## 1　任意のポート番号を使用して通信を行う

当社の Ethernet 通信バスコントローラ（750-841）は Modus/TCP(UDP)を標準装備していますが、Codesys（WAGO-IO-PRO CAA）の Ethernet ライブラリを利用してプログラミングすることにより、任意のポート番号（80、25、502 等代表的なﾎﾟｰﾄ番号は除く）で他の Ethernet 機器と通信する送受信プログラムを作成することができます。この資料では 750-841 がクライアント(ｻｰﾊﾞにﾎﾟｰﾄｵｰﾌﾟﾝ動作を行う) Ethernet_Client と クライアントからリクエストを受け付ける Ethernet_server　のファンクションブロックについて基本的な動作をサンプルプログラムと共に解説します。尚、Ethernet_Client と Ethernet_server については　ST(ｽﾄﾗｸﾁｬｰﾄﾞﾃｷｽﾄ) で記述されておりプログラムの内容はすべてオープンとなっております。

## 2．　プログラムのダウンロード

ダウンロードしたライブラリとプログラムファイルを自己解凍後、フォルダにコピーしてください。

1）＜ライブラリ＞

841library.EXE を解凍すると　Ethernet.lib　と　Ethernet_OpenClose.lib　の２つのファイルが出てきますのでこれを以下のディレクトリにコピーしてください。

コピー先→

C:¥Program Files¥WAGO Software¥CoDeSys V2.3¥Targets¥WAGO¥Libraries¥32_Bit

2）＜プログラム＞

解凍後フォルダごと以下のディレクトリにコピーしてください。

コピー先→

C:¥Program Files¥WAGO Software¥CoDeSys V2.3¥Projects

その後 Codesys(WAGO-IO-PRO CAA)を起動して各々のプロジェクトを開きます。以下の４つのプログラムが入っています

① 841_EthernetClientExample.pro
通信先（サーバ）に対して指定された任意のポート番号で接続要求をしてデータを送受信するための基本的なプログラムです。自分側から通信先（サーバ）に対してポートオープン要求を行います。

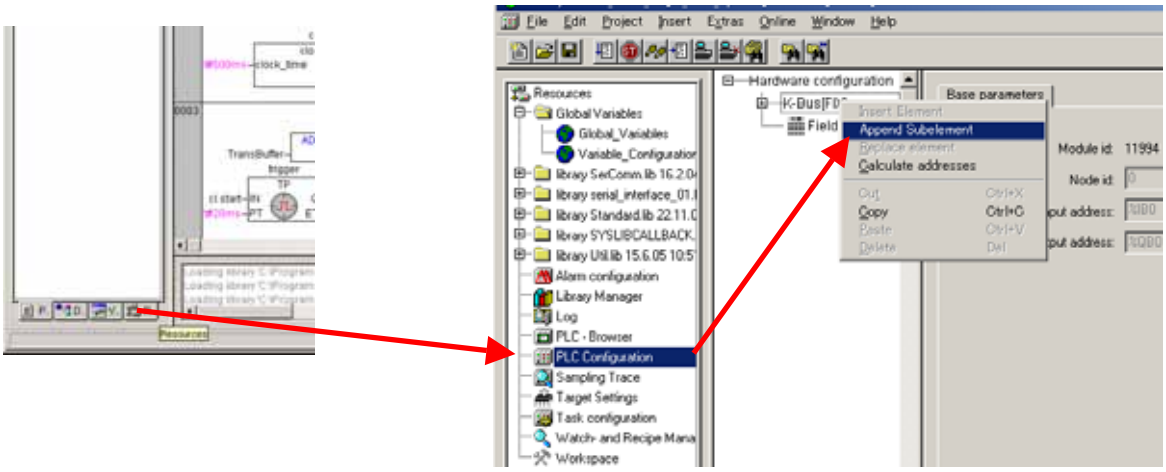
② 841_EthernetServerExample.pro
クライアント側から任意のポート番号で接続要求があった場合にこれに応答して受動的にコネクションを確立してデータを送受信するためのプログラムです。

③ 841_EthernetClientExample_1.pro および 841_EthernetServerExample_1.pro
この２つのプログラムはクライアントからビットの操作をして、その状態を受信したサーバ側が送られたデータを判断して、対応したデータを返信するという２つ一組で動作するサンプルプログラムです。

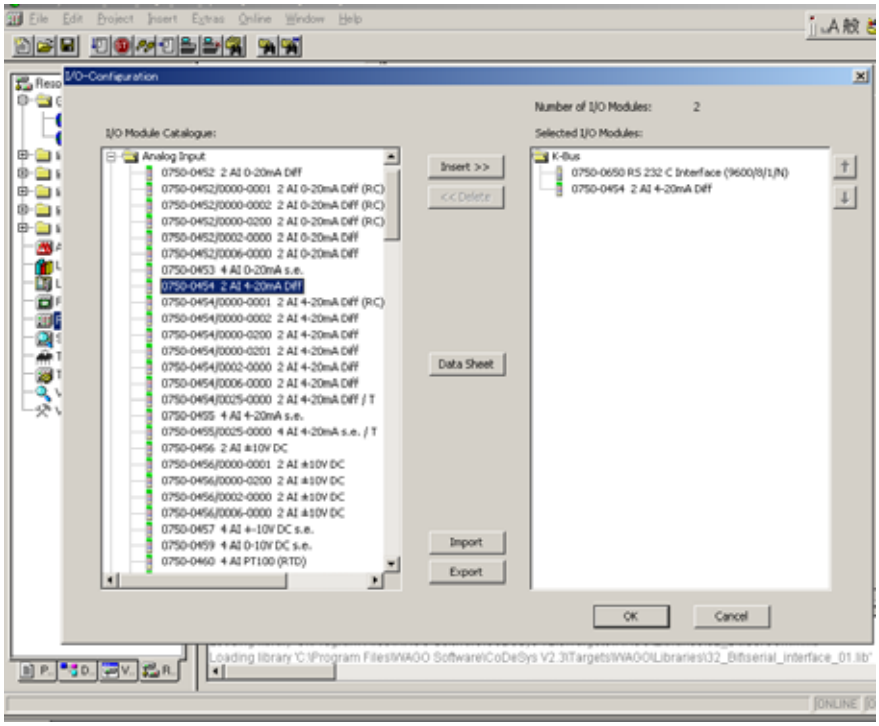
## 3． PLC コンフィグレーションの設定

Codesys(WAGO-IO-PRO-CAA)では、PLC コンフィグレーションを正しく行わないと動作しないことがあります。リソースメニューから PLC Configuration メニューをクリックすると以下のように Hardware configuration 画面が表示されますので実際に接続されているﾓｼﾞｭｰﾙに合わせて PLC ｺﾝﾌｨｸﾞﾚｰｼｮﾝを行ってください。

K-Bus[FIX] を右クリックすることによって ポップアップメニューを出して Append Subelemet をクリックするとモジュールを選択する画面が出てきます。

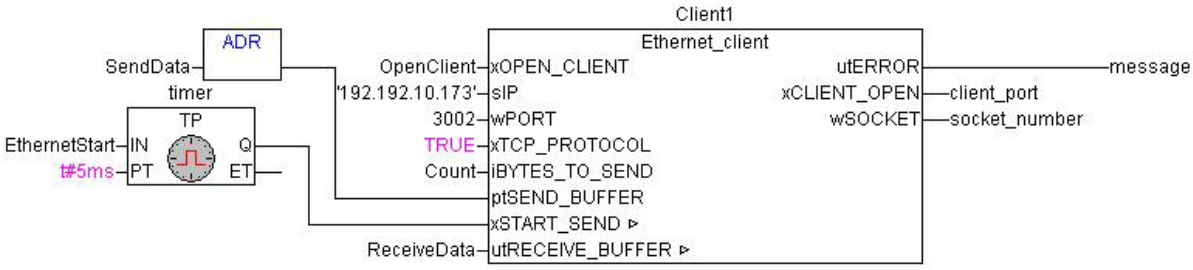



バスカプラに近い順番に接続されているモジュールを選択して追加していってください。該当するモジュールが見当たらない場合は、同じ種類の同じビット幅のモジュールを選択してください。



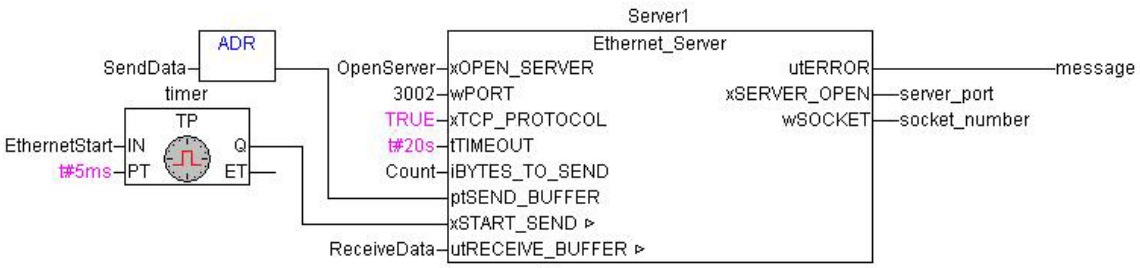
# 4．　ファンクションブロック（FB）の解説

●Ethernet_Client ファンクションブロックのパラメータ




| 入力パラメータ | データ形式 | 説明 |
|---|---|---|
| xOPEN_Client | BOOL | ここに入力されるビットが ON になった時、サーバに対してポートオープンのリクエスト送信を行います。ここを FALSE にするとポートをクローズします。 |
| sIP | String | 通信先（サーバ）の IP アドレスを文字列で入力します。 |
| wPORT | INT | 通信したいポート番号（destination）を入力します。 |
| xTCP_PROTOCOL | BOOL | ここに TRUE を入力すると TCP 通信、FALSE を入力すると UDP 通信を行います。 |
| ptSEND_BUFFER | BYTE ARRAY、STRING など | 送信データは BYTE　ARRAY（バイト列）や STRING（文字列）などが可能ですが、一度変数に入力してからその変数をアドレスポインタで指定してください。 |
| iBYTES_TO_SEND | INT | ポートオープン後、こちら（クライアント側）から送信したいデータの総バイト数を入力します。 |
| xSTART_SEND | BOOL | ON の立ち上がりエッジで送信動作を行ないます。ON を継続すると連続で送信してしまいます。またこの変数は IN_OUT 変数であるため直接絶対アドレス（%IX など）を直接入力することはできません。TP（FB）を介して数 mS 程度のパルスを入れるか、他の FB の IN_OUT に割り当てられる変数を利用してください。 |
| utRECEIVE_BUFFER | typEthernet_buffer（専用型式） | サーバ側から返信されたバイトを受信するバッファです。1500 バイトのリングバッファになっています。（ここで定義した名前）.Data[1]～[1500]にバイト単位で受信したデータが格納され、（ここで定義した名前）.Index でリングバッファのどの位置まで入ったか（１～1500）を整数で返します。 |
| utERROR | ETH_ERROR（専用型式） | Ethernet 通信が正常に行われている時、あるいは通信先のサーバが存在していない時は　０　を表示します。ポートオープンの動作を行っていて、サーバが存在していても、応答しない時は３２７９０を表示します。 |
| xCLIENT_OPEN | BOOL | 現在ポートをオープン中かどうかを BOOL 値で示します。 |
| wSOCKET | WORD | 電源投入時から、サーバ側に対してポートオープンをかけた回数とサーバ側かたタイムアウトの RST パケットを受信した回数の合計値。65535 までカウントすると 0 に戻ります。 |

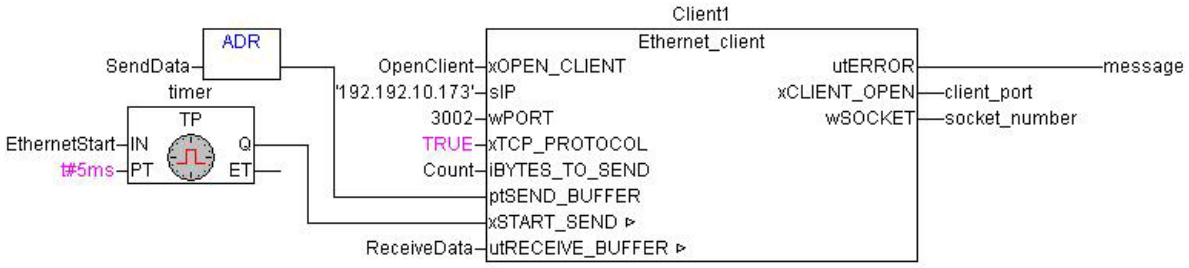
●Ethernet_Server ファンクションブロックのパラメータ設定




| 入力パラメータ | データ形式 | 説明 |
|---|---|---|
| xOPEN_SERVER | BOOL | ここに入力されるビットが ON に成っている時、Client 側からの通信要求に応答します。ここが FALSE の場合は応答動作行いません。 |
| wPORT | INT | 通信したいポート番号（Client 側から見るとこのポート番号は destination のポート番号になります）を入力します。Client 側で設定するポート番号とあわせておく必要があります。 |
| xTCP_PROTOCOL | BOOL | ここに TRUE を入力すると TCP 通信、FALSE を入力すると UDP 通信を行います。 |
| tTIMEOUT | TIME | TCP 接続において、クライアント側がポートクローズの通知を行わないで動作を終了した場合（電源断など）、設定した時間内で自らのポートを自動的にクローズする時間を設定します。 |
| iBYTES_TO_SEND | INT | こちら（サーバ側）から送信したいデータの総バイト数を入力します。 |
| ptSEND_BUFFER | BYTE ARRAY 、STRING 等 | 送信データは BYTE ARRAY（バイト列）や STRING（文字列）などが可能ですが、一度変数に入力してからその変数をアドレスポインタで指定してください。 |
| xSTART_SEND | BOOL | ON の立ち上がりエッジで送信動作を行ないます。ON を継続すると連続で送信してしまいます。またこの変数は IN_OUT 変数であるため直接絶対アドレス（%IX など）を直接入力することはできません。TP（FB）を介して数 mS 程度のパルスを入れるか、他 FB の IN_OUT に割り当てられる変数を利用してください。通常このタイミング信号はクライアントからポートをオープンされている状態で送信するもので、相手からポートオープンされていないにも関わらず送信を続けると 750-842 の動作が途中でハングアップしてしまいますのでご注意ください。 |
| utRECEIVE_BUFFER | typEthernet_buffer（専用型式） | クライアント側から送信されたバイトを受信するバッファです。1500 バイトのリングバッファになっています。（ここで定義した名前）.Data[1]～[1500]にバイト単位で受信したデータが格納され、（ここで定義した名前）.Index でリングバッファのどの位置まで入ったか（1～1500）を整数で返します。 |
| utERROR | ETH_ERROR（専用型式） | Ethernet 通信が正常に行われている時は「NO_ERROR」を表示します。 |
| xCLIENT_OPEN | BOOL | 現在 FB が動作中かどうかを BOOL 値で示します。 |
| wSOCKET | WORD | 電源投入時から数えて、クライアントからポート・クローズされた回数およびタイムアウト設定によって RST パケットを送信した回数の合計値。65535 までカウントすると 0 に戻ります。 |

## ５．　サンプルプログラムの解説

- 841_EthernetClientExample.pro の動作解説




①

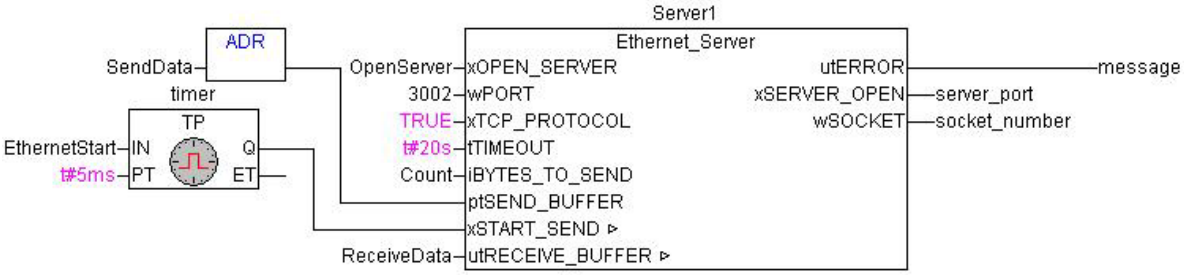
OpenClient を **TRUE** にするとサーバ（この場合は 192.192.10.173）にポート接続の動作を行います。サーバが見つかり、コネクションを確立した場合は massage には常に　0 が出力されています。サーバが見つからない場合は３２７９０が出力されます。コネクションを確立しておきたい場合は　**TRUE**　を保持してください。

②EthernetStart に **TRUE** を代入するとその立ち上がりエッジで ptSEND_BUFFER に入力されている変数の値を　Count　(iBYTES_TO_SEND)で設定されたバイト数だけ頭から送信します。Timer(TP)で設定するパルス時間は数 ms～10ms くらいに設定してください。

③SendData は BYTE　ARRAY（バイトデータの一次元配列）で定義してありますが、STRING で定義して文字列を代入しておいても自動的に ASCII バイトに変換して送信することが可能です。

④サーバから受信したデータ（ｱﾌﾟﾘｹｰｼｮﾝ層に載っているデータのみ）は ReceiveData.Data[1]～ReceiveData.Data[1500]のリングバッファに格納されます。ReceveData.Index でリングバッファのどの位置まで入ったか（１～1500）を整数で得られますのでこれを利用するとリングバッファ上データを必要分だけ読み出すなどの処理が可能です。

● 841_EthernetServerExample.pro の動作解説




①OpenServer を **TRUE** にすることによってクライアントからのポート接続要求に応答するようになります。通常の通信を行う場合にはここは必ず　**TRUE**　にしておいてください。

②EthernetStart に TRUE を代入するとその立ち上がりエッジで ptSEND_BUFFER に入力されている変数の値を　Count　(iBYTES_TO_SEND)で設定されたバイト数だけ頭から送信します。Timer(TP)で設定するパルス時間は数 ms～10ms くらいに設定してください。TCP 通信おいてはクライアントからポートをオープンされている状態で送してください。相手からポートオープンされていないにも関わらず送信を続けると 750-842 の動作が途中でハングアップしてしまいますのでご注意ください。

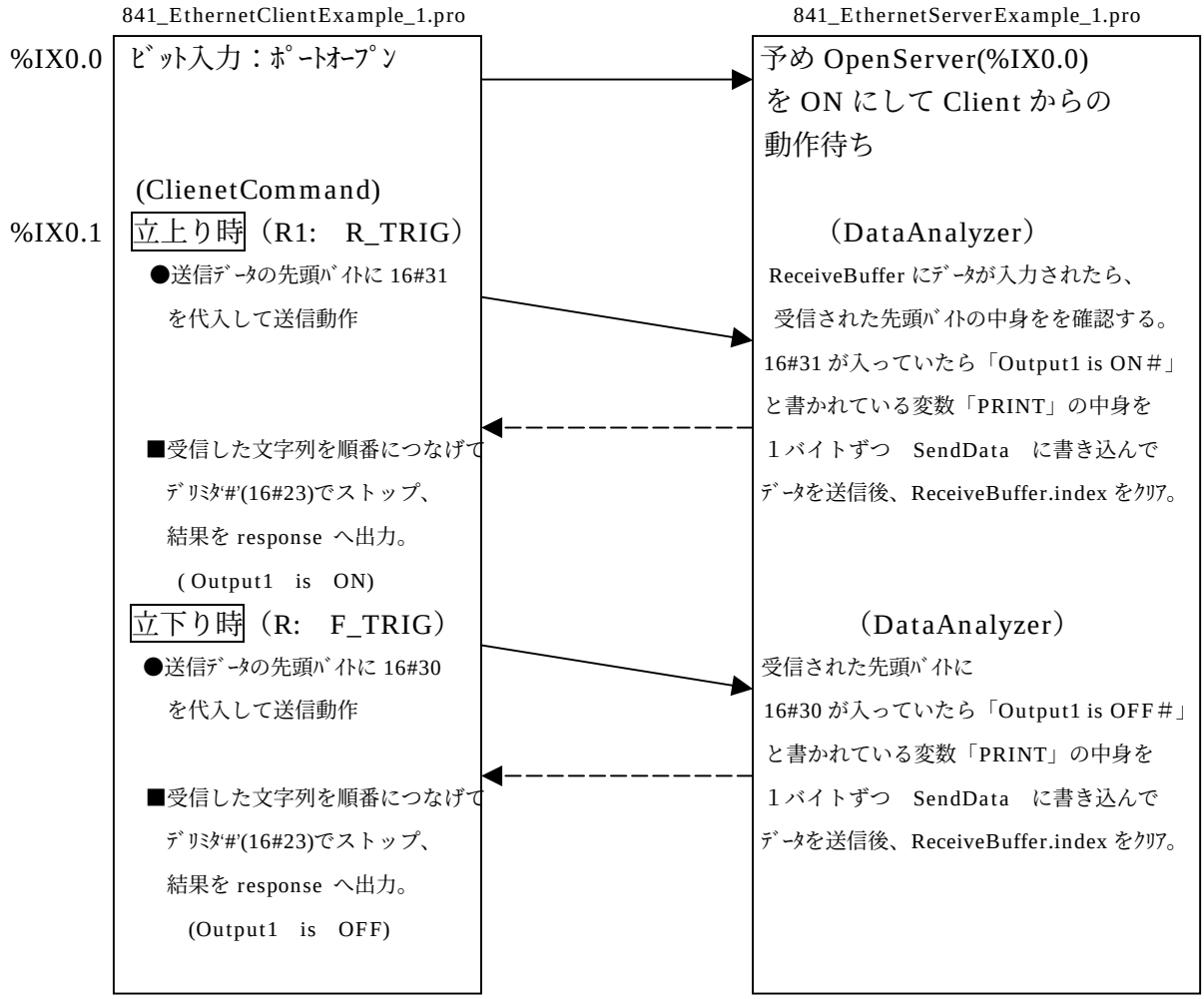
③SendData は BYTE　ARRAY（バイトデータの一次元配列）で定義してありますが、STRING で定義して文字列を代入しておいても自動的に ASCII バイトに変換して送信することが可能です。

④TCP 接続において、クライアント側がポートクローズの送信を行わないで動作を終了した場合（例えば電源断など）、設定した時間内で自らのポートを自動的にクローズします。

⑤サーバから受信したデータ（ｱﾌﾟﾘｹｰｼｮﾝ層に載っているデータのみ）は ReceiveData.Data[1]～ReceiveData.Data[1500]のリングバッファに格納されます。ReceveData.Index でリングバッファのどの位置まで入ったか（１～1500）を整数で得られますのでこれを利用するとリングバッファ上データを必要分だけ読み出すなどの処理が可能です。

●841_EthernetClientExample_1.pro　および　841_EthernetServerExample_1.pro

これらのプログラムは一対でクライアント送信とサーバ応答のプログラム例を組んだ例です。クライアントからポートオープンの動作でコネクションを確立した後は、相互に通信することが可能ですが、通常はクライアントからリクエスト信号（バイト列）を送って、これをサーバ側で解釈し、これに対応する信号（バイト列）で返すという動作が一般的です。




これらのプログラムに使用している　Ethernet_Client　と　Ethernet_Server　のﾌｧﾝｸｼｮﾝﾌﾞﾛｯｸは TCP/IP の標準手順に従っていますので、Ethernet ポートが装備された Ethernet 機器でポート番号とリクエスト、アンサーのデータ形式が決まっていればこれらのﾌｧﾝｸｼｮﾝﾌﾞﾛｯｸを使ってこれらの機器と通信させることが可能です。但し、ポート番号 80、25 等の既に規格化されているポート番号と Modbus/TCP（ポート番号 502）については設定できませんのでご注意ください。

以上