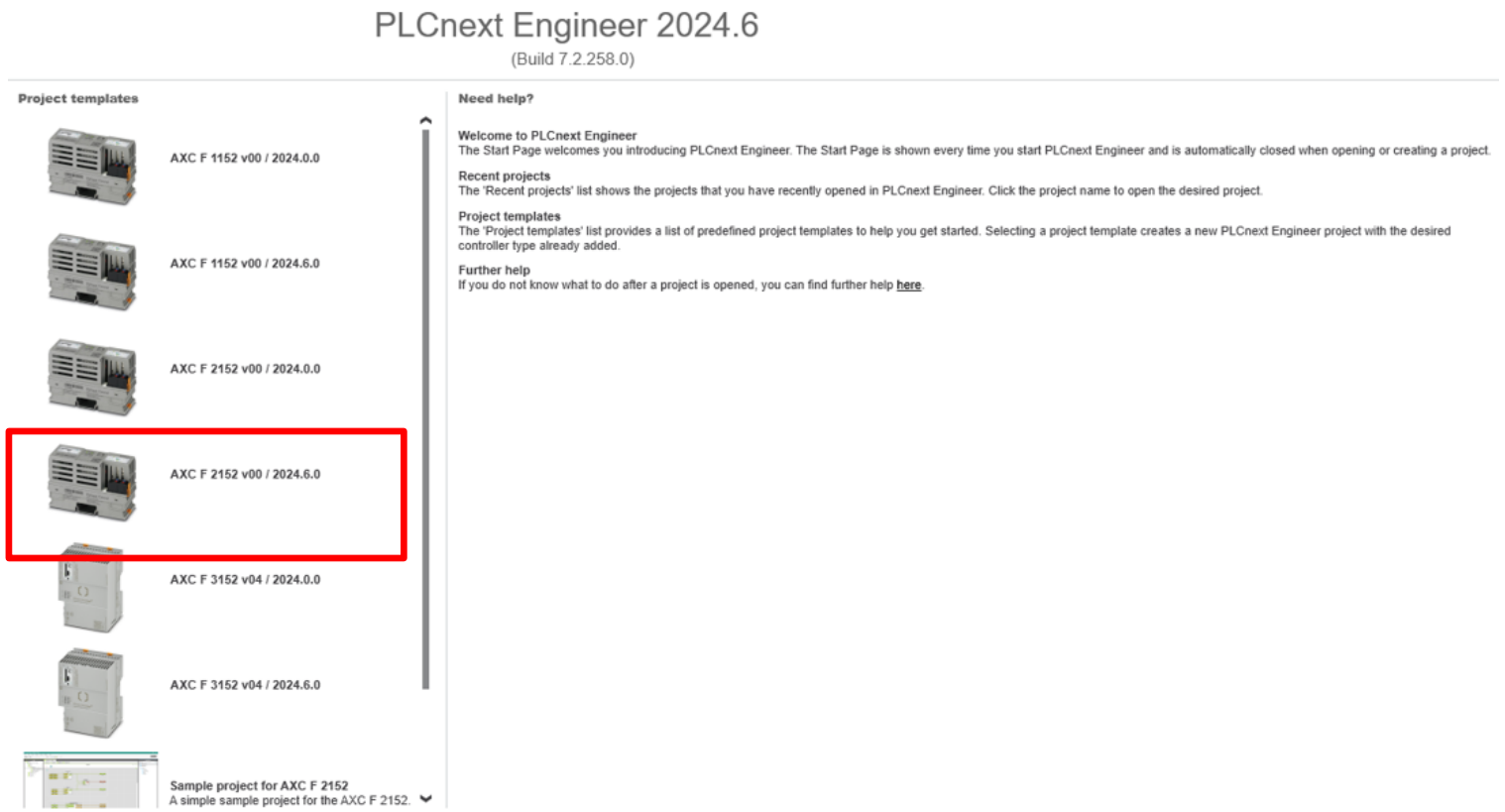






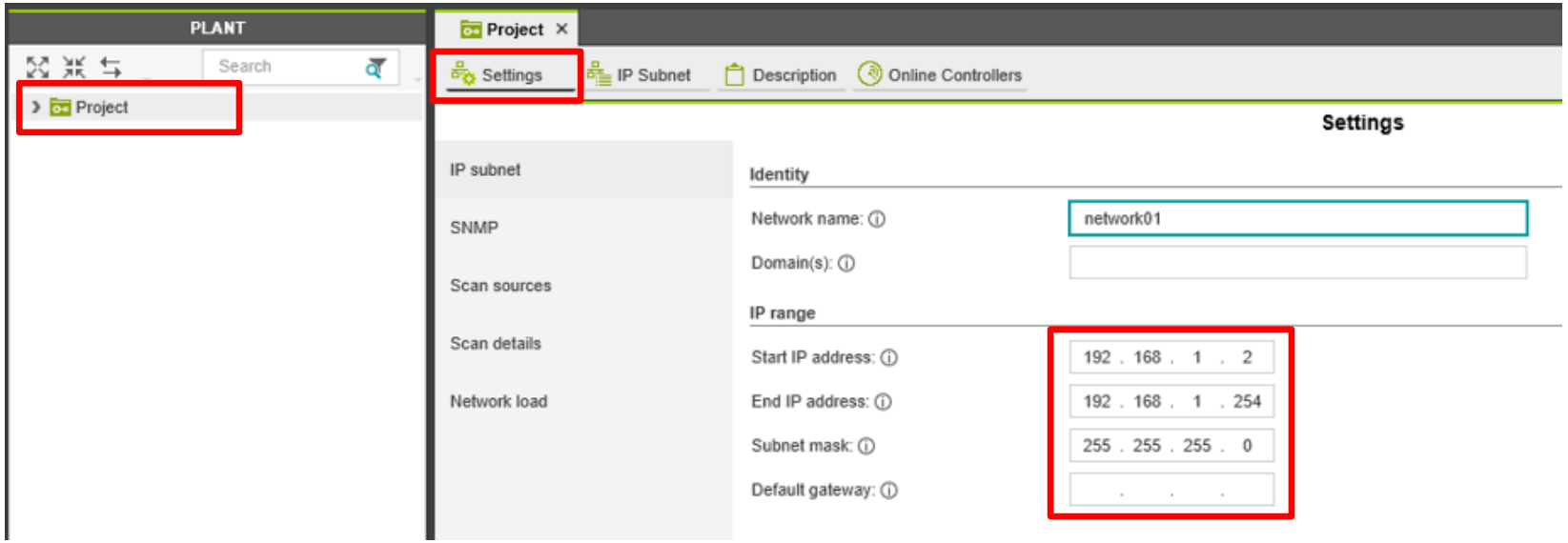
PLCnext Engineerでの段取り

- PLCnext Engineerを開きます。
  ここではFW2024.6.0に合わせて
  ”Project templates”より
  “AXC F2152 v00/2024.6.0”を
  クリックします。



PLCnext Engineerでの段取り

- “PLANT”->”Project”をダブルクリックします。
  “Settings”タブで念のため、”IP range”の
  ”Start IP address”および
  “End IP address”が添付のように
  なっているのか確認してください。
  今回はAXC F 2152のIPアドレスを
  192.168.1.10に設定します。
  添付の範囲になっていない場合は、
  同じアドレス範囲で変更してください。



PLCnext Engineerでの段取り

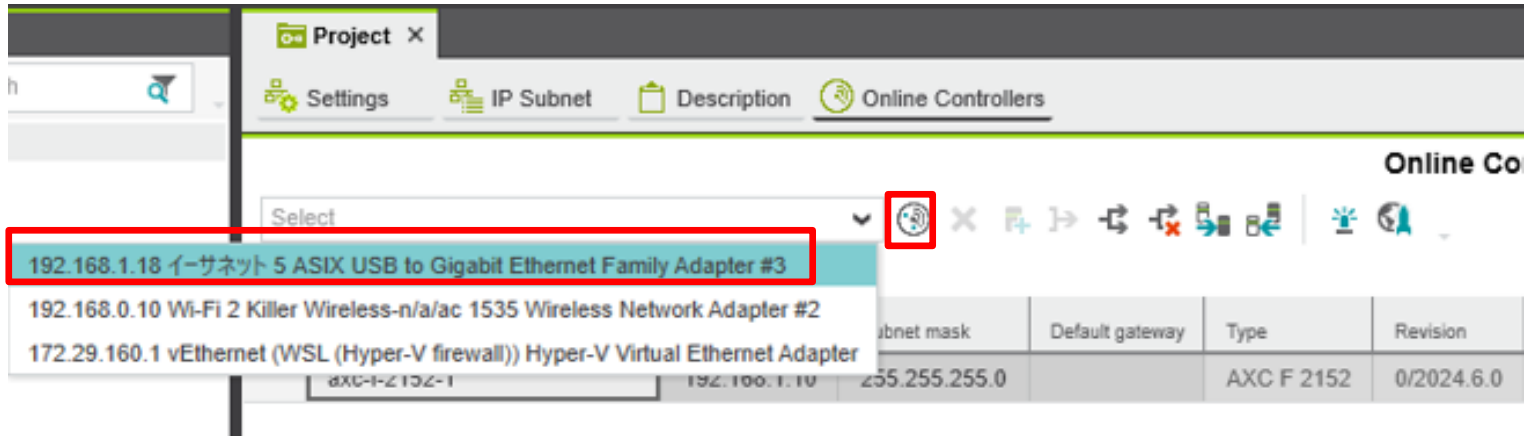
- “Online Controllers”タブをクリックします。

  PCとAXC F2152とをネットワーク接続しているNICを選択します。

  この時、PCのIPアドレスは192.168.1.xのネットワークアドレスに変更する必要があります。(今回は192.168.1.18と設定)

  選択後、 をクリックすると同一ネットワーク上の機器、つまりAXC F2152(192.168.1.10)を探索します。

  見つかれば、そのMACアドレスをクリックします。



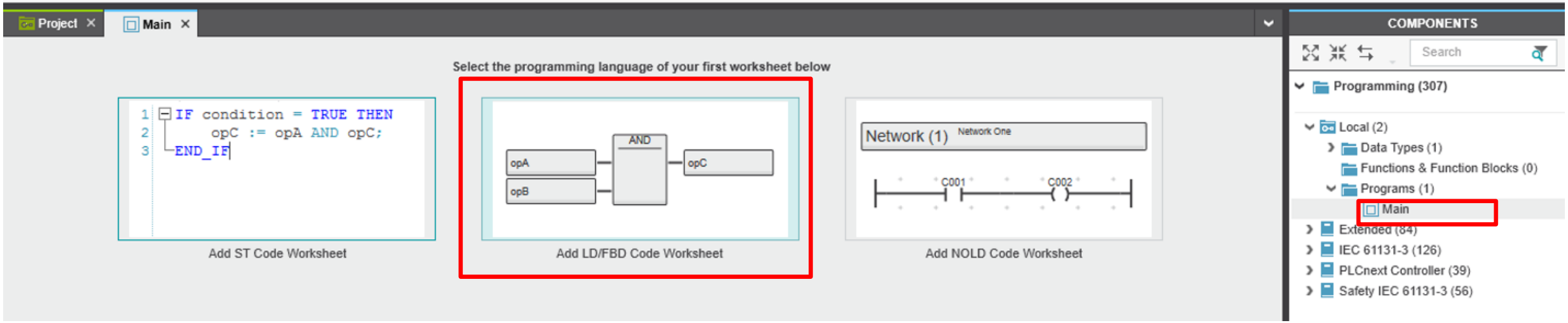
PLCnext Engineerでの段取り

- 同一ネットワーク上に機器が見つかれば、そのMACアドレスを選択します。
  “IP address”, “Subnet mask”, “Default gateway”が設定されます。
  問題なければ、Statusがチェックマークになります。

Name of station (Project)	IP address	Subnet mask	Default gateway	Type	Revision	Status	Name of station (Online)	IP address	Subnet mask	Default gateway	Type	Revision
axc-f-2152-1	192.168.1.10	255.255.255.0	192.168.1.254	AXC F 2152	0/2024.6.0	✓	axc-f-2152-1	192.168.1.10	255.255.255.0	192.168.1.254	AXC F 2152	2/2024.6.0

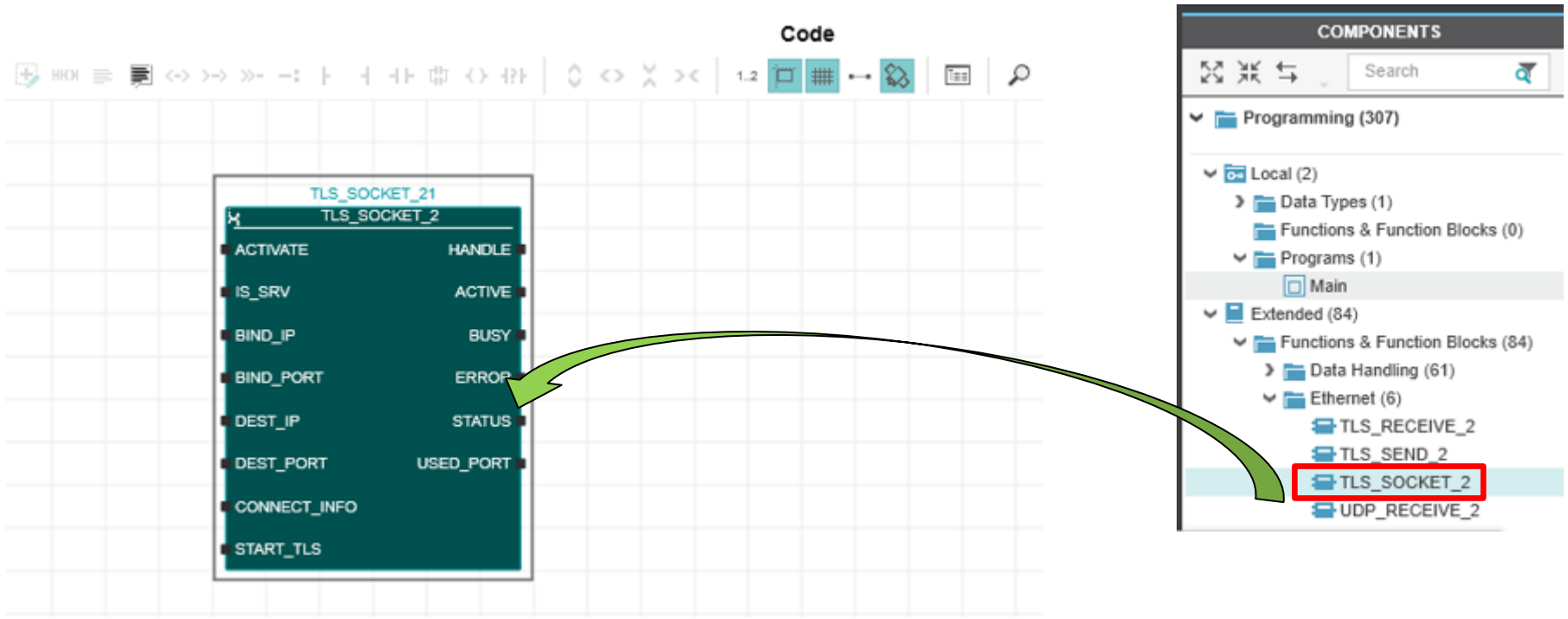
PLCnext Engineerでの段取り

- “COMPONENTS”->”Programming”->”Local”->”Programs”->”Main”をクリックします。
  真ん中のワークシートがプログラム言語の選択画面になります。
  今回は”Add LD/FBD Code Worksheet”を選択します。



PLCnext Engineerでの段取り

- “COMPONENTS”->”Programming”->”Extended”->”Functions & Function Blocks”->”Ethernet”にある”TLS_SOCKET_2”を真ん中のワークシートへドラッグ&ドロップします。



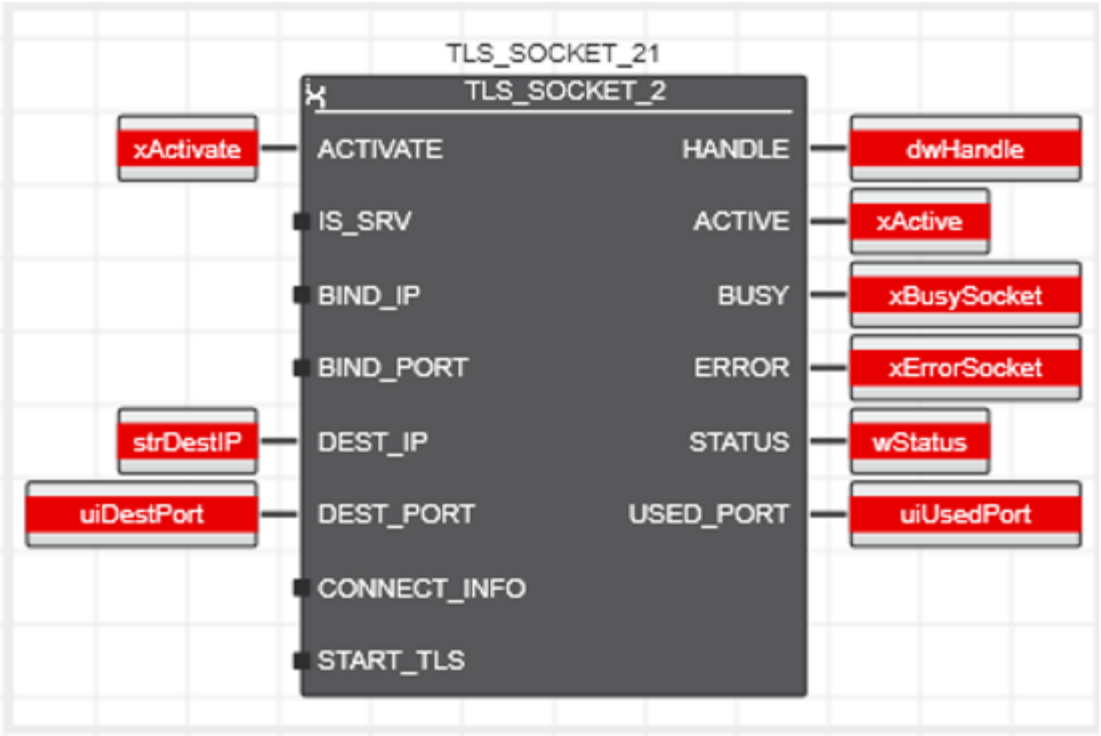
PLCnext Engineerでの段取り

- 下記を参考に各入出力に変数名を定義します。(変数名は任意です。下記は本デモとしての変数名になります。)
  型などが未定義なので、赤色で変数名が表示されます。

xActivate

strDestIP

uiDestPort



dwHandle

xActive

xBusySocket

xErrorSocket

wStatus

uiUsedPort

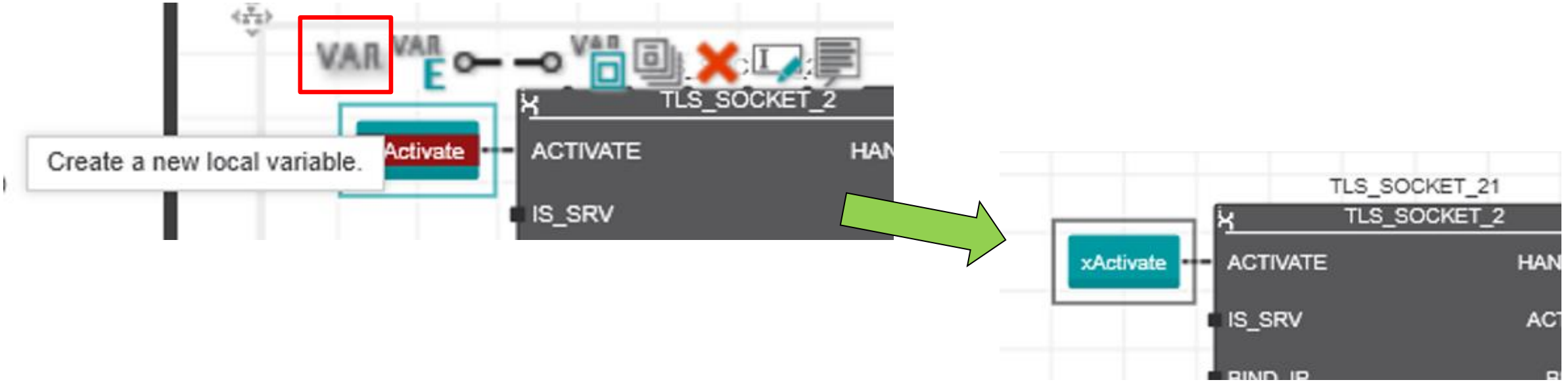
PLCnext Engineerでの段取り

- 赤色の未定義の変数をクリックします。

  すると、上部にアイコンが表示されます。

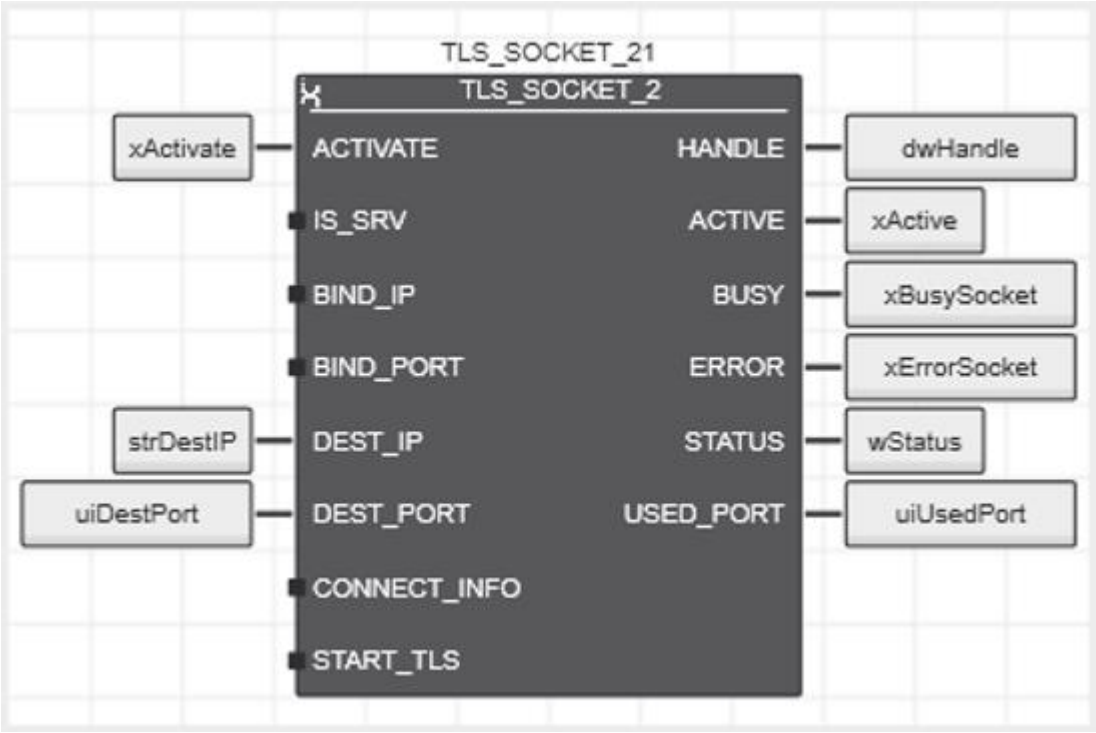
  今回使用する変数は全て、内部で使用するもの(I/Oなどハード的に外部と入出力するものではない)ですので、

  “VAR”をクリックし、ローカル変数として定義します。定義が出来ると赤色が消えます。



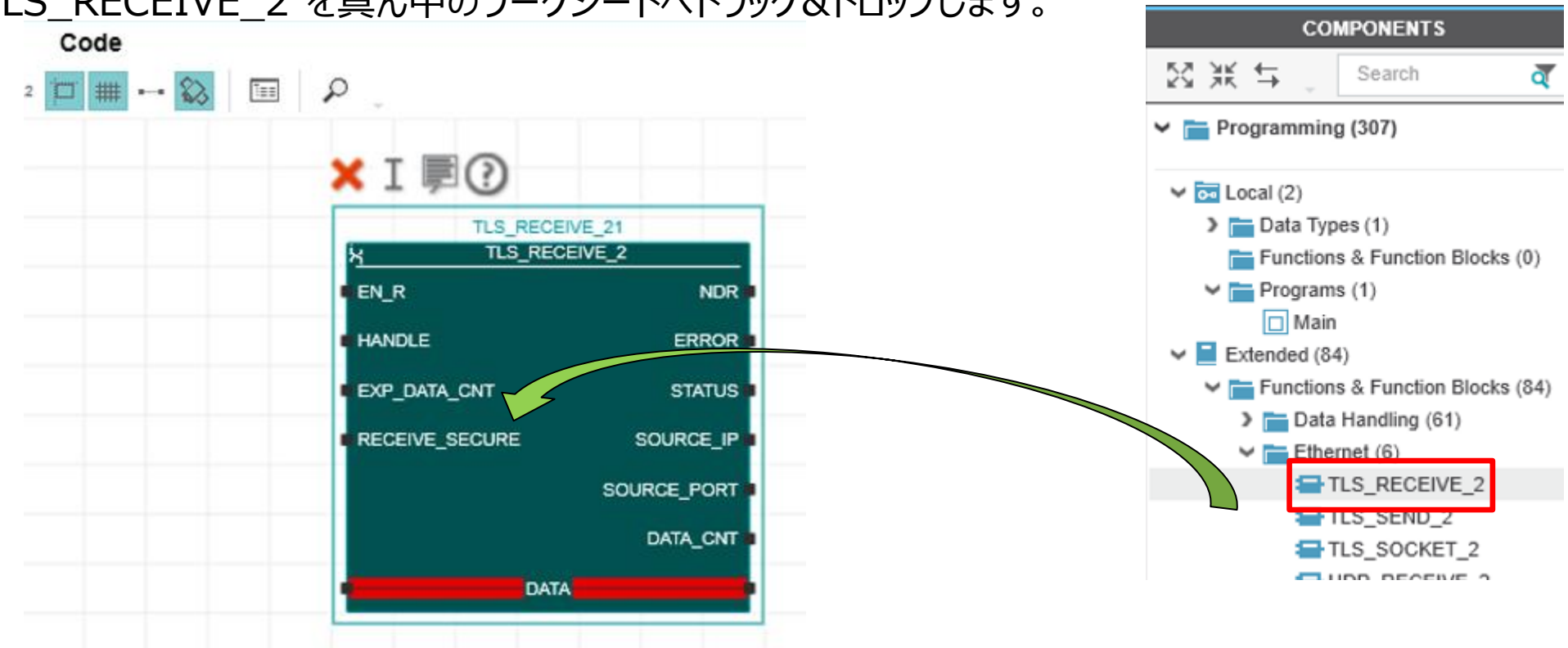
PLCnext Engineerでの段取り

- 他の変数に対しても同様の操作を行い、未定義の変数名に対して、定義を行います。



PLCnext Engineerでの段取り

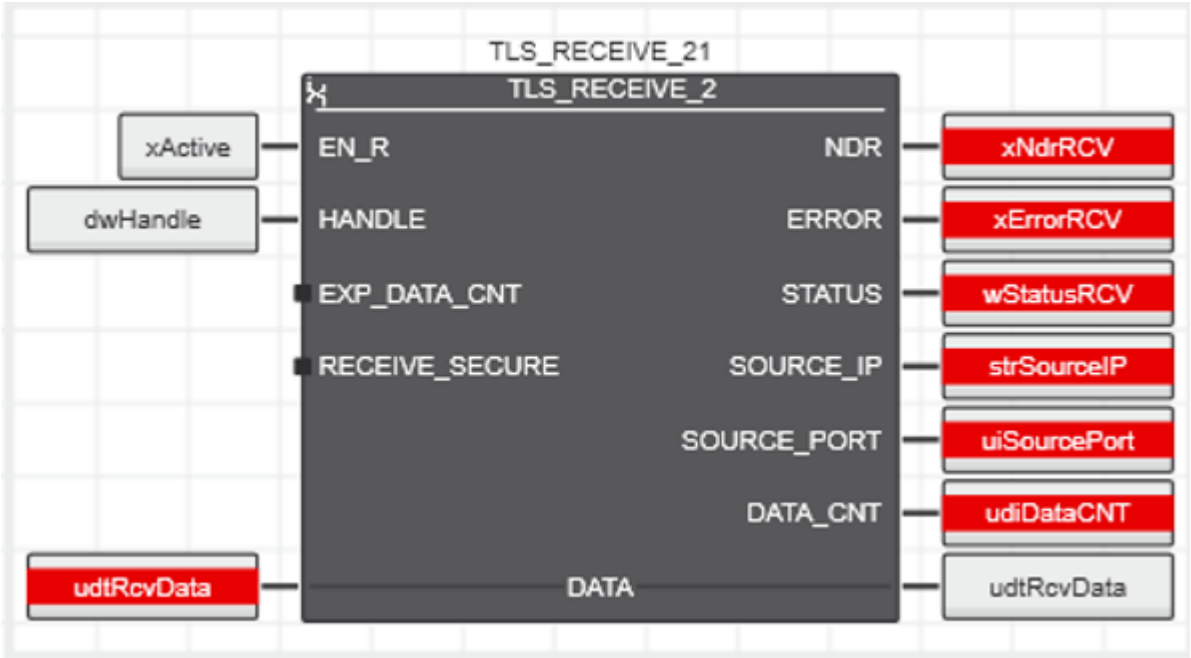
- “COMPONENTS”->”Programming”->”Extended”->”Functions & Function Blocks”->”Ethernet”にある”TLS_RECEIVE_2”を真ん中のワークシートへドラッグ&ドロップします。



PLCnext Engineerでの段取り

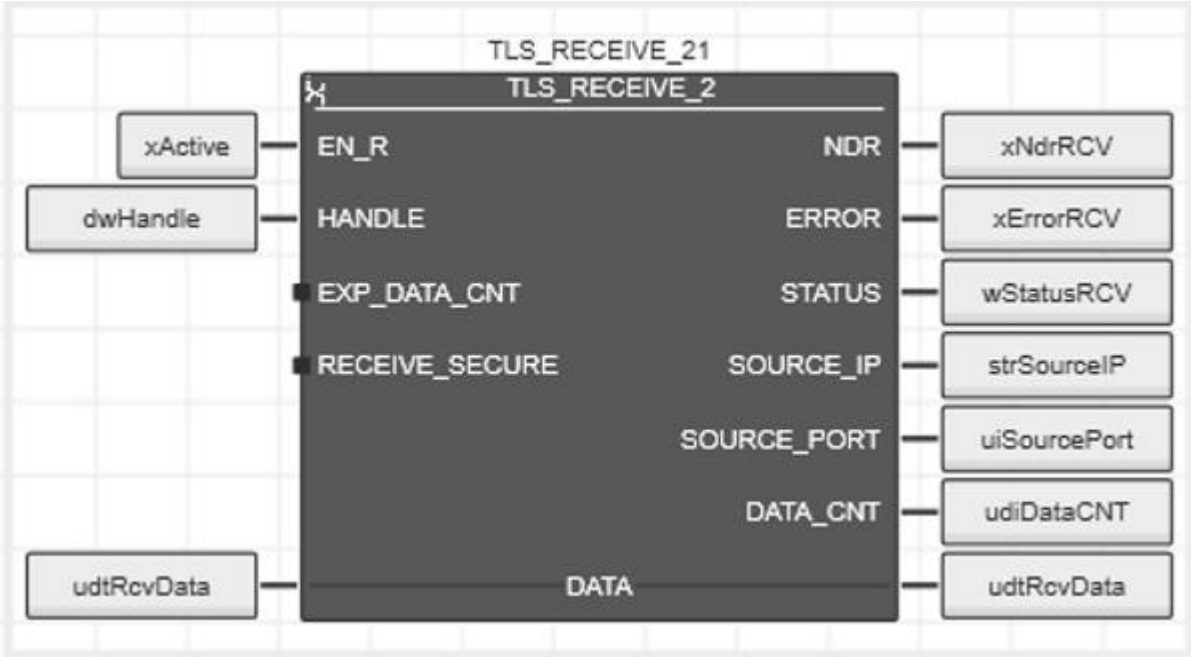
- 下記を参考に各入出力に変数名を定義します。(変数名は任意です。下記は本デモとしての変数名になります。)
  型などが未定義なので、赤色で変数名が表示されます。

“EN_R”,”HANDLE”に対してはそれぞれ”TLS_SOCKET_21”の”xActive”, “dwHandle”とします。



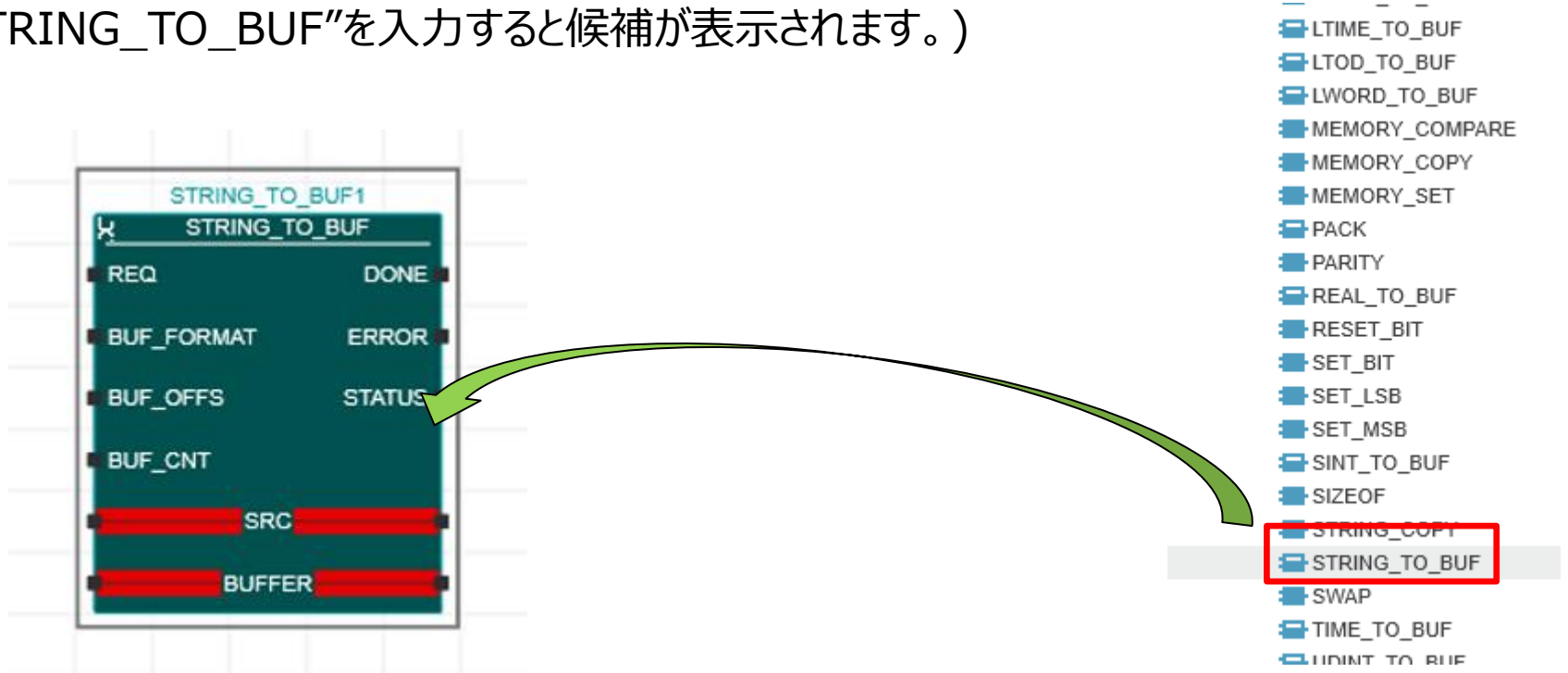
PLCnext Engineerでの段取り

- 未定義の変数名に対して、”TLS_SOCKET_21”と同様の操作でローカル変数定義を行います。



PLCnext Engineerでの段取り

- “COMPONENTS”->”Programming”->”Extended”->”Functions & Function Blocks”->
  “Data Handling”にある”STRING_TO_BUF”を真ん中のワークシートへドラッグ&ドロップします。
  (検索バーに”STRING_TO_BUF”を入力すると候補が表示されます。)



PLCnext Engineerでの段取り

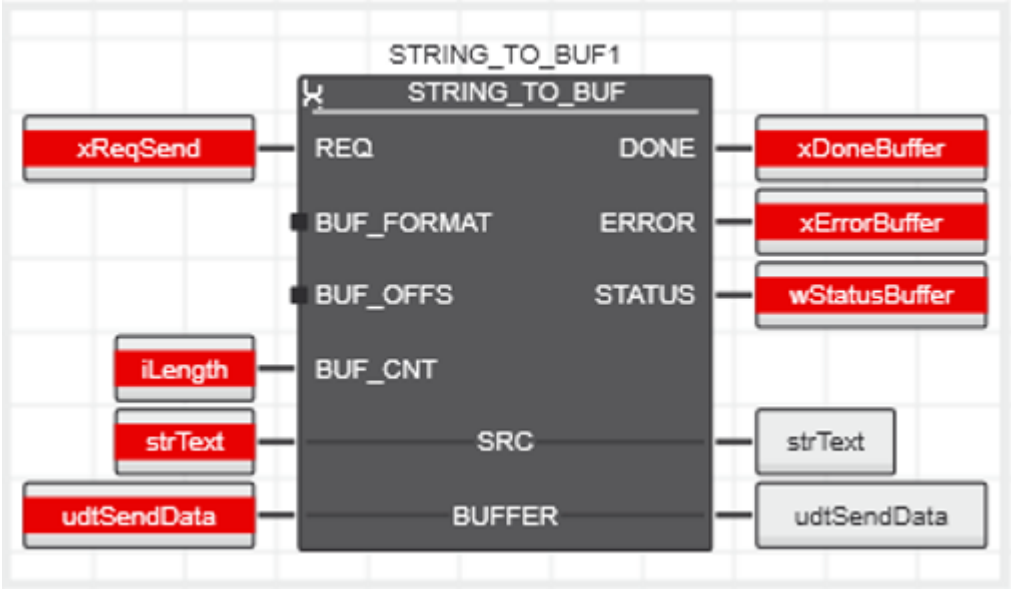
- 下記を参考に各入出力に変数名を定義します。(変数名は任意です。下記は本デモとしての変数名になります。)
  型などが未定義なので、赤色で変数名が表示されます。

xReqSend

iLength

strText

udtSendData



xDoneBuffer

xErrorBuffer

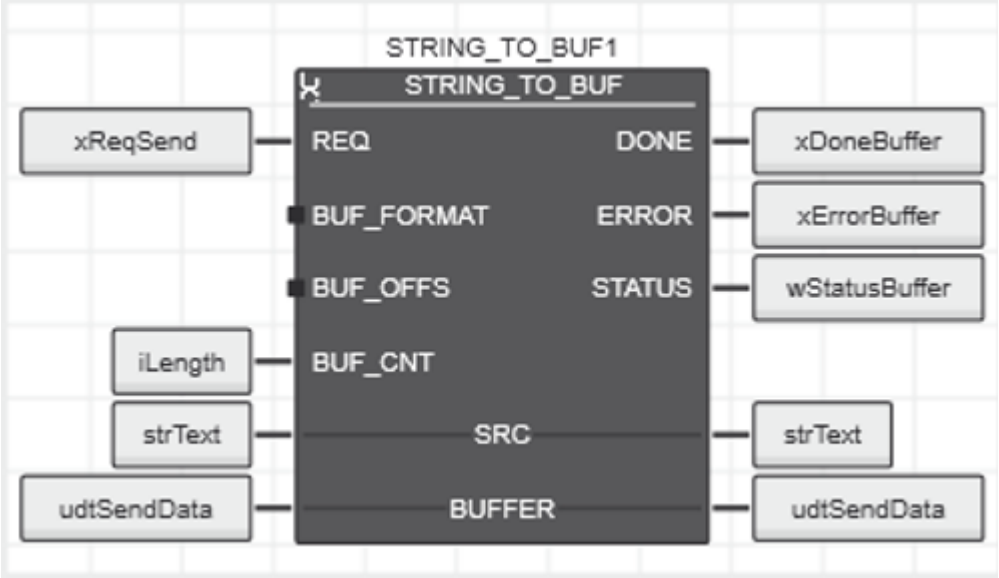
wStatusBuffer

strText

udtSendData

PLCnext Engineerでの段取り

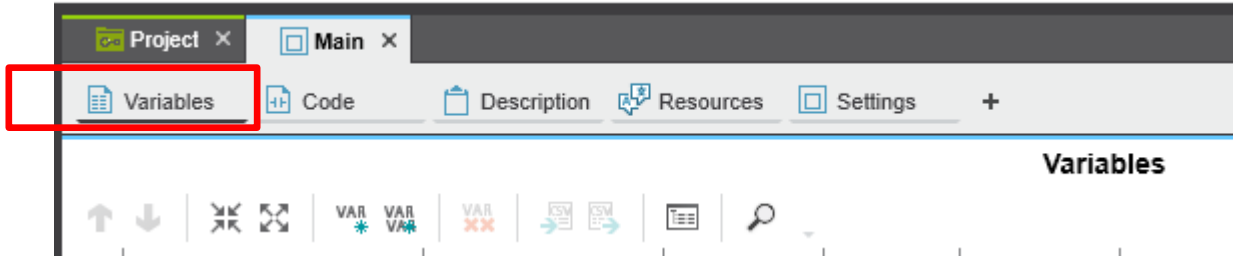
- 未定義の変数名に対して、”TLS_SOCKET_21”らと同様の操作でローカル変数定義を行います。

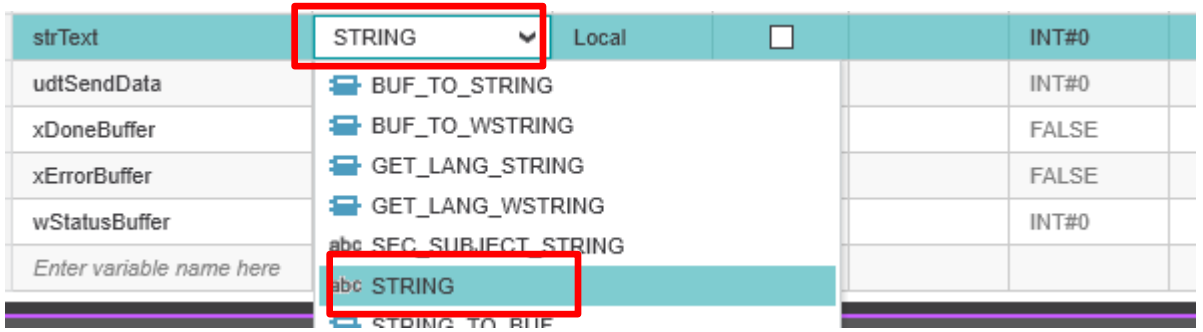


PLCnext Engineerでの段取り

- “Main”のワークシートで”Variables”タブを選択します。

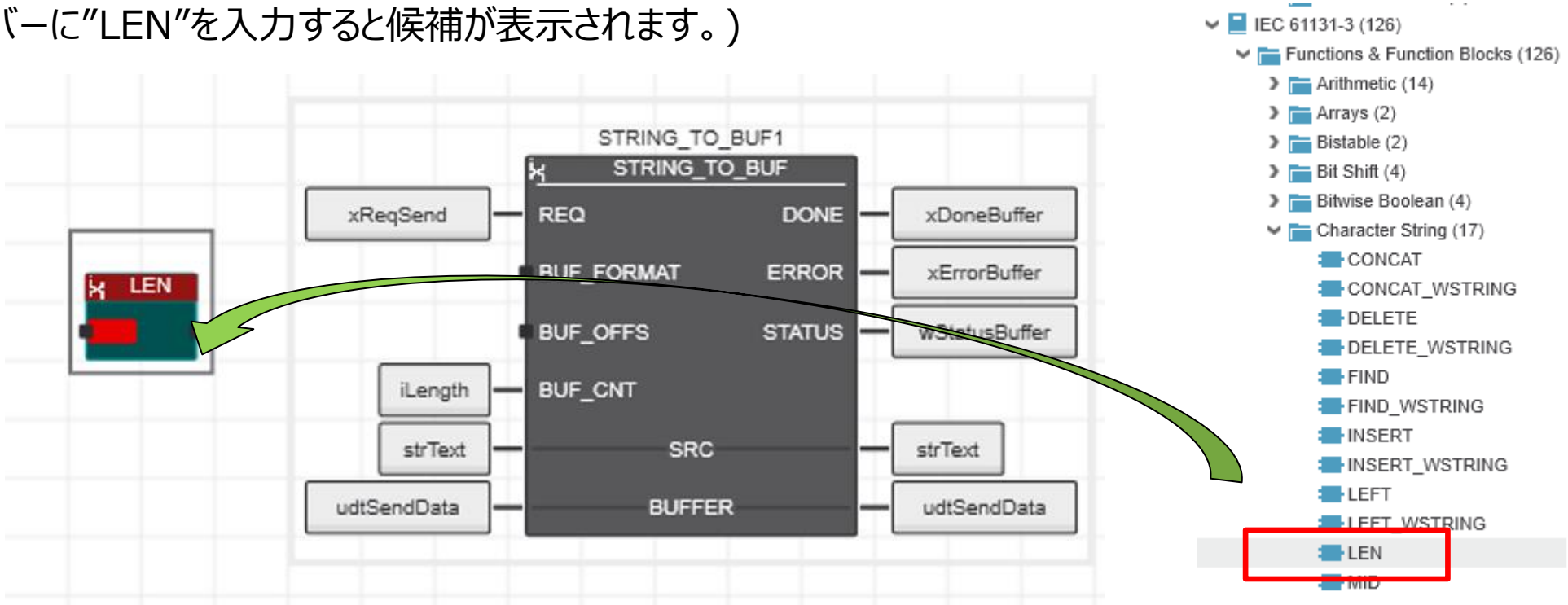
  “strText”変数名の“Type”欄を”STRING”型へ変更します。





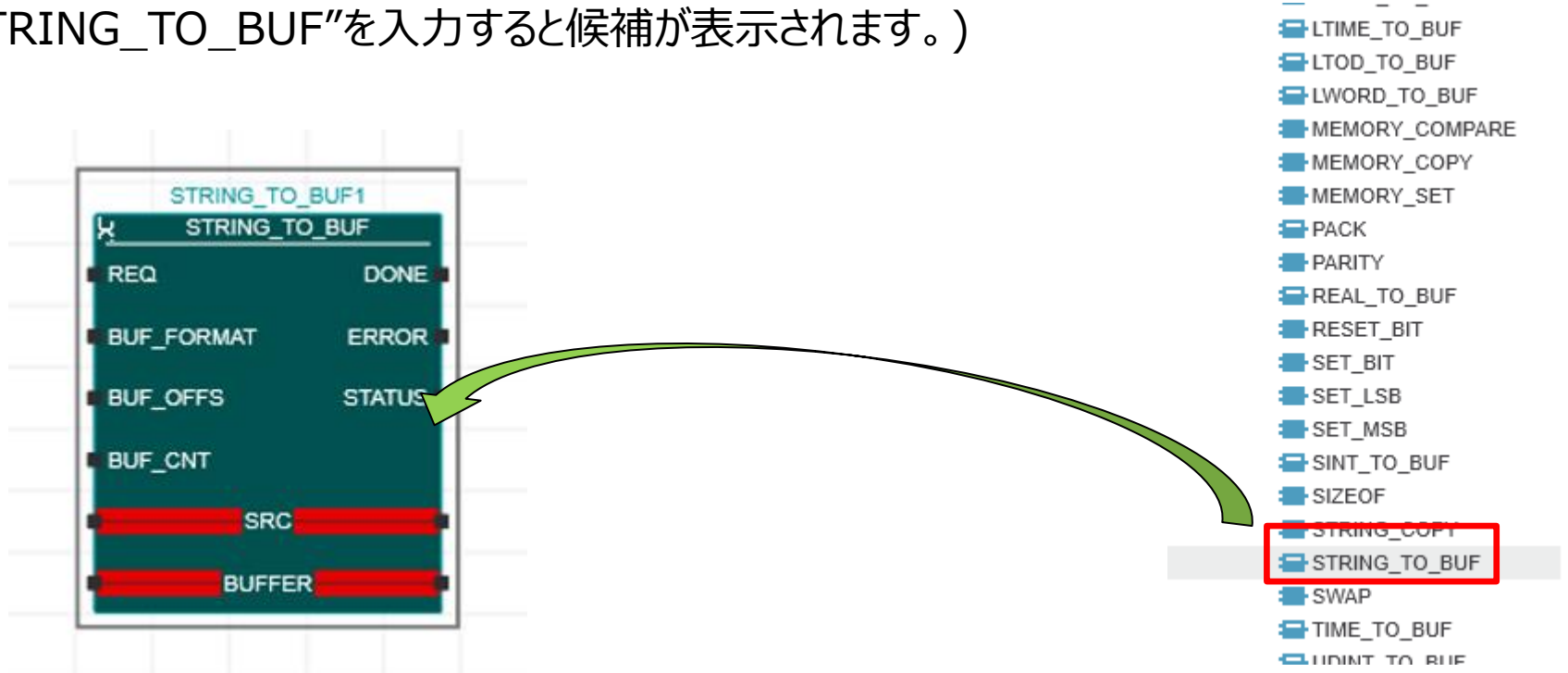
PLCnext Engineerでの段取り

- “COMPONENTS”->”Programming”->”IEC 61131-3”->”Functions & Function Blocks”->“Character String”にある”LEN”を真ん中のワークシートへドラッグ&ドロップします。
  (検索バーに”LEN”を入力すると候補が表示されます。)



PLCnext Engineerでの段取り

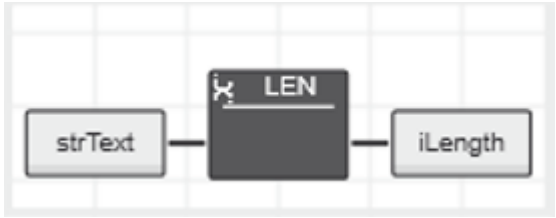
- “COMPONENTS”->”Programming”->”Extended”->”Functions & Function Blocks”->
  “Data Handling”にある”STRING_TO_BUF”を真ん中のワークシートへドラッグ&ドロップします。
  (検索バーに”STRING_TO_BUF”を入力すると候補が表示されます。)



PLCnext Engineerでの段取り

- 下記を参考に各入出力に変数名を定義します。(変数名は任意です。下記は本デモとしての変数名になります。)
  “strText”, “iLength”は、”STRING_TO_BUF1”の”strText”, “iLength”になります。

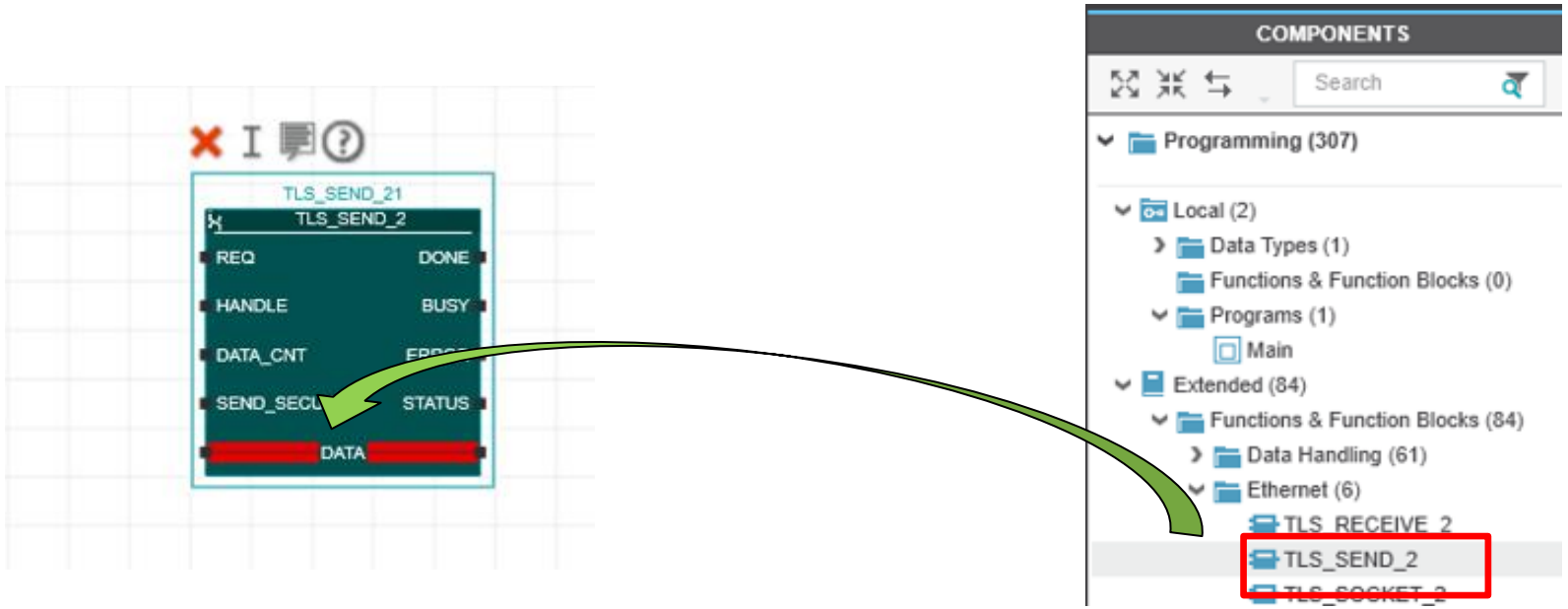
strText



iLength

PLCnext Engineerでの段取り

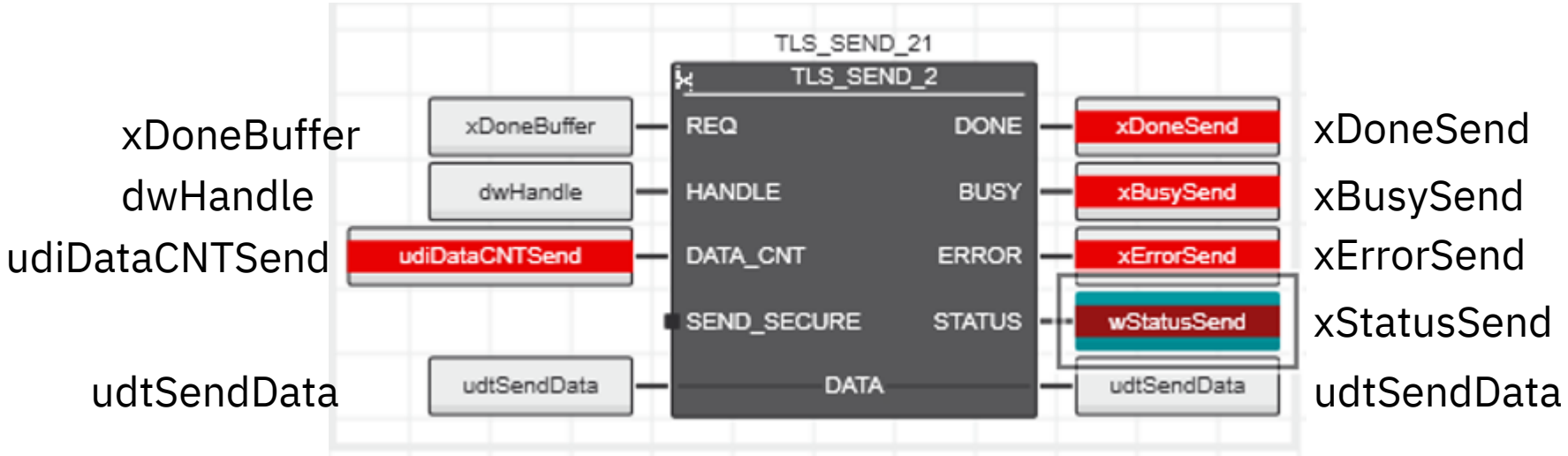
- “COMPONENTS”->”Programming”->”Extended”->”Functions & Function Blocks”->”Ethernet”にある”TLS_SEND_2”を真ん中のワークシートへドラッグ&ドロップします。



PLCnext Engineerでの段取り

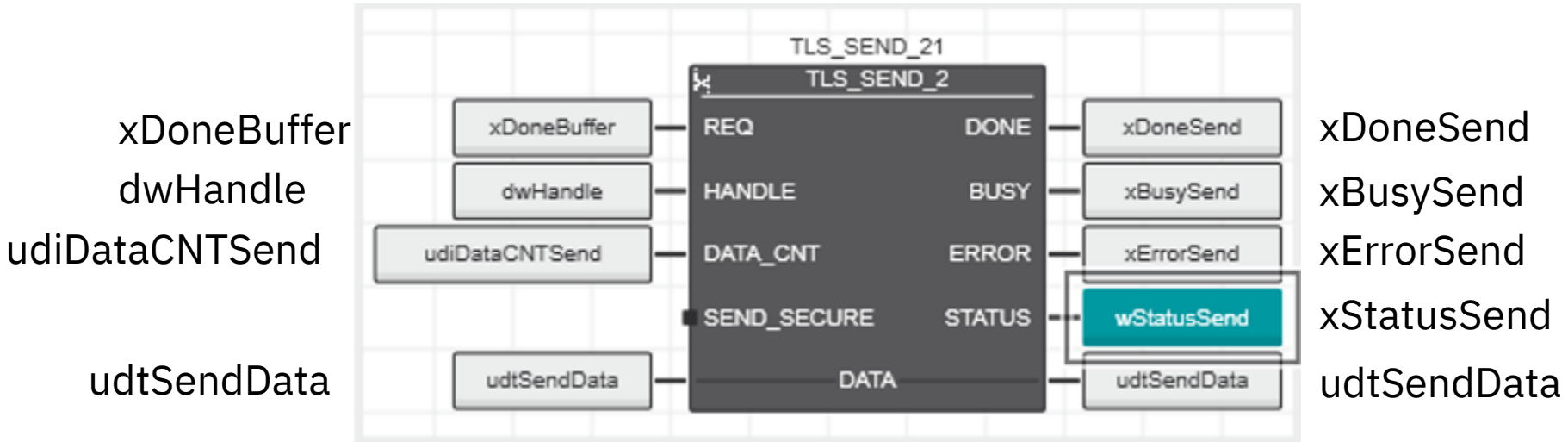
- 下記を参考に各入出力に変数名を定義します。(変数名は任意です。下記は本デモとしての変数名になります。)
  型などが未定義なので、赤色で変数名が表示されます。

“REQ”,”HANDLE”はそれぞれ”STRING_TO_BUF1”の”xDoneBuffer”、
”TLS_SOCKET_21”の “dwHandle”とします。



PLCnext Engineerでの段取り

- 未定義の変数名に対して、”TLS_SOCKET_21”らと同様の操作でローカル変数定義を行います。



PLCnext Engineerでの段取り

- “Main”のワークシートで”Variables”タブを選択します。

  “strDestIP”変数の“Init”欄を接続する相手先IPアドレス(今回はPCの”192.168.1.18”)へ変更します。

  “uiDestPort”変数の“Init”欄を接続する相手先ポート番号(今回は”12345”)へ変更します。

  “strText”変数の“Init”欄を送信するメッセージ(今回は”Hello, World.”)へ変更します。

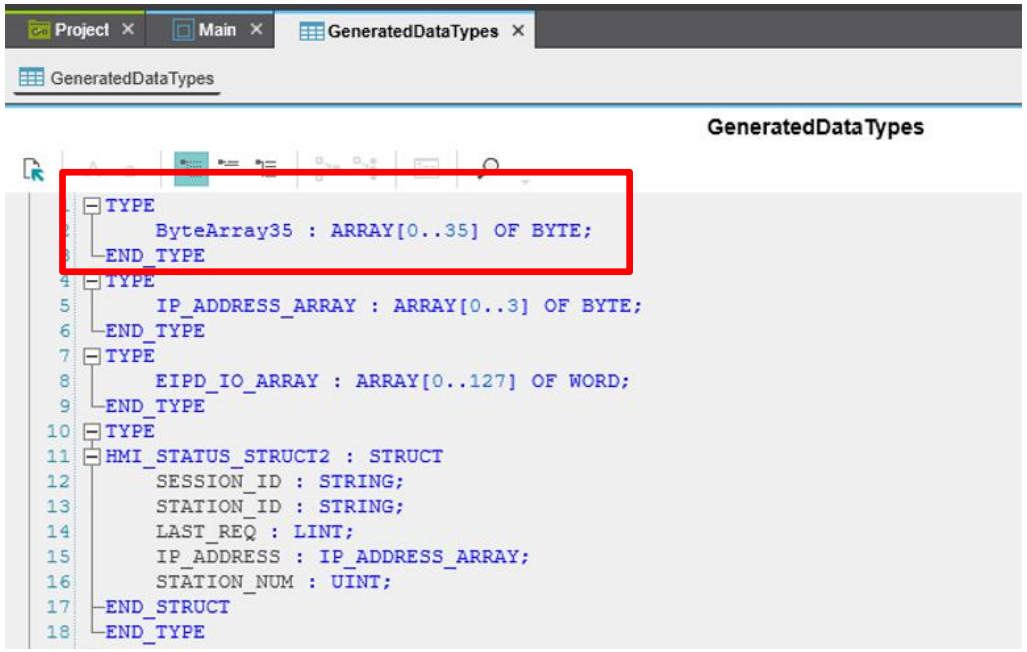
Name	Type	Usage	Translate	Comment	Init
efault					
TLS_SOCKET_21	TLS_SOCKET_2	Local	☐		
xActivate	BOOL	Local	☐		FALSE
strDestIP	STRING	Local	☐		92.168.1.18'
uiDestPort	UINT	Local	☐		UINT#12345
strText	STRING	Local	☐		ello, World.'

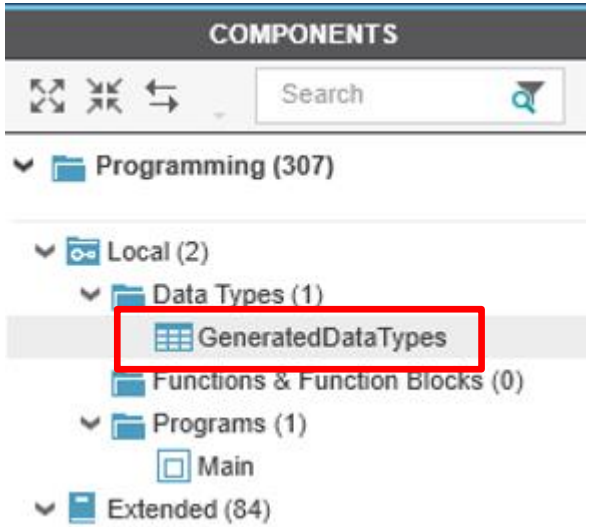
PLCnext Engineerでの段取り

- “COMPONENTS”->”Programming”->“Local”->”Data Types”->
  ”GeneratedDataTypes”を選択します。
  すると型や構造体が定義されたワークシート画面が真ん中に表示されます。

今回は”ByteArray35”という型を使用します。

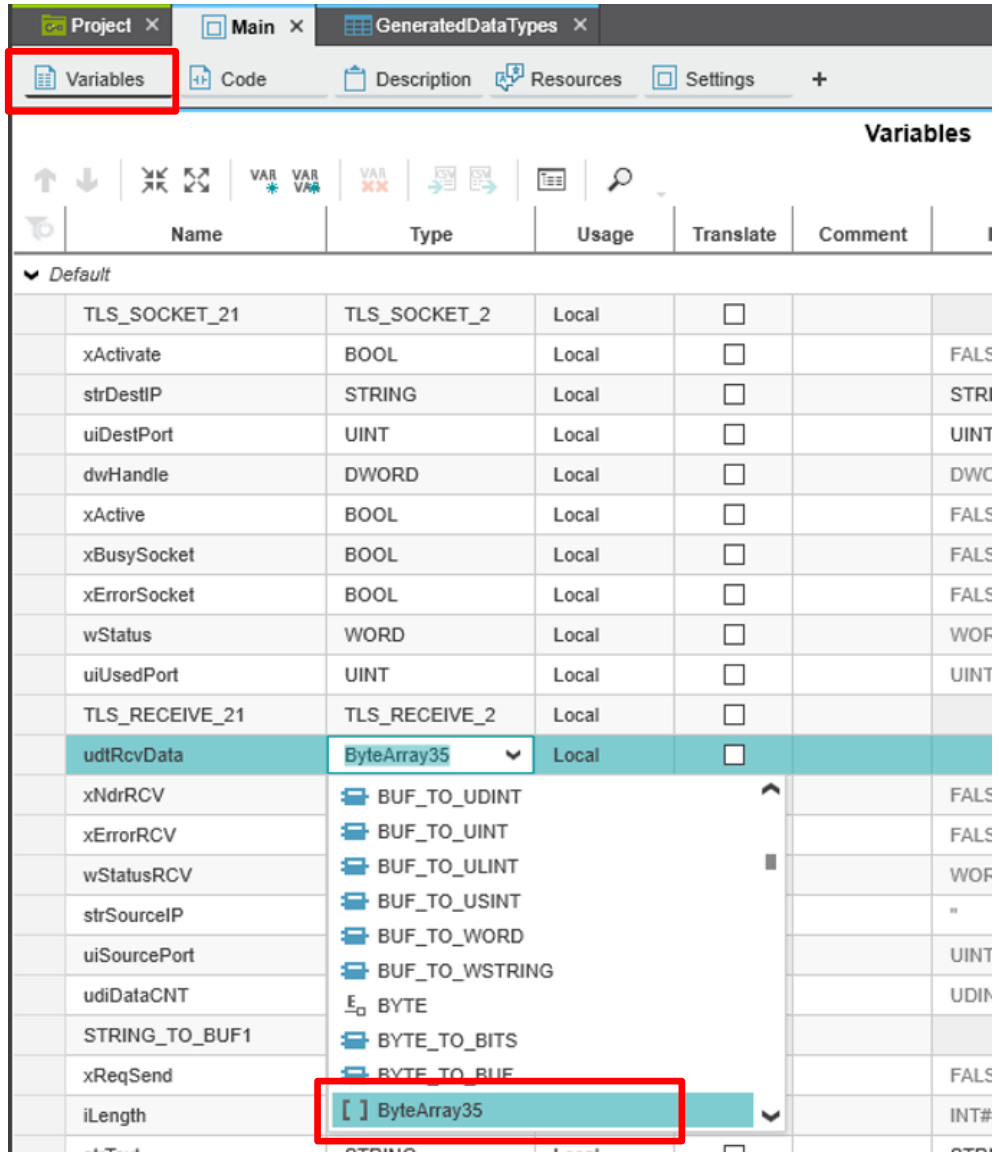
定義されていない場合は下記の赤枠に沿って記述してください。





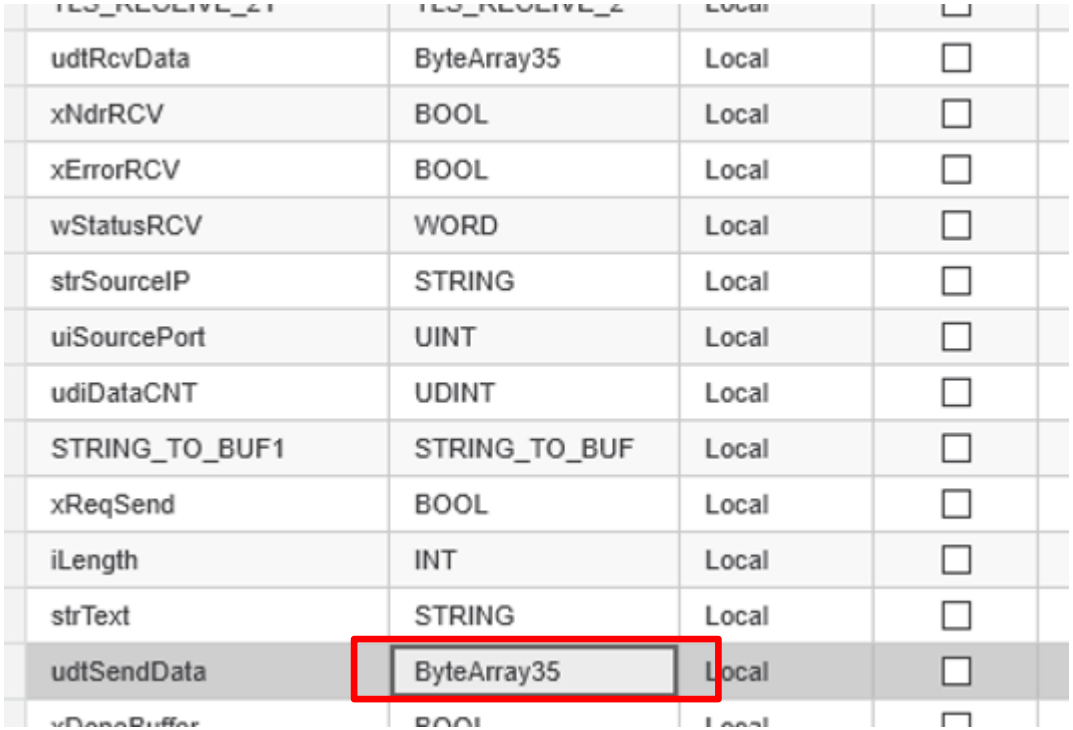
PLCnext Engineerでの段取り

- “Main”->”Variables”タブを選択します。
  変数名”udtRcvData”の”Type”欄を選択し、
  “ByteArray35”を選択します。
  0-35byte分、つまり半角36文字分のデータが扱えるようになります。



PLCnext Engineerでの段取り

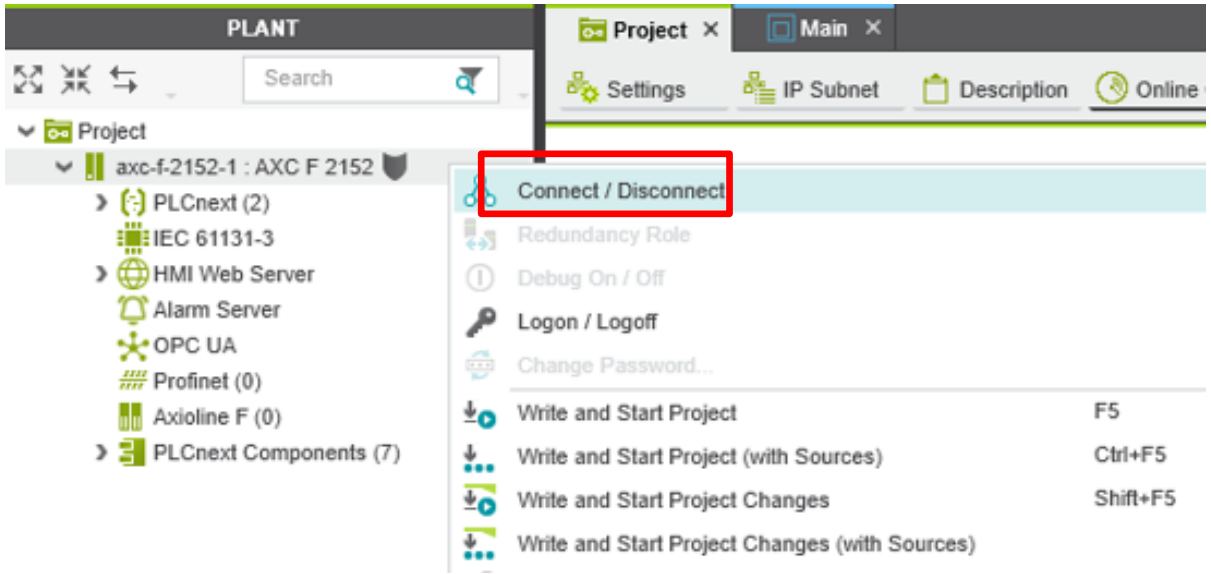
- 同様に変数名”udtSendData”の”Type”欄を選択し、
  “ByteArray35”を選択します。
  0-35byte分、つまり半角36文字分のデータが扱えるようになります。



udtRcvData	ByteArray35	Local	☐
xNdrRCV	BOOL	Local	☐
xErrorRCV	BOOL	Local	☐
wStatusRCV	WORD	Local	☐
strSourceIP	STRING	Local	☐
uiSourcePort	UINT	Local	☐
udiDataCNT	UDINT	Local	☐
STRING_TO_BUF1	STRING_TO_BUF	Local	☐
xReqSend	BOOL	Local	☐
iLength	INT	Local	☐
strText	STRING	Local	☐
udtSendData	ByteArray35	Local	☐

PLCnext Engineerでの段取り

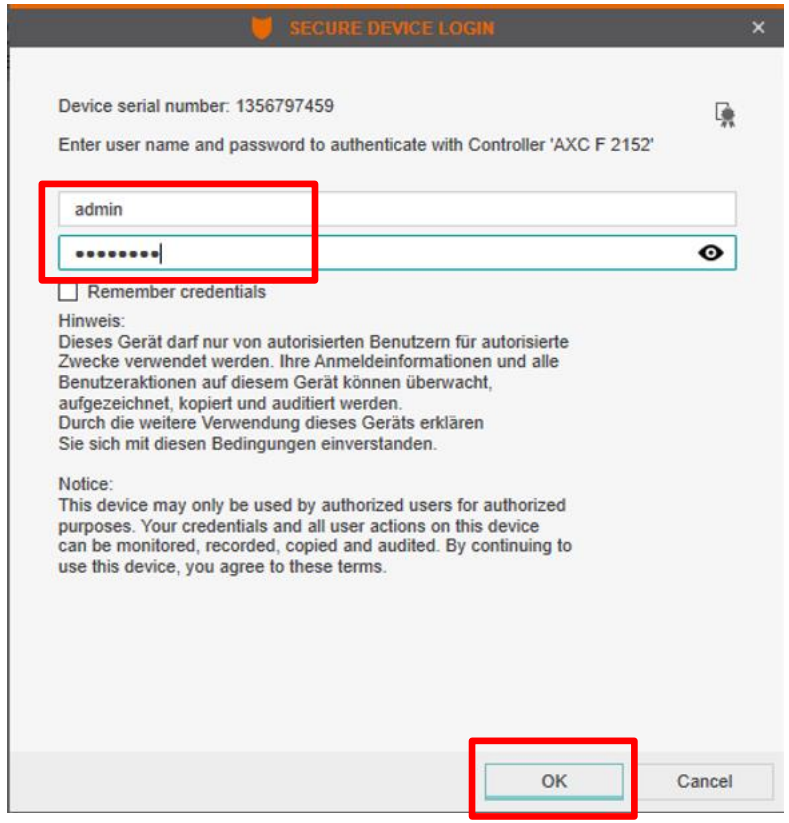
- “PLANT”->”Project”->”axc-f-2152-1:AXC F 2152”を右クリックします。
  “Connect/Disconnect”をクリックします。



PLCnext Engineerでの段取り

- ログイン画面が表示されます。

  初期設定の場合は上段に”admin”、下段のパスコードにはAXC F 2152本体に印字されているPasswordを入力してください。

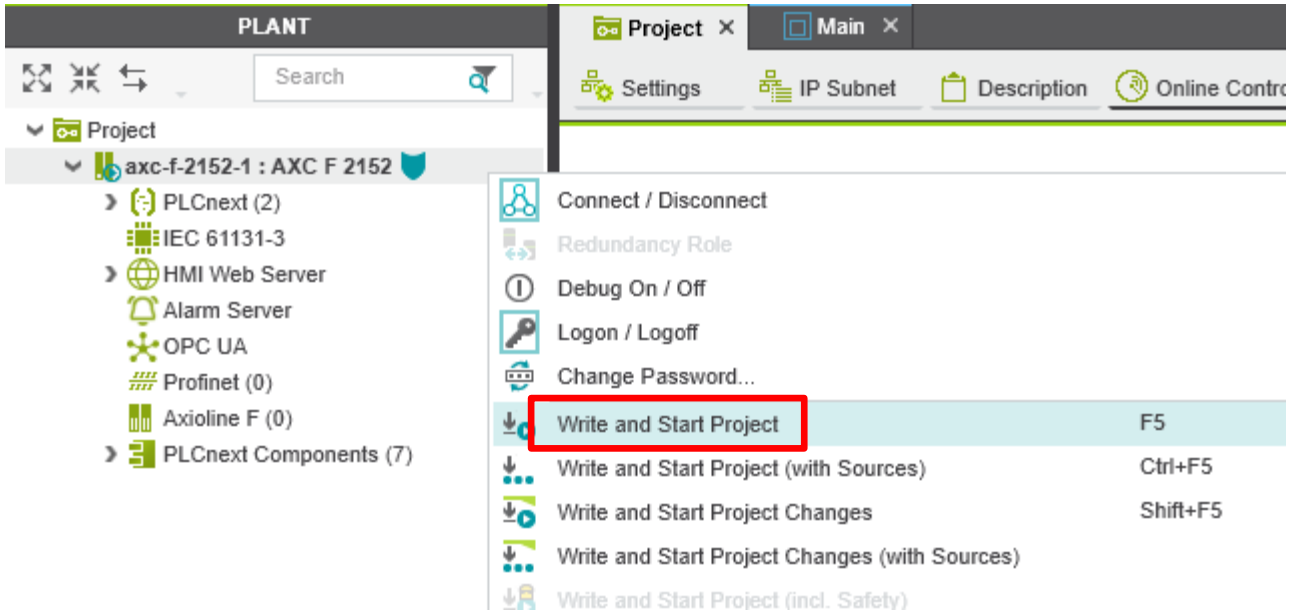


PLCnext Engineerでの段取り

- ログインが完了すると接続完了マークが青色になります。

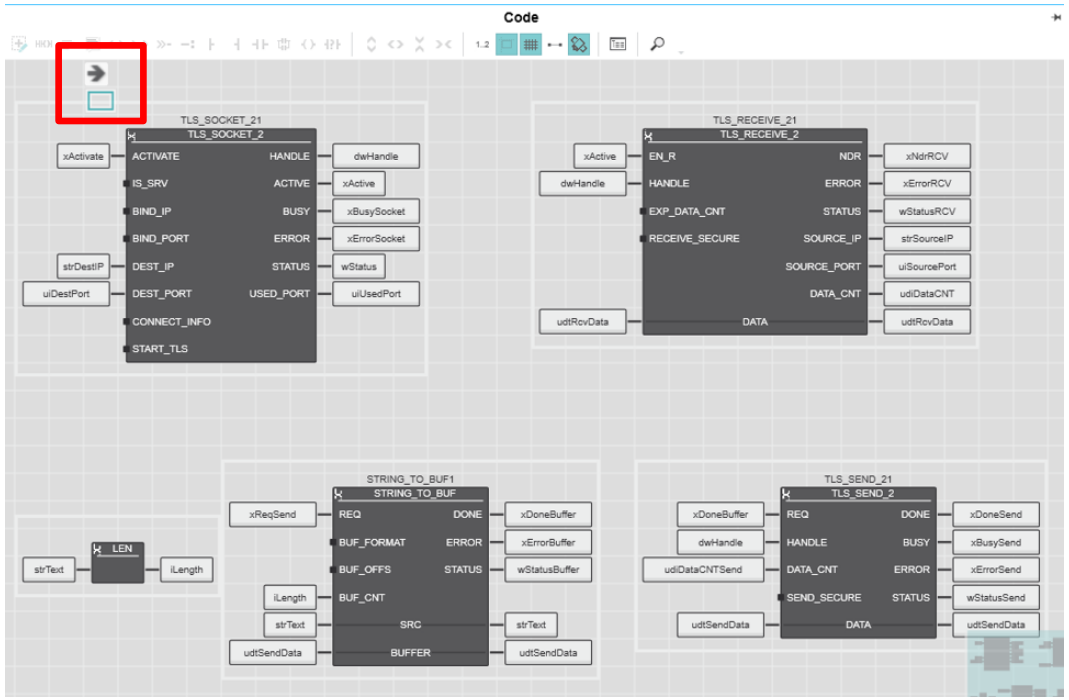
  “PLANT”->”Project”->”axc-f-2152-1:AXC F 2152”を右クリックします。

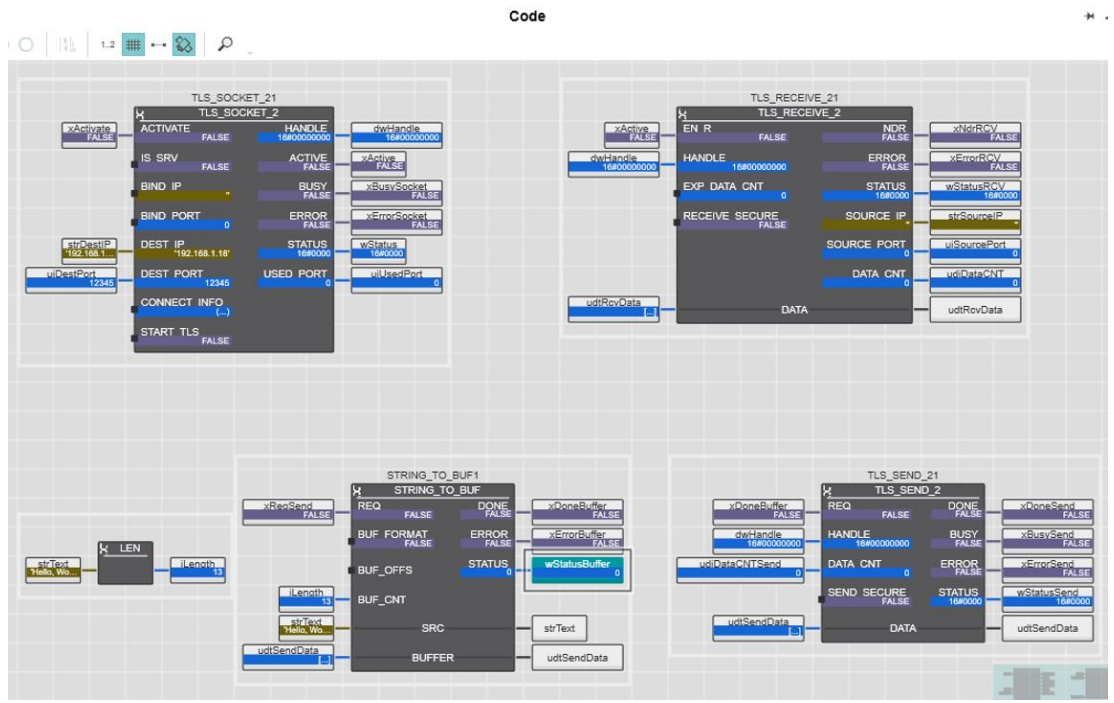
  “Write and Start Project”をクリックしてプロジェクトデータをAXC F 2152へ書き込みます。



PLCnext Engineerでの段取り

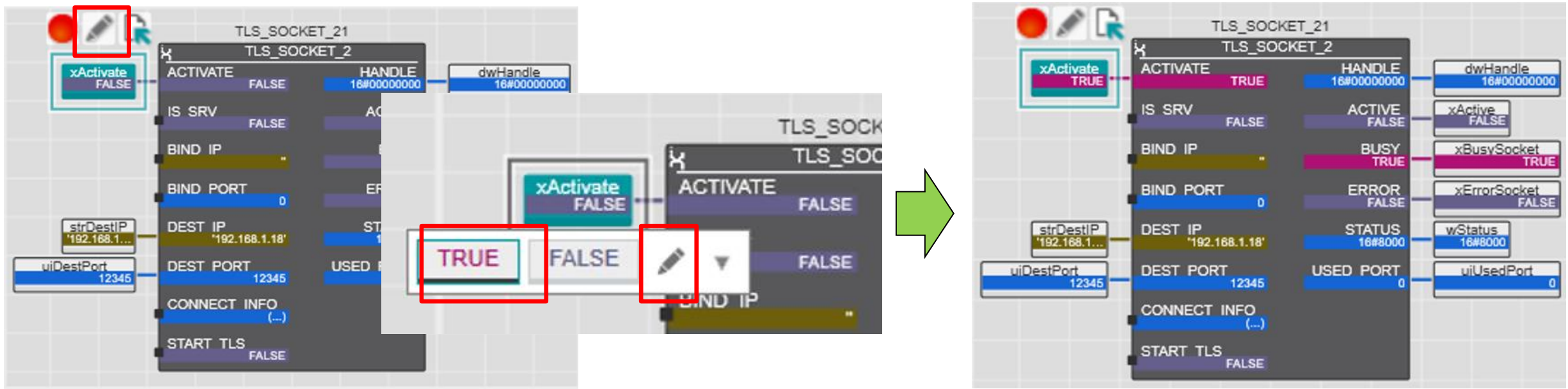
- プロジェクトデータインストール後、実行状態の画面へ遷移します。
  ブランク箇所にカーソルを持っていき”➡”が表示されます。
  クリックすると、現在の変数の状態が分かります。





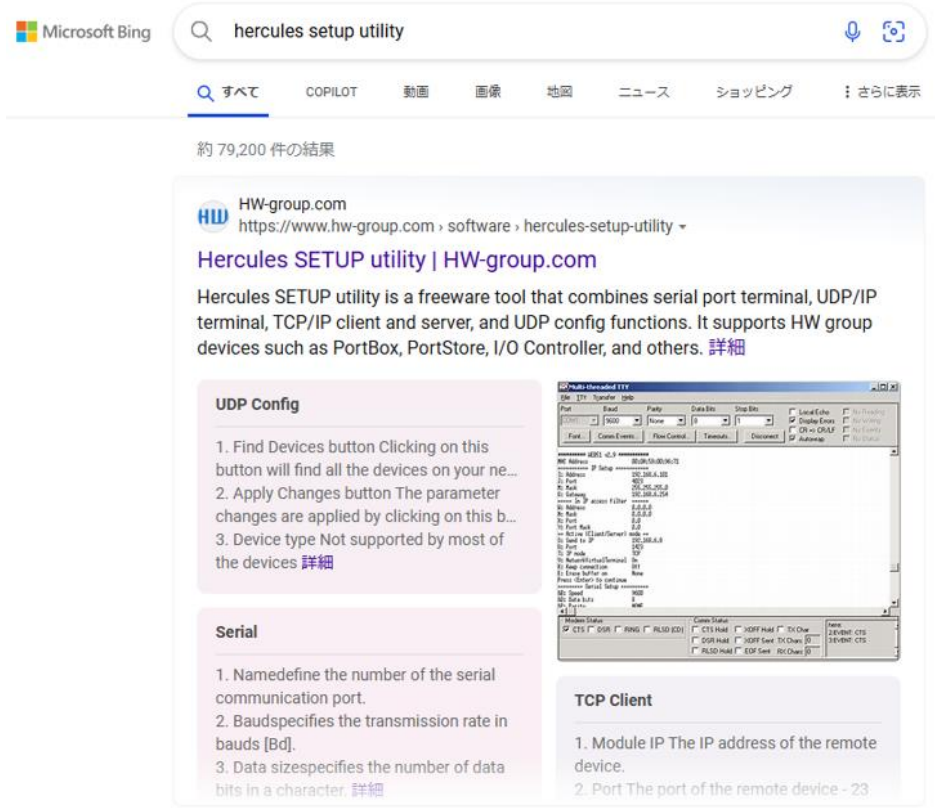
PLCnext Engineerでの段取り

- “TLS_SOCKET_21”の”xActivate”をクリックし、ペンマークをクリックします。
  “TRUE”を選択し、ペンマークを押すことでTRUEの値に書き込みます。
  TCP ClientとしてTCP Serverへソケット通信を実行するようになります。



Hercules SETUP utility側での段取り

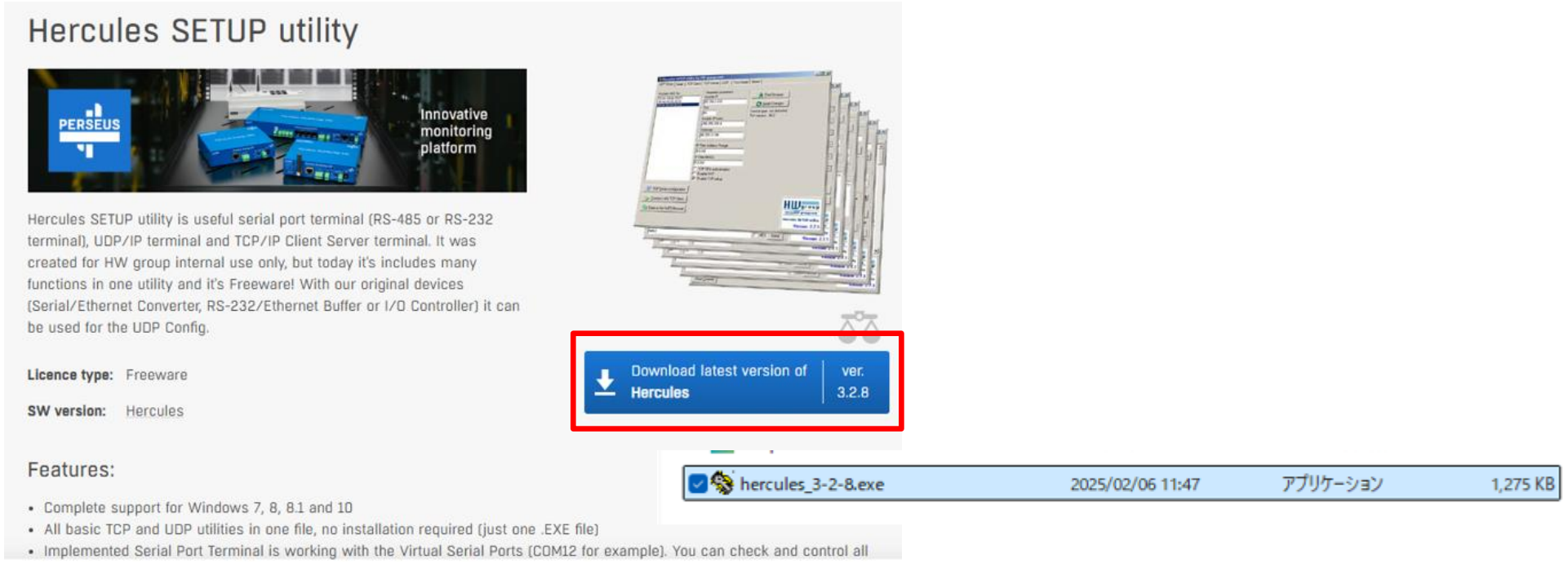
- PC側でのTCP/IP通信のやり取りに”Hercules SETUP utility”アプリケーションを使用します。
  インターネット環境のPCでWebブラウザを開き、検索ボックスに”hercules setup utility”と入力し、検索します。
  検索結果の”Hercules SETUP utility”を開きます。



Hercules SETUP utility側での段取り

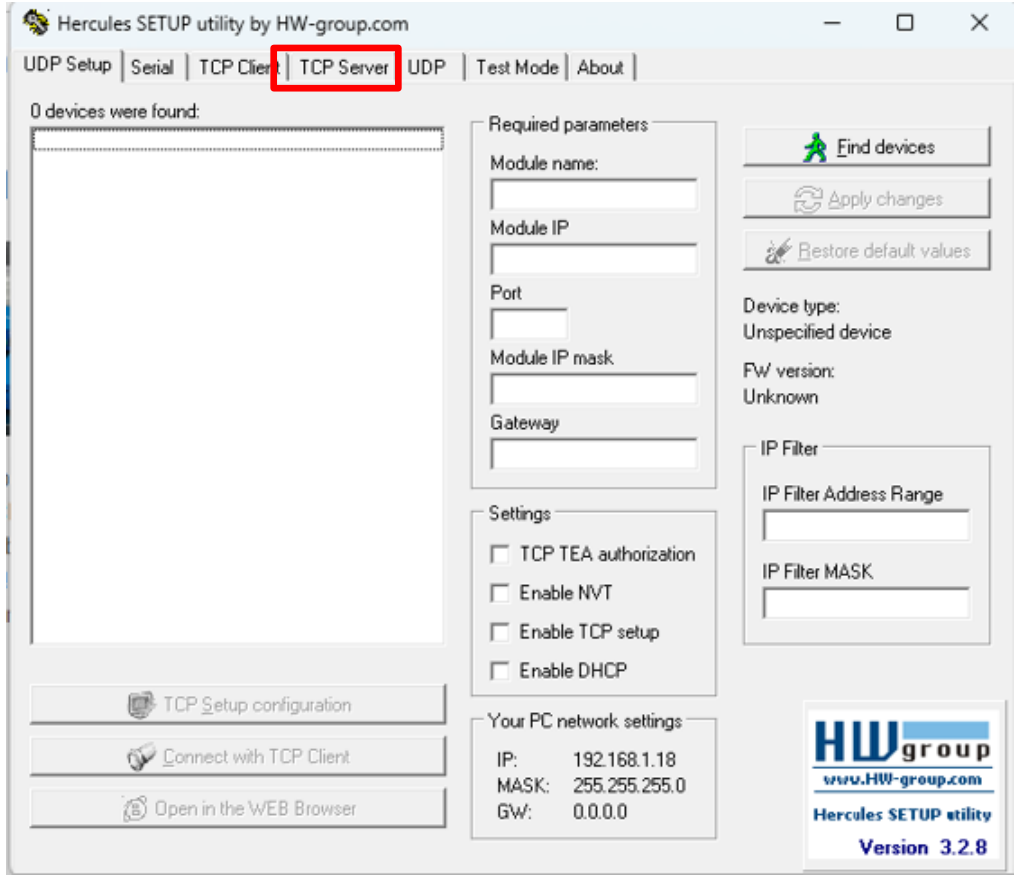
- “Download latest version of Hercules”をクリックします。

  “Hercules_*-*-*.exe”ファイル(*はversionにより異なる)がダウンロードされますので、クリックして実行します。



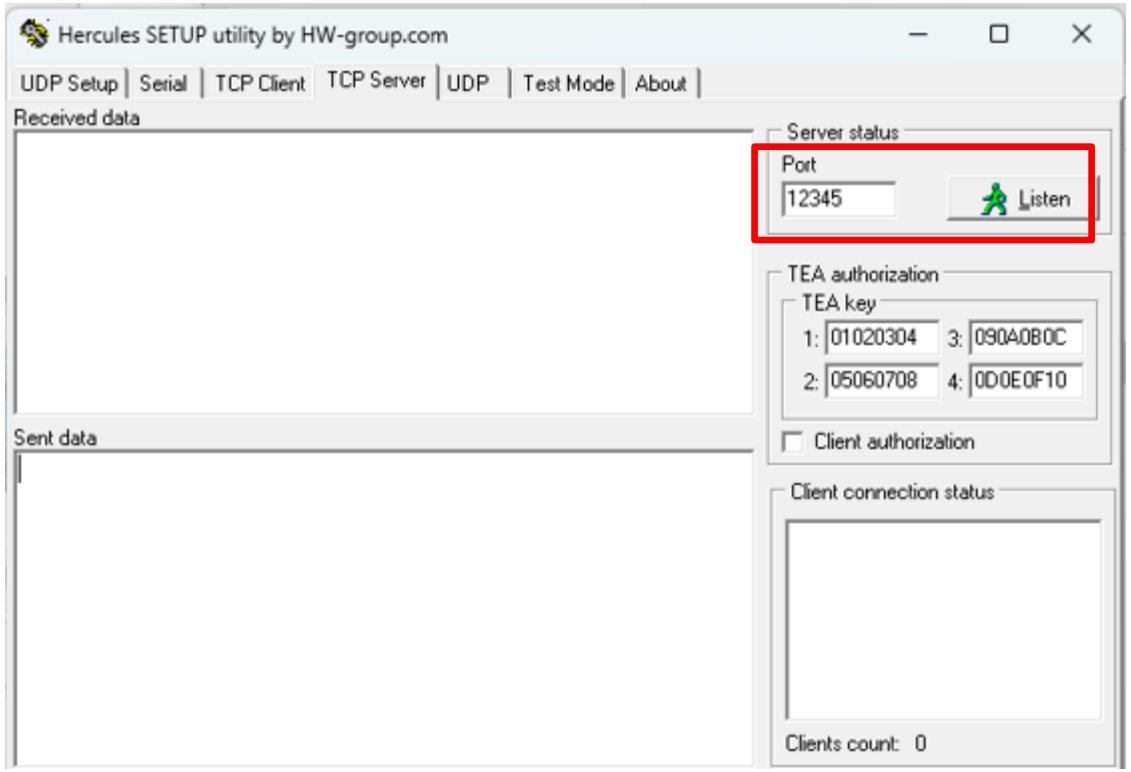
Hercules SETUP utility側での段取り

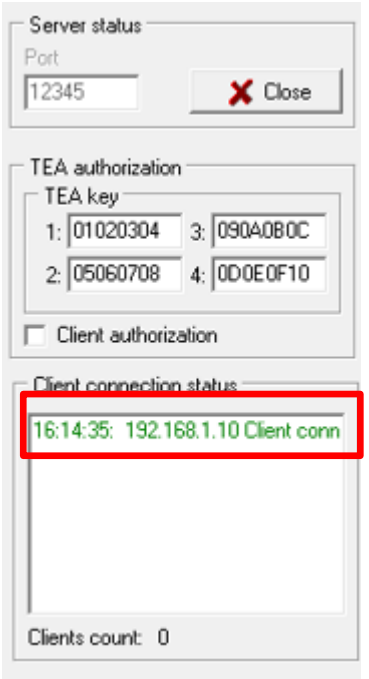
- exeファイルをクリックすると、下記のような画面が表示されます。
  “TCP Server”をクリックします。



Hercules SETUP utility側での段取り

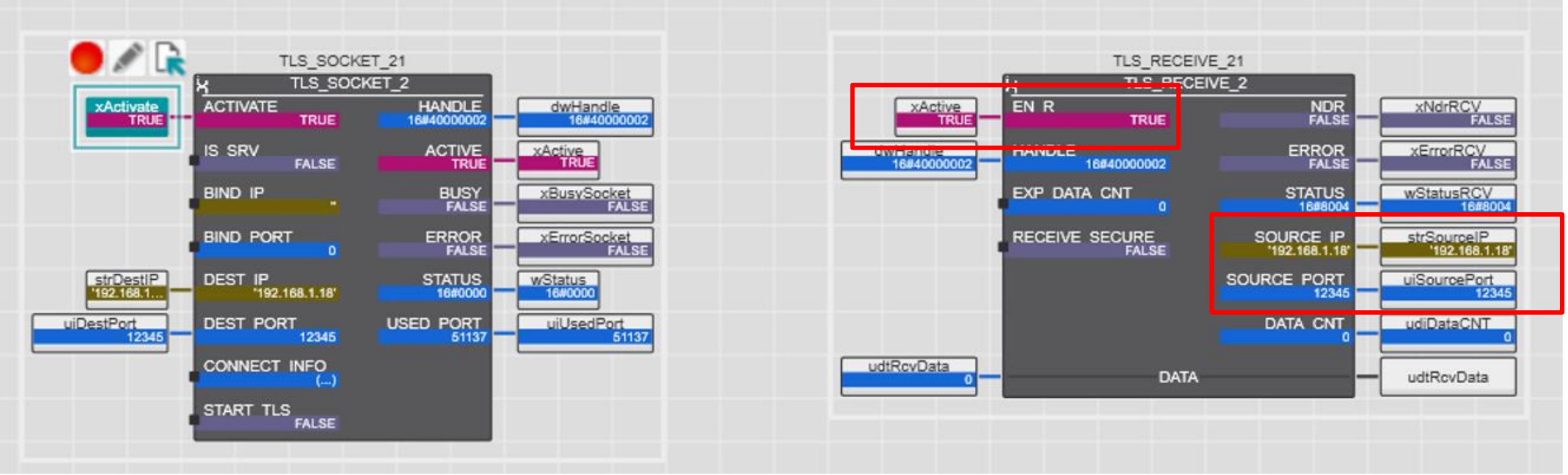
- “TCP Server”画面に遷移します。”Server status”のPort番号に”12345”を入力し、”Listen”をクリックします。
  PCがAXC F 2152とコネクションを開始します。





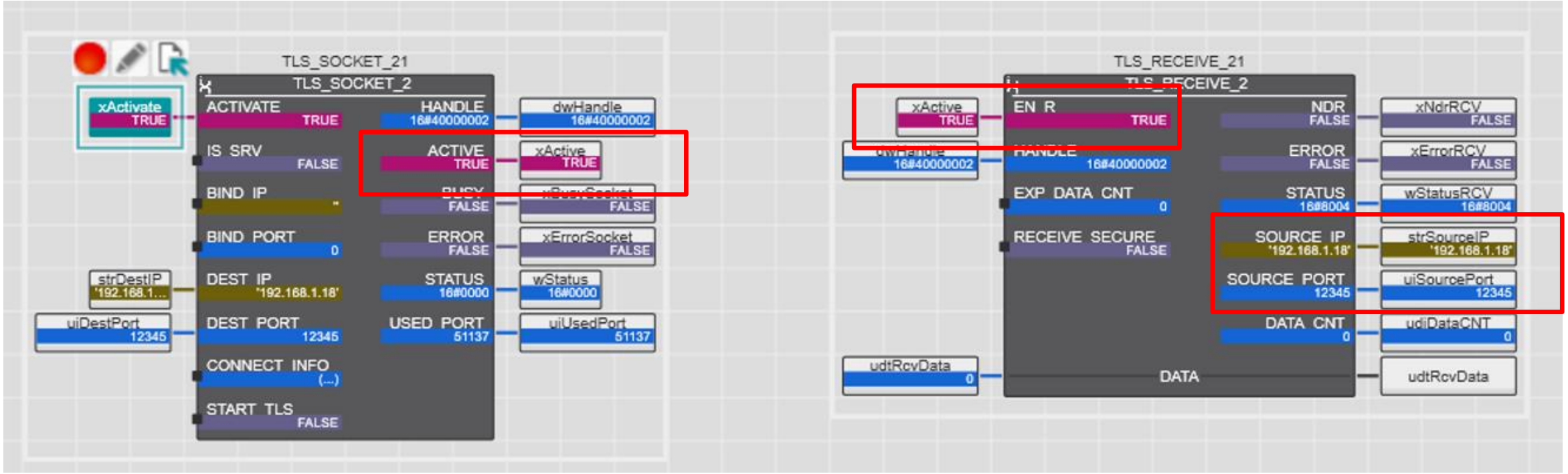
TCP/IP通信の確認

- 再度PLCnext Engineerの画面に戻ります。すると“TLS_RECEIVE_21”の”xActive”がTRUEになり、
  PC側のIPアドレス及び先ほどHercules SETUP utilityで設定したport番号が返されます。
  TCP/IP通信の接続が完了したことを示します。



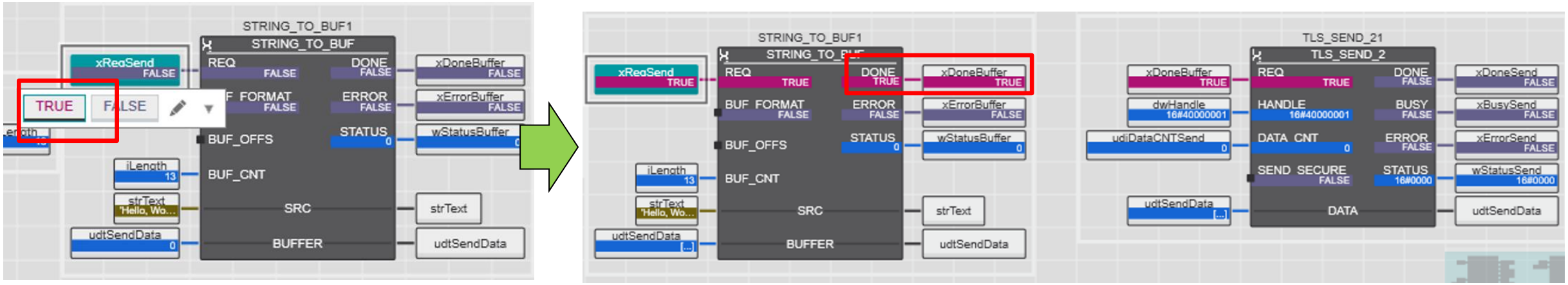
TCP/IP通信の確認

- 再度PLCnext Engineerの画面に戻ります。すると“TLS_SOCKET_21”の”xActive”がTRUEになり、
  PC側のIPアドレス及び先ほどHercules SETUP utilityで設定したport番号が返されます。
  TCP/IP通信の接続が完了したことを示します。



TCP/IP通信の確認

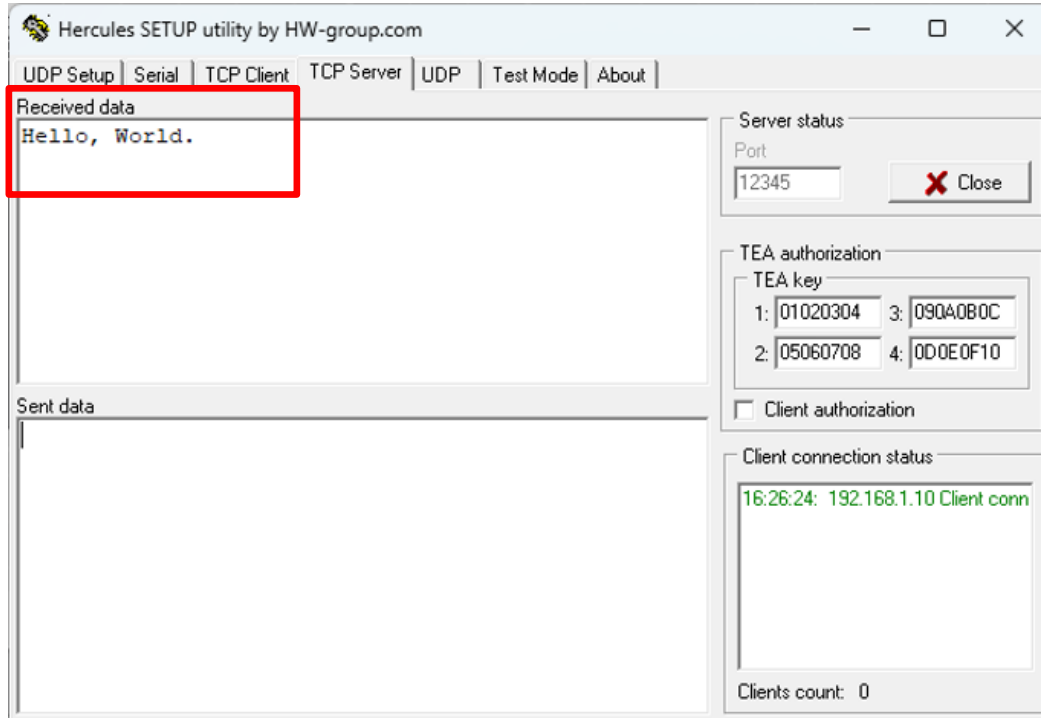
- AXC F 2152側からPCへメッセージ(“Hello, World.”)を送信します。
  “STRING_TO_BUF1”の”xReqSend”を”TRUE”にします。
  すると”xDoneBuffer”がTRUEとなり、
  ”strText”の文字データが”udtSendData”を経由してPCへ送信されたことを示します。



TCP/IP通信の確認

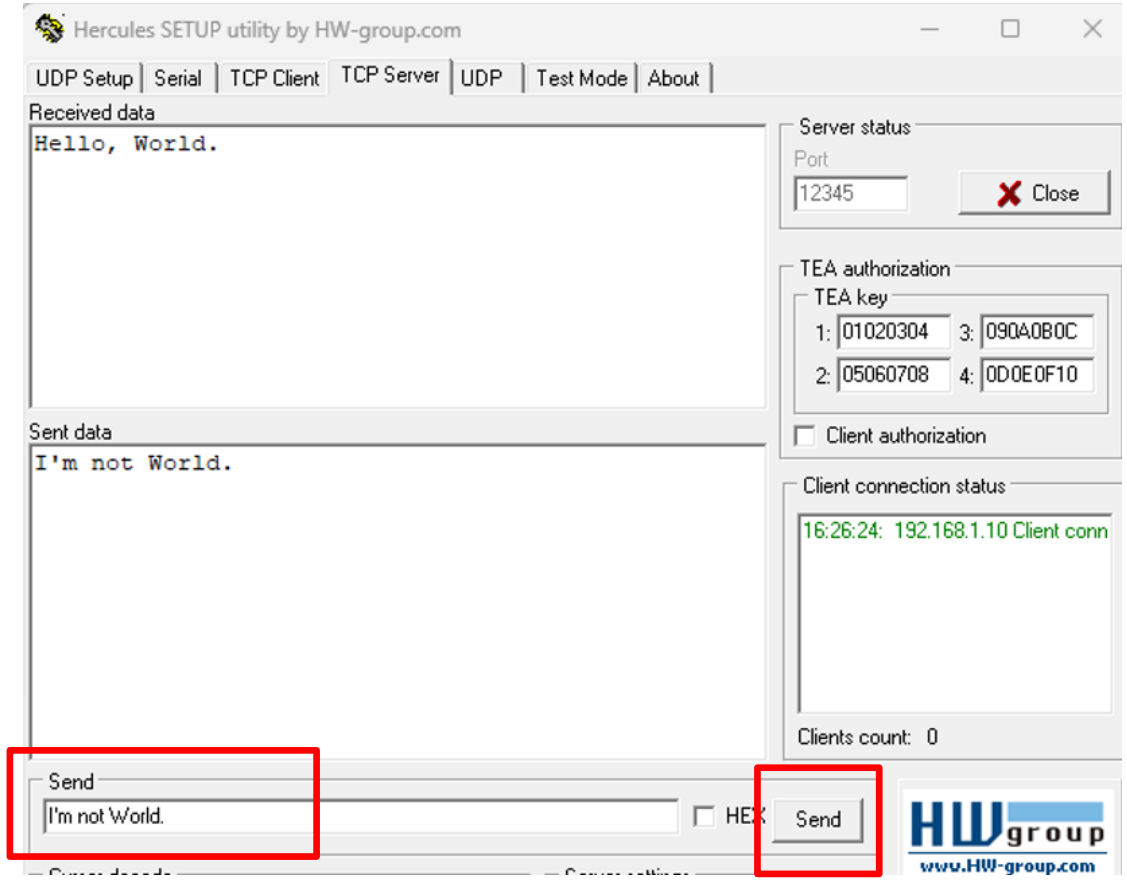
- “Hercules SETUP utility”に戻ります。

  “Received data”欄に”Hello, World.”が表示されることを確認します。



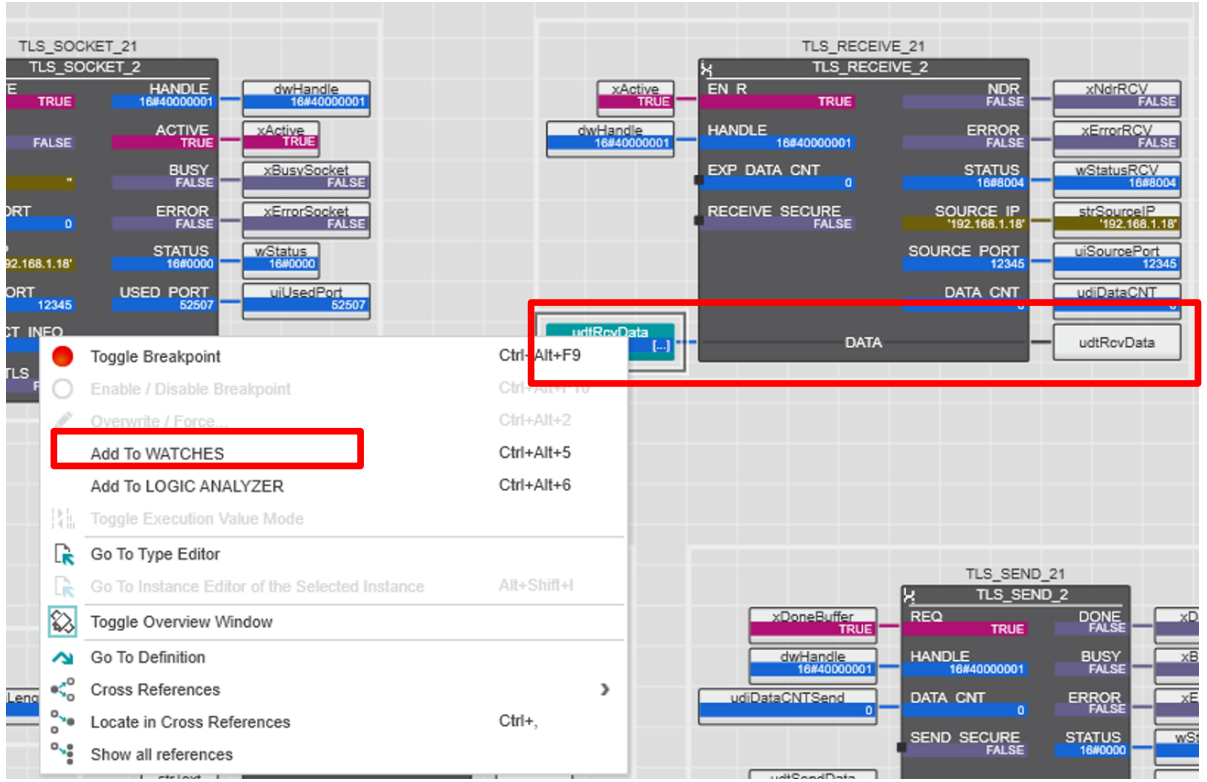
TCP/IP通信の確認

- 今度はPC側からAXC F 2152へメッセージを送信します。“Send”欄に”I’m not World.”と入力し、“Send”ボタンをクリックします。(同時に”Sent data”欄に同様の文字列が表示されます)



TCP/IP通信の確認

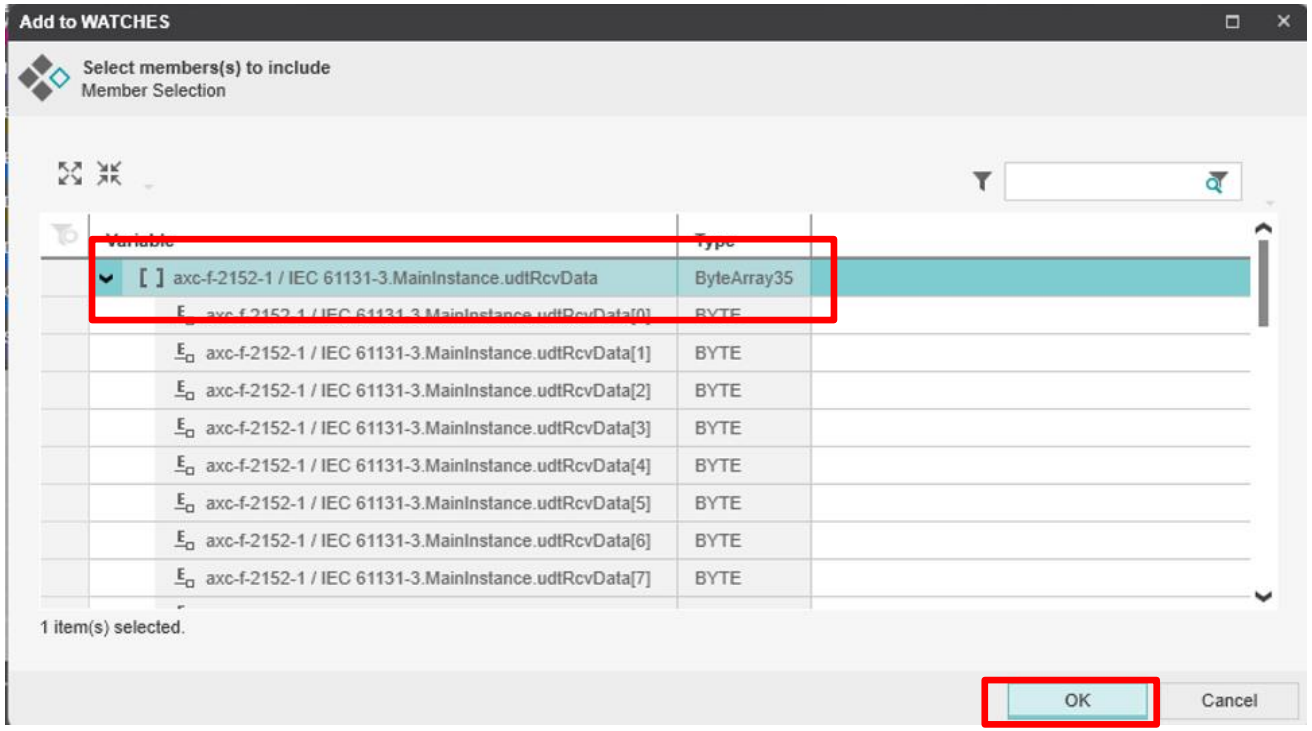
- 再度PLCnext Engineerの画面へ戻ります。PC側から送られたメッセージは”TLS_RECEIVE_21”の“udtRcvData”に転送されます。”udtRcvData”を右クリックし、”Add To WATCHES”をクリックします。



TCP/IP通信の確認

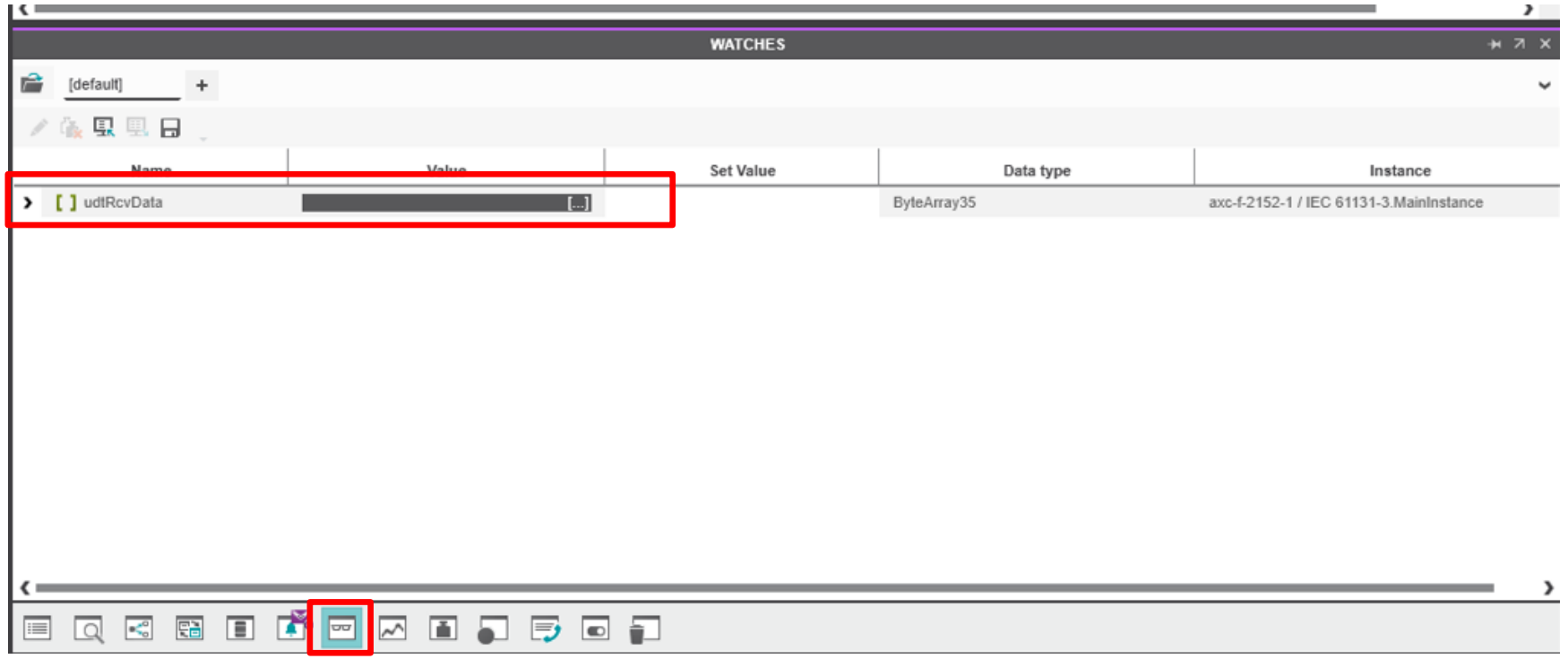
- ”Add To WATCHES”のダイアログ画面が表示されます。

  一番上の項目がアクティブ状態かを確認の上、”OK”をクリックします。



TCP/IP通信の確認

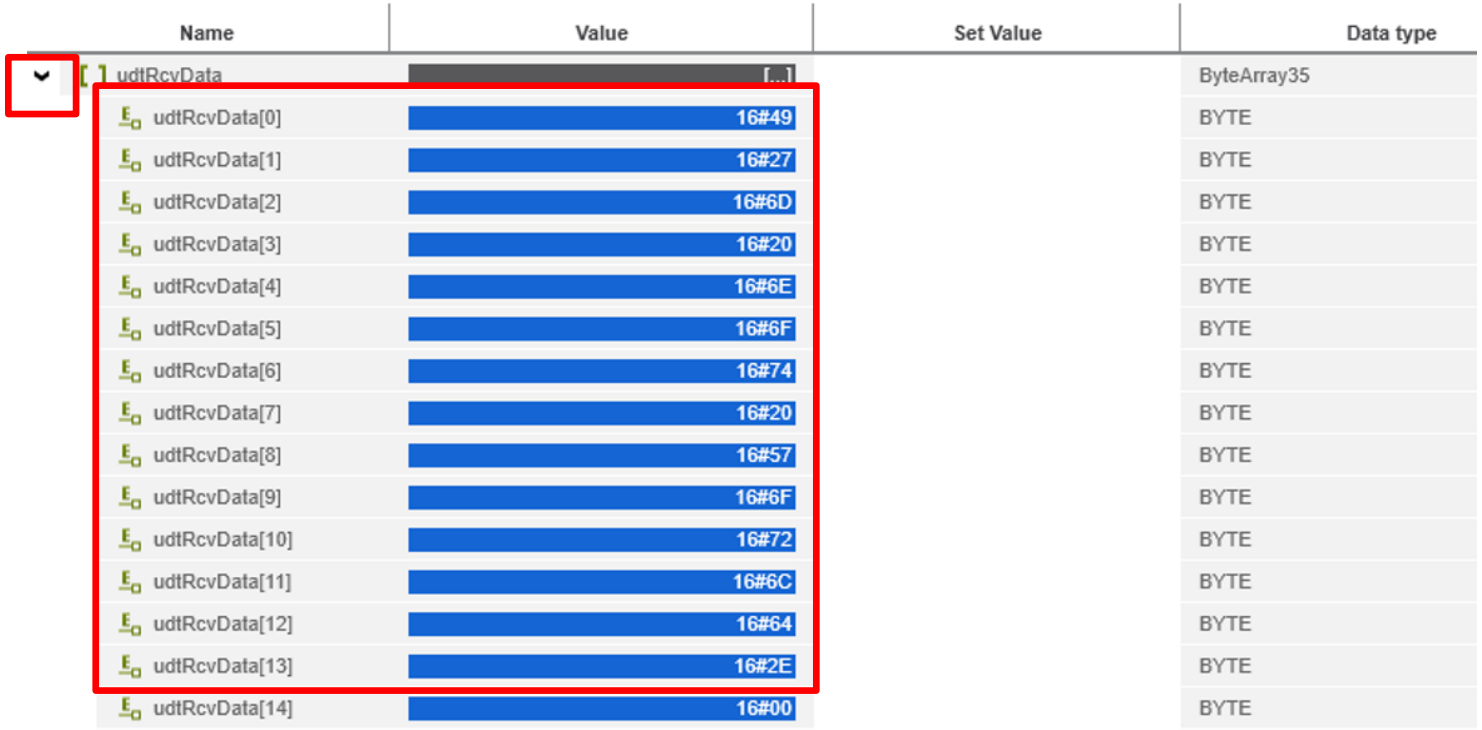
- 画面真ん中下部に アイコンがありますのでそちらをクリックするとWATCHES画面が表示されます。
  先ほど登録した”udtRcvData”が表示されています。



TCP/IP通信の確認

- 変数を展開すると各バイトのデータが表示されます。

  アスキーコードの値で格納されているため、各々のバイトをまとめて”I’m not World.”が送られていることが分かります。



Name	Value	Set Value	Data type
udtRcvData	[...]		ByteArray35
udtRcvData[0]	16#49		BYTE
udtRcvData[1]	16#27		BYTE
udtRcvData[2]	16#6D		BYTE
udtRcvData[3]	16#20		BYTE
udtRcvData[4]	16#6E		BYTE
udtRcvData[5]	16#6F		BYTE
udtRcvData[6]	16#74		BYTE
udtRcvData[7]	16#20		BYTE
udtRcvData[8]	16#57		BYTE
udtRcvData[9]	16#6F		BYTE
udtRcvData[10]	16#72		BYTE
udtRcvData[11]	16#6C		BYTE
udtRcvData[12]	16#64		BYTE
udtRcvData[13]	16#2E		BYTE
udtRcvData[14]	16#00		BYTE

Thank you

PHOENIX CONTACT
INSPIRING INNOVATIONS