

documentation modbus TCP client ADDON

Table des matières

MB_TCP_ReadWriteReg	3
Comment fonctionne le bloc fonction ?	3
la machine a état	3
TestReadWrite	5
Comment fonctionne le programme de test ?	5
Test du programme	6

MB_TCP_ReadWriteReg

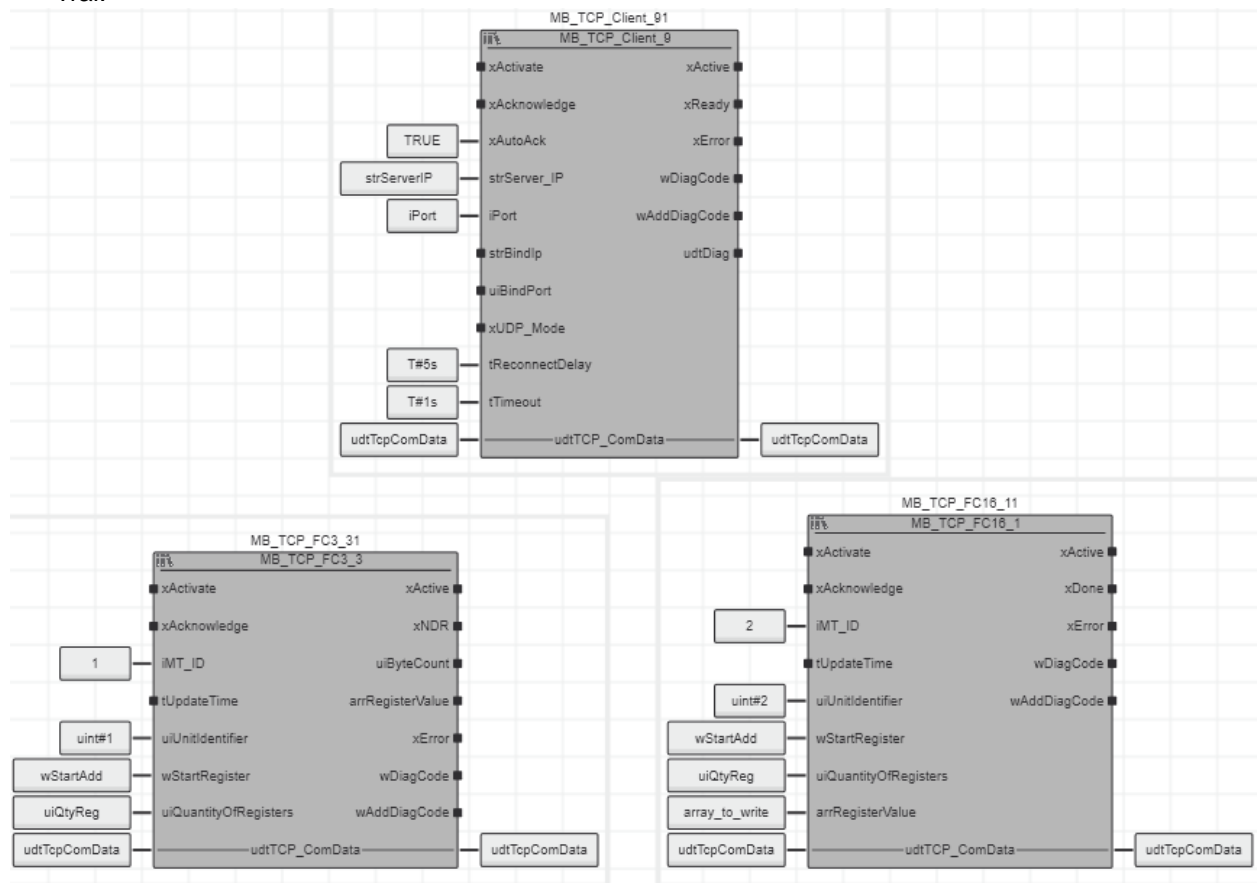
Le bloc fonction **MB_TCP_ReadWriteReg** est fourni en l'état. Il permet de lire et/ou écrire (fonctions FC03 et FC16) un tableau de mots en utilisant la librairie **Modbus_TCP** que l'on peut trouver dans Plcnext Store. La bibliothèque utilisée est Modbus_TCP_13.zip.

Créé avec HelpNDoc Personal Edition: [Générateur de documentation Qt Help gratuit](#)

Comment fonctionne le bloc fonction ?

Le bloc fonction **MB_TCP_ReadWriteReg** est constitué de 3 blocs fonctions issus de la librairie Modbus_TCP:

- **MB_TCP_Client** : qui permet la connexion au serveur, par défaut le port est le 502
- **MB_TCP_FC3** : qui est activé une fois que la connexion a été établie et si la variable xAskRead est vrai.
- **MB_TCP_FC16** : qui est activé une fois que la connexion a été établie et si la variable xAskWrite est vrai.



Créé avec HelpNDoc Personal Edition: [Générateur de documentation iPhone gratuit](#)

la machine a état

Une machine a été gère le bon fonctionnement de l'ensemble :
En cas de bon fonctionnement ou d'erreur, un message est affiché

CASE iState OF

```
0 :(* Connect to server *)
  TON1(IN := FALSE); //reset timer
  MB_TCP_FC3_31.xActivate:=FALSE; //desactivate FC3 (wait connexion to server)
  MB_TCP_FC16_11.xActivate := FALSE; //desactivate FC16 (wait connexion to server)
```

```

MB_TCP_Client_91.xActivate:=xConnect; //try to connect to server
if ( MB_TCP_Client_91.xReady = TRUE ) THEN // Connexion OK ?
    iState := 10;
    xError := FALSE;
    strError:= 'com OK';
ELSIF ( MB_TCP_Client_91.xError = TRUE ) THEN
    iState := 100;
END_IF

10 : (* What to do : READ OR WRITE ?*)
TON1(IN := FALSE); //reset timer
MB_TCP_FC3_31.xActivate:=FALSE; //desactivate FC3 (each time)
MB_TCP_FC16_11.xActivate := FALSE; //desactivate FC16 (each time)
if ( MB_TCP_Client_91.xReady = FALSE ) THEN (*lost connexion ?*)
    MB_TCP_Client_91.xActivate:=FALSE; //desactivate client and go back to begin
    iState:=0;
ELSE
    if ( xAskWrite = TRUE ) THEN
        iState := 20;
    elsif ( xAskRead = TRUE ) THEN
        iState := 30;
    ELSE //add message if no Write and no Read send
        strError:= 'com OK-> no frame';
    END_if;
END_IF;

20 : //Writing registers (FC16)
MB_TCP_FC16_11.xActivate := TRUE; //enable FC16 function
if ( MB_TCP_FC16_11.xError = TRUE ) THEN
    iState := 200;
elsif ( MB_TCP_FC16_11.xDone = TRUE ) THEN //is Response from server ?
    // data from MB_TCP_FC16_11.arrRegisterValue has been sent
    if ( xAskRead = TRUE ) THEN //Do we have to read registers ?
        iState := 30;
    ELSE
        iState := 10;
        xError := FALSE;
        strError:= 'FC16 OK';
    END_IF;
END_IF;

30 //Reading registers (FC3)
MB_TCP_FC3_31.xActivate:=TRUE; (*start comm FC3*)
if ( MB_TCP_FC3_31.xError = TRUE ) THEN
    iState := 300;
elsif ( MB_TCP_FC3_31.xNDR = TRUE ) THEN //is Response from server ?
    FOR i := 1 TO 125 DO //copy data read
        array_to_read[i]:=MB_TCP_FC3_31.arrRegisterValue[i];
    END_FOR;
    xError := FALSE;
    //message are different (depend on what has been ask)
    if ( xAskWrite = TRUE ) THEN
        strError:= 'com & FC3 OK';
    ELSE
        strError:= 'FC3 OK';
    END_IF;
    iState := 10;
END_IF;

100: (* ERROR Connexion TCP server *)
xError := TRUE;
strError:= CONCAT(' ', 'ERROR connexion');

```

```

MB_TCP_Client_91.xActivate:=FALSE;//reset connexion to server
TON1(IN := TRUE, PT := T#2s);//wait 2s
if ( TON1.Q = TRUE ) THEN
    iState := 0;
end_if;

```

```

200:(* Error frame FC16 *)
xError := TRUE;
strError:= CONCAT(" 'ERROR FC16'");
TON1(IN := TRUE, PT := T#2s);
MB_TCP_FC16_11.xActivate := FALSE; //reset COM FC16
if ( TON1.Q = TRUE ) THEN //wait 2s
    iState := 10;
end_if;
300:(* Error frame FC3 *)
xError := TRUE;
strError:= CONCAT(" 'ERROR FC3'");
MB_TCP_FC3_31.xActivate:=FALSE; //reset COM FC3
TON1(IN := TRUE, PT := T#2s); //wait 2s
if ( TON1.Q = TRUE ) THEN
    iState := 10;
end_if;

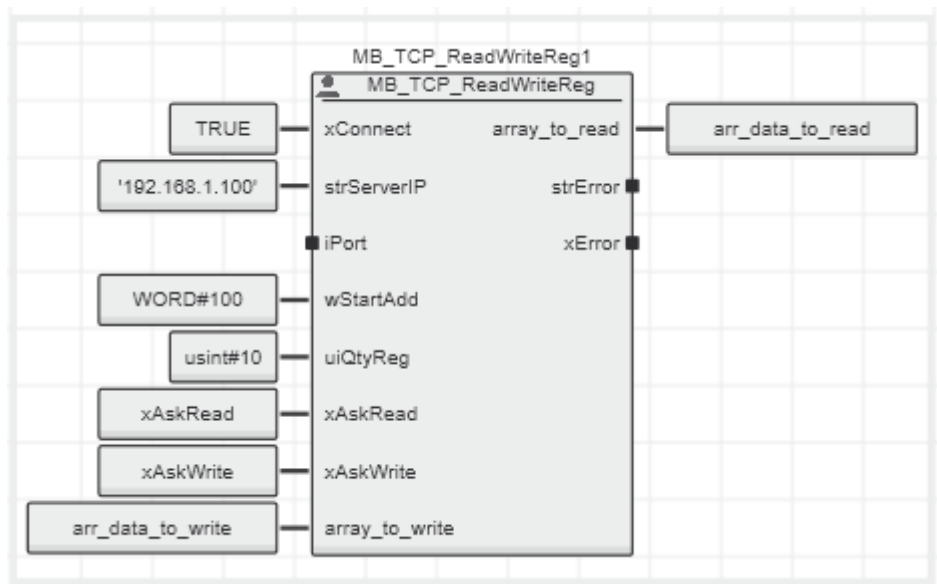
```

END_CASE

Cr    avec HelpNDoc Personal Edition: [Analyseur de projet de HelpNDoc : incroyable assistant de documentation](#)

TestReadWrite

Le programme de test du bloc fonction va se connecter automatiquement (xConnect = true) au serveur 192.168.1.100 sur le port 502 (port par d  faut). Il va alors   crire les les 10 premiers mots de MW100    MW109 (xAskWrite=TRUE) et lire ces 10 mots (AskRead = TRUE).



Cr    avec HelpNDoc Personal Edition: [Transformez votre flux de travail de documentation avec l'interface utilisateur intuitive de HelpNDoc](#)

Comment fonctionne le programme de test ?

Il y a 2 programmes ex  cut  s en plus de l'appel du bloc fonction **MB_TCP_ReadWriteReg**

le programme **init_data** qui permet d'initialiser le tableau qui sera envoyé via la fonction FC16 vers le serveur :

```
(* init du tableau au démarrage *)
if start = true then
  FOR i := 1 TO 125 DO
    arr_data_to_write[i] := TO_WOrd(i*10);
  END_FOR
  start := false;
end_if;
```

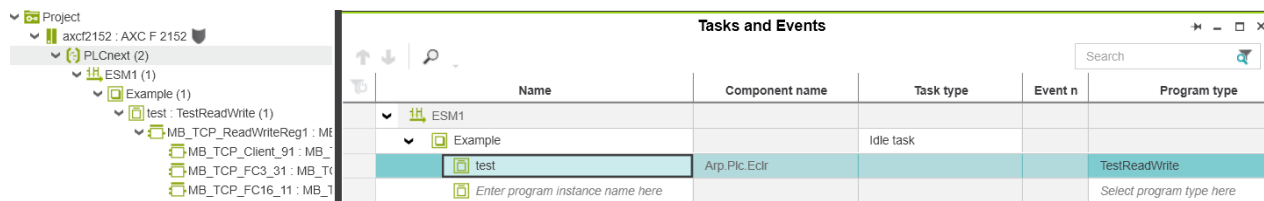
Le programme **test_prog** qui va tester les données lues sur le serveur (FC3) et les comparer aux valeurs envoyées (FC16). Si toutes les valeurs sont égales alors **ERROR = FALSE** et le test du serveur TCP modbus pour les fonction FC3 et FC16 est validé

```
(* test des 10 valeurs écrites sur le serveur modbus TCP *)
Error:=FALSE;
FOR i := 1 TO 10 DO
  if( arr_data_to_write[i] <> arr_data_to_read[i] ) THEN
    ERROR := TRUE;
  end_if;
END_FOR
```

Créé avec HelpNDoc Personal Edition: [Convertir facilement des documents Word en livres électroniques avec HelpNDoc](#)

Test du programme

Pour le test du programme **testReadWrite**, ajouter ce programme dans Plant PLCNext, on pourra utiliser Idle Task (tache la moins prioritaire).



Après avoir téléchargé le programme, passer en mode instance et visualiser les tableaux de lecture et écriture et la variable ERROR

MB_TCP_ReadWriteReg x axcf2152 / PLCnext x TestReadWrite x test x

GDS Port List [1] Variables [2] appel_FB [3] init_data [4] test_prog [5] Resources [6]

appel_FB

WATCHES

[default] Watch1 +

Name	Value	Set Value
[] arr_data_to_write	[...]	
[] arr_data_to_read	[...]	
● ERROR	FALSE	

N'oubliez pas de lancer le serveur modbus TCP.

MODBUS Eth. TCP/IP PLC - Simulator (port: 502)

Connected (1/10) : (received/sent) (465668/465667) Serv Rx: Tx:

Pause

Stop tracking

Clear

☒ Show time

☐ Log Decode

```

20:14:51.497 Read Holding Regs from 100 for 10.
20:14:51.497 TX:3B FF 00 00 00 17 01 03 14 00 0A 00 14 00 1E 00 28 00 32 00 3C 00 46 00 50 00 5A 00 64
20:14:51.528 RX:3C 00 00 00 00 1B 02 10 00 64 00 0A 14 00 0A 00 14 00 1E 00 28 00 32 00 3C 00 46 00 50 00 5A 00 64
20:14:51.528 Write multiple registers from 100 for 10 registers.
20:14:51.528 TX:3C 00 00 00 00 06 02 10 00 64 00 0A
20:14:51.560 RX:3C 01 00 00 00 1B 02 10 00 64 00 0A 14 00 0A 00 14 00 1E 00 28 00 32 00 3C 00 46 00 50 00 5A 00 64
20:14:51.560 Write multiple registers from 100 for 10 registers.
20:14:51.560 TX:3C 01 00 00 00 06 02 10 00 64 00 0A
20:14:51.605 RX:3C 02 00 00 00 06 01 03 00 64 00 0A
20:14:51.605 Read Holding Regs from 100 for 10.
20:14:51.605 TX:3C 02 00 00 00 17 01 03 14 00 0A 00 14 00 1E 00 28 00 32 00 3C 00 46 00 50 00 5A 00 64
20:14:51.636 RX:3C 03 00 00 00 1B 02 10 00 64 00 0A 14 00 0A 00 14 00 1E 00 28 00 32 00 3C 00 46 00 50 00 5A 00 64

```

Créé avec HelpNDoc Personal Edition: [Analyseur de projet de HelpNDoc : incroyable assistant de documentation](#)