APPLICATION NOTE

AC500 LINE MODE WITH AC500 V3 CPU'S

WORKING WITH A HUGE NUMBER OF MODBUS SOCKETS AT SAME TIME

Contents

1 Introduction

1.1 Scope of the document

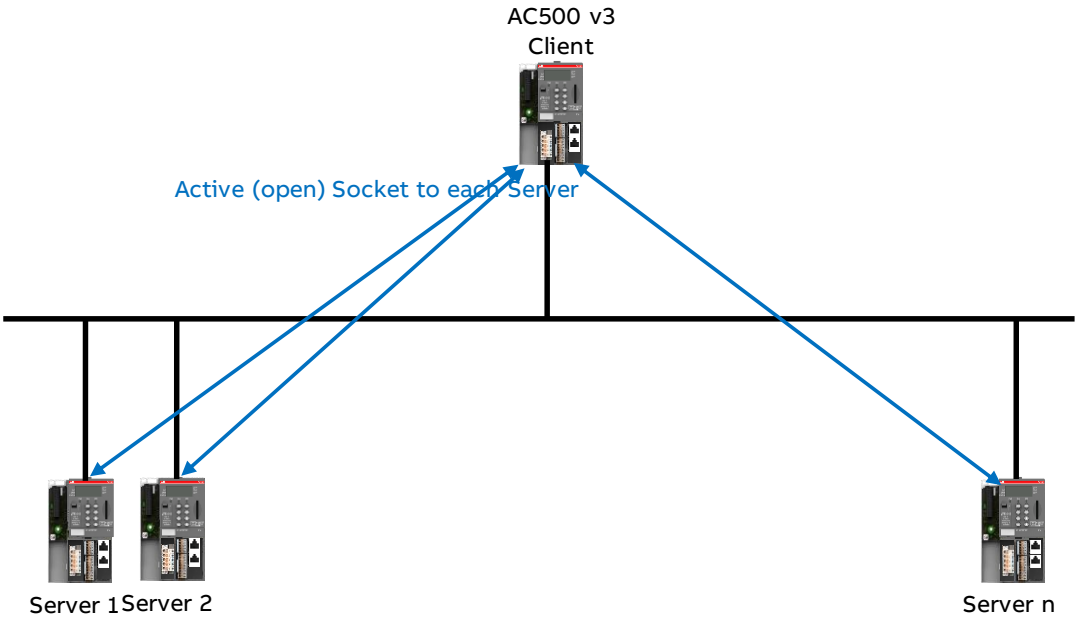
Infrastructure Projects like Tunnel need very often a very high number of Modbus Connections at the same time. To fulfill this Requirement AC500 V3 has the opportunity to increase the number of Modbus Sockets with an additional Runtime License “Line Mode”.

1.2 Compatibility

The application note is based on the below engineering system versions. They should also work with newer versions, nevertheless some small adaptations may be necessary, for future versions.

- AC500 V3 PLC FW 3.2.x
- Automation Builder 2.2.1 or newer

1.3 Overview



n = 150 (PM5630)
n = 250 (PM5650
n = 600 (PM5670/75)

2 Using line Mode

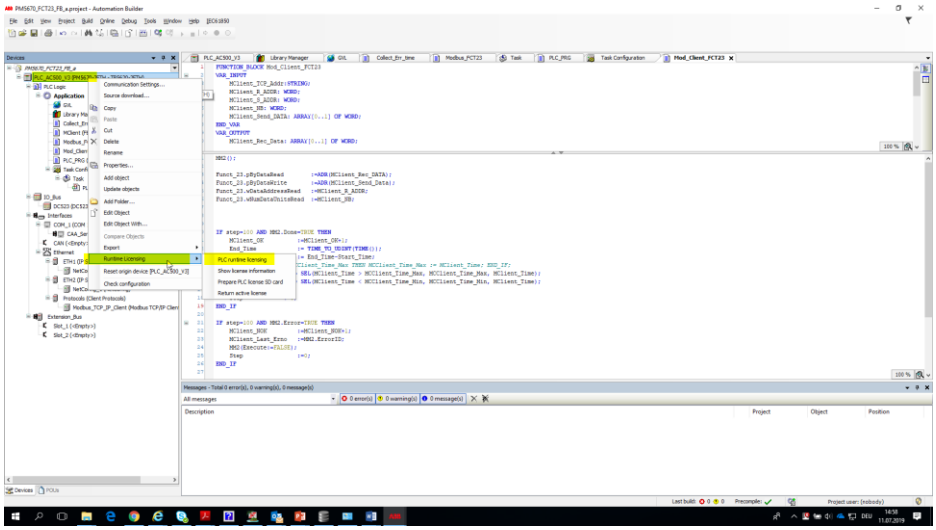
Without using the Line Mode, the AC500 V3 PLC's support the following number of Modbus Sockets

PM5630	30
PM5650	50
PM5670	120

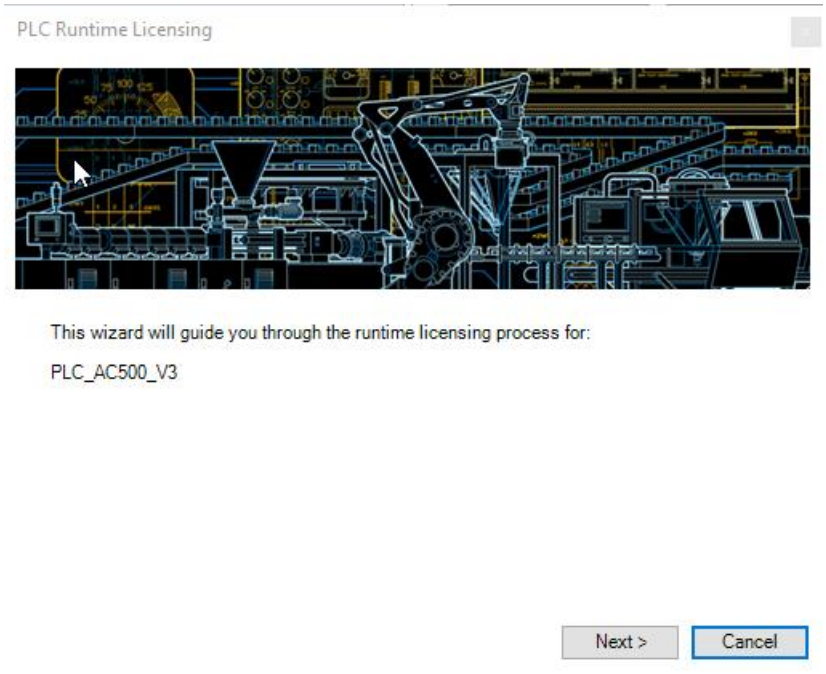
Using an additional Runtime License for Line Mode multiplies this Number of Sockets with Factor 5.

2.1 Adding Runtime License

In Automation Builder Project Right Mouse Click to the PLC Node

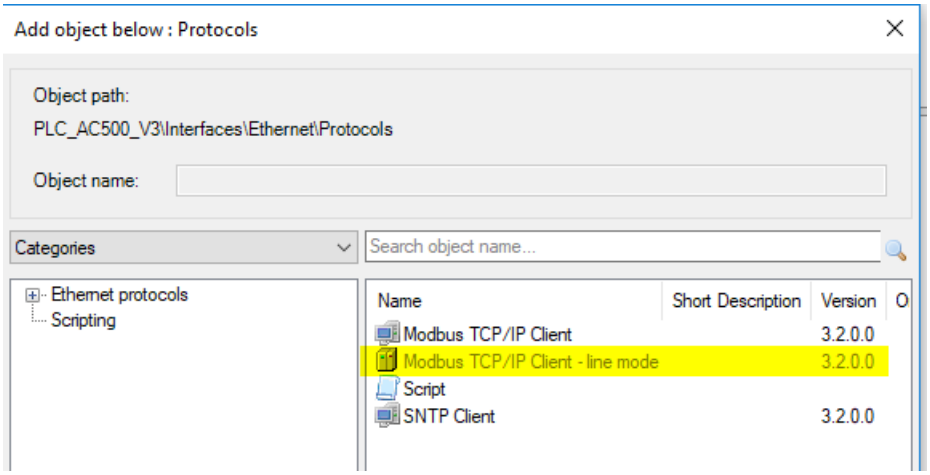


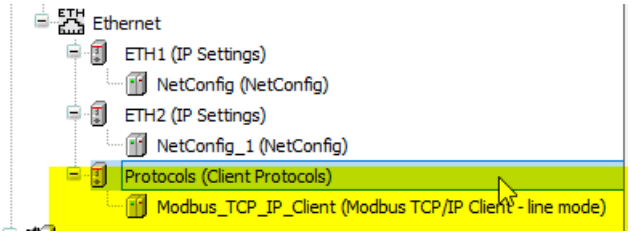
Choose PLC Runtime Licensing and you will be guided to the Installation/Registration process



2.2 Configuration in Automation Builder

After successfully activating the License please add “Modbus TCP/IP Client line mode below protocols



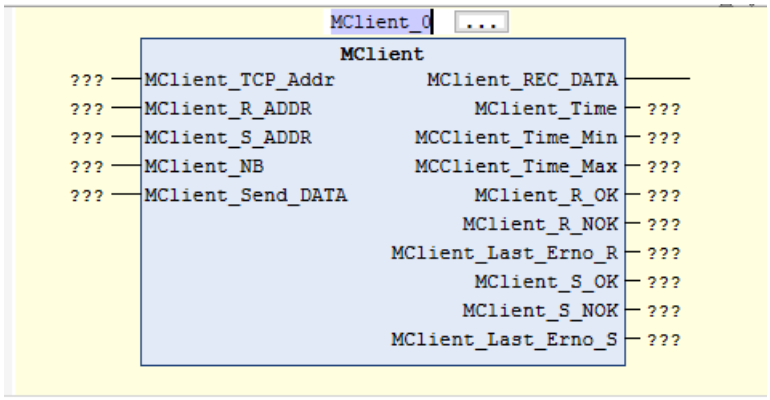


Now the code can be implemented

2.3 Setting up Modbus Client FB

Mostly in such projects there are a lot of Modbus TCP servers with similar functionalities. Therefor it makes sense to create a FB with this functionality and later one you call instances of this FB.

2.3.1 FB with Function Codes 3 (Read) and 16 (Write)

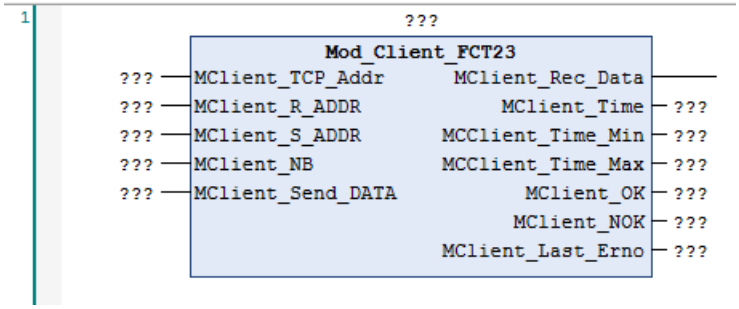



```
FUNCTION_BLOCK MClient
VAR_INPUT
    MClient_TCP_Addr:DWORD;
    MClient_R_ADDR: WORD;
    MClient_S_ADDR: WORD;
    MClient_NB: WORD;
    MClient_Send_DATA: ARRAY[0..1] OF WORD;
END_VAR
VAR_OUTPUT
    MClient_REC_DATA: ARRAY [0..1] OF WORD;
    MClient_Time: UDINT;
    MCClient_Time_Min: UDINT := 16#FFFFFFFF;
    MCClient_Time_Max: UDINT := 0;
    MClient_R_OK:WORD;
    MClient_R_NOK: WORD;
    MClient_Last_Erno_R: AC500_ModbusTCP.ERROR_ID;
    MClient_S_OK:WORD;
    MClient_S_NOK: WORD;
    MClient_Last_Erno_S: AC500_ModbusTCP.ERROR_ID;
END_VAR
VAR
    Step:INT;
    MM:ModTcpMast2;
    Start_Time: UDINT;
    End_Time: UDINT;
    Rec_data: ARRAY [0..99] OF WORD;
    Send_data: ARRAY [0..99] OF WORD;
    MCClient_N: WORD;
END_VAR
```

```
MM();              // call Modbus master with current inputs

IF step=0 AND MM.Busy=FALSE THEN    // start read data from Modbus TCP server
    MM(Execute:=TRUE, Eth:=1, IPAdr:=MClient_TCP_Addr, UnitID:=0, Fct:=3, Addr:=MClient_R_ADDR, Nb:=MClient_NB, Data:=ADR(Rec_data), RespTimeout:=1000,
    KeepAlive:=3000, ByteOrder:=AC500_ModbusTcp.CAA.ENDIANESS.BIG, Port:=502, ResetOnClose:=FALSE);
    Start_Time := TIME_TO_UDINT(TIME());            // read current time in ms
    Step:=100;                                      // go to "wait read ready"
END_IF

IF step=100 AND MM.DONE=TRUE THEN   // wait for ready read data -> success
    MClient_REC_DATA[0] :=Rec_data[0];
    MClient_REC_DATA[1] :=Rec_data[1];
    MClient_R_OK        :=MClient_R_OK+1;
    MM(EXECUTE:=FALSE);
    Step                :=200;                      // go to "write data"
END_IF

IF step=100 AND MM.ERROR=TRUE THEN  // wait for ready read data -> fail
    MClient_R_NOK       :=MClient_R_NOK+1;
    MClient_Last_Erno_R :=MM.ErrorID;
    MM(EXECUTE:=FALSE);
    Step                :=0;                        // go to start new cycle - "read data"
END_IF
```

```
IF step=200 AND MM.Busy=FALSE THEN                // start "write data"
    Send_data[0] := MClient_Send_DATA[0];
    Send_data[1] := MClient_Send_DATA[1];
    MM(Execute:=TRUE, Eth:=1, IPAdr:=MClient_TCP_Addr, UnitID:=0, Fct:=16, Addr:=MClient_S_ADDR, Nb:=MClient_NB,    Data:=ADR(Send_data), RespTimeout:=1000,
    KeepAlive:=3000, ByteOrder:=AC500_ModbusTcp.CAA.ENDIANESS.BIG, Port:=502, ResetOnClose:=FALSE);
    Step                :=300;
END_IF

IF step=300 AND MM.DONE=TRUE THEN   // wait for ready write data -> success
    MClient_S_OK        := MClient_S_OK+1;
    End_Time            := TIME_TO_UDINT(TIME());
    MClient_Time        := End_Time-Start_Time;
    MCClient_Time_Max := SEL(MClient_Time > MCClient_Time_Max, MCClient_Time_Max, MClient_Time);
    MCClient_Time_Min := SEL(MClient_Time < MCClient_Time_Min, MCClient_Time_Min, MClient_Time);
    MM(EXECUTE:=FALSE);
    Step                :=0;
END_IF

IF step=300 AND MM.ERROR=TRUE THEN  // wait for ready write data -> fail
    MClient_S_NOK       :=MClient_S_NOK+1;
    MClient_Last_Erno_S :=MM.ErrorID;
    MM(EXECUTE:=FALSE);
    Step                :=200;
END_IF
```

2.3.2 FB with Function Code 23 (Read and Write in one job)

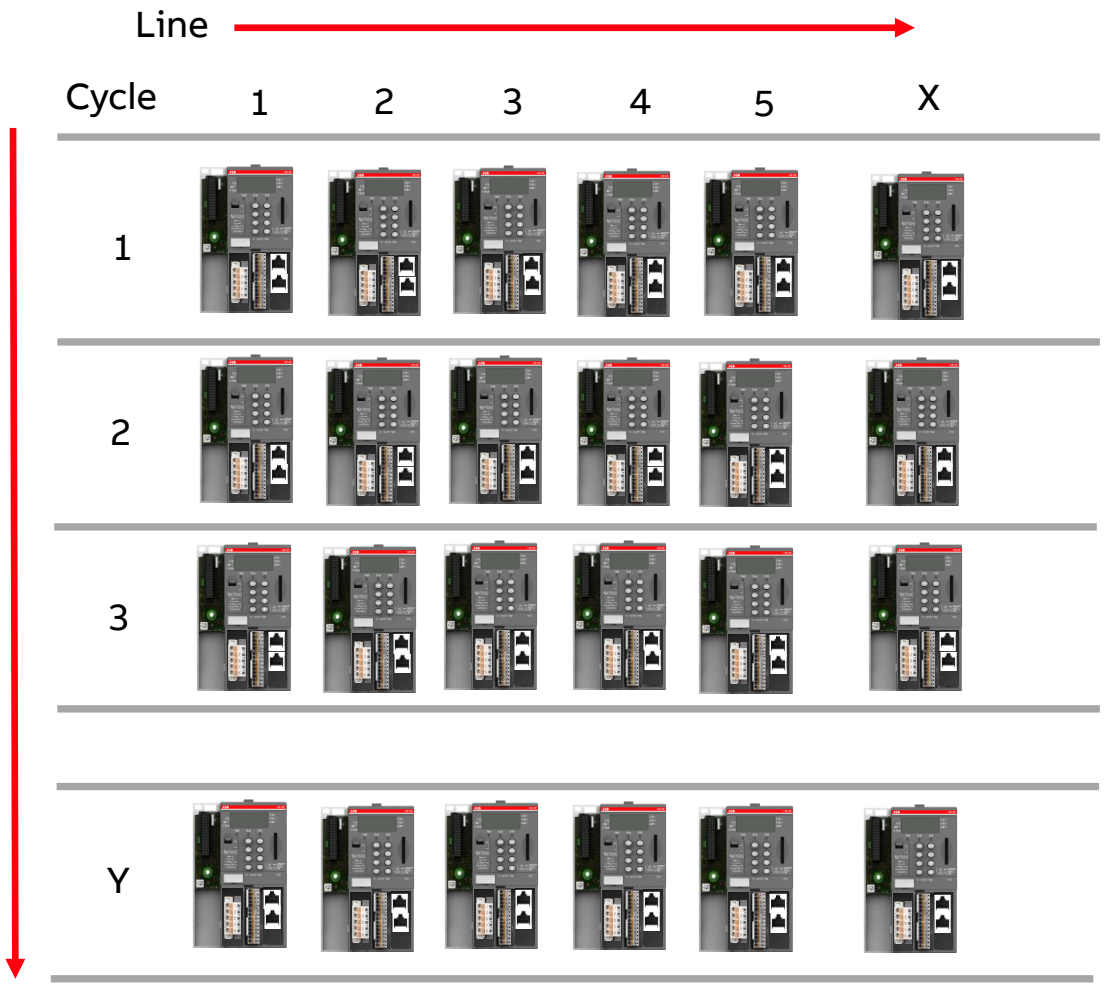



```
FUNCTION_BLOCK Mod_Client_FCT23
VAR_INPUT
    MClient_TCP_Addr:STRING;
    MClient_R_ADDR: WORD;
    MClient_S_ADDR: WORD;
    MClient_NB: WORD;
    MClient_Send_DATA: ARRAY[0..1] OF WORD;
END_VAR
VAR_OUTPUT
    MClient_Rec_Data: ARRAY[0..1] OF WORD;
    MClient_Time: UDINT;
    MCClient_Time_Min: UDINT := 16#FFFFFFFF;
    MCClient_Time_Max: UDINT := 0;
    MClient_OK:WORD;
    MClient_NOK: WORD;
    MClient_Last_Erno: AC500_ModbusTCP.ERROR_ID;
END_VAR
VAR
    MM2: ModTcpMast2;
    Funct_23: ETH_MOD_FCT23_TYPE;
    Start_Time: UDINT;
    step: INT;
    End_Time: UDINT;
END_VAR
```

```
MM2();

Funct_23.pByDataRead        :=ADR(MClient_Rec_DATA);
Funct_23.pByDataWrite       :=ADR(MClient_Send_Data);
Funct_23.wDataAddressRead   :=MClient_R_ADDR;
Funct_23.wNumDataUnitsRead  :=MClient_NB;



IF step=100 AND MM2.Done=TRUE THEN
    MClient_OK          :=MClient_OK+1;
    End_Time            := TIME_TO_UDINT(TIME());
    MClient_Time        := End_Time-Start_Time;
    MCClient_Time_Max := SEL(MClient_Time > MCClient_Time_Max, MCClient_Time_Max, MClient_Time);
    MCClient_Time_Min := SEL(MClient_Time < MCClient_Time_Min, MCClient_Time_Min, MClient_Time);
    MM2(Execute:=FALSE);
    Step            :=0;
END_IF

IF step=100 AND MM2.Error=TRUE THEN
    MClient_NOK         :=MClient_NOK+1;
    MClient_Last_Erno   :=MM2.ErrorID;
    MM2(Execute:=FALSE);
    Step                :=0;
END_IF

IF step=0 AND NOT MM2.Busy THEN
    MM2(Execute:=TRUE, Eth:=1, IPAdr:=AC500_Ethernet.IP_ADR_STRING_TO_DWORD(MClient_TCP_Addr), UnitID:=0, Fct:=23, Addr:=MClient_S_ADDR, Nb:=MClient_NB,
    Data:=ADR(Funct_23),
    RespTimeout:=1000, KeepAlive:=10000, ByteOrder:=AC500_ModbusTcp.CAA.ENDIANESS.BIG, Port:=502, ResetOnClose:=FALSE);
    Start_Time      := TIME_TO_UDINT(TIME());           // read current time in ms
    Step            :=100;
END_IF
```

2.4 Using a Step chain for calling the instances

Working with a high number of Modbus Client instances (e.g. 240) create a high CPU Load if calling all the instances to the same time. To avoid this high load, we recommend calling the FB's in a step chain. That mean call the first 50 instances, wait a dedicated time (e.g. 20ms) and then call the next number of Instances.



Declaration global or local in IEC program:

Instance (=socket) of POU ModTcpMast2 for each server, e.g.:
fbModMast_11, fbModMast_12, .., fbModMast_1X
fbModMast_21, fbModMast_22, .., fbModMast_2X
..
fbModMast_Y1, fbModMast_Y2, .., fbModMast_YX

Note: set input KeepAlive >= runtime for all Steps 1 .. Y
→ socket should not be closed during runtime of all steps

Call in IEC program – step chain with „Y“ steps:

```
Step1: (* call all POUs of cycle 1 *)
        fbModMast_11(Execute := TRUE);
        fbModMast_12(Execute := TRUE);
        fbModMast_13(Execute := TRUE);
        ..
        fbModMast_1X(Execute := TRUE);
Step2: (* call all POUs of cycle 2 *)
        fbModMast_21(Execute := TRUE);
        fbModMast_22(Execute := TRUE);
        fbModMast_23(Execute := TRUE);
        ..
        fbModMast_2X(Execute := TRUE);
StepY: (* call all POUs of cycle Y *)
```

2.4.1 Declaration of the Instances

```
6       (* Instances of Modbus TCP client in old rack = Rack_1  --> all CI522-MODTCP *)
7       MClient_CI522_1_1: Mod_Client_FCT23;
8       MClient_CI522_2_1: Mod_Client_FCT23;
9       MClient_CI522_3_1: Mod_Client_FCT23;
10      MClient_CI522_4_1: Mod_Client_FCT23;
11      MClient_CI522_5_1: Mod_Client_FCT23;
12      MClient_CI522_6_1: Mod_Client_FCT23;
```

```
263     MClient_CI522_34_3: Mod_Client_FCT23;
264     MClient_CI522_35_3: Mod_Client_FCT23;
265
266     MBP: INT;
267     Start_MB: UDINT;
268     Act_Time:UDINT;
269 END_VAR
```

2.4.2 Call of the Instances

Start with the first instances

```
IF MBP=0 THEN
    Start_MB := TIME_TO_UDINT(TIME());          // read current time in ms

(* CI522-MODTCP Server - Rack_1 20x *)

MClient_CI522_1_1(MClient_TCP_Addr:='192.168.0.120',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_2_1(MClient_TCP_Addr:='192.168.0.121',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_3_1(MClient_TCP_Addr:='192.168.0.122',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_4_1(MClient_TCP_Addr:='192.168.0.123',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_5_1(MClient_TCP_Addr:='192.168.0.124',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_6_1(MClient_TCP_Addr:='192.168.0.125',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_7_1(MClient_TCP_Addr:='192.168.0.126',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_8_1(MClient_TCP_Addr:='192.168.0.127',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_9_1(MClient_TCP_Addr:='192.168.0.128',    MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_10_1(MClient_TCP_Addr:='192.168.0.129',   MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
```

```
MClient_eCo_16_1(MClient_TCP_Addr:='192.168.09.116',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_17_1(MClient_TCP_Addr:='192.168.09.117',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_18_1(MClient_TCP_Addr:='192.168.09.118',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_19_1(MClient_TCP_Addr:='192.168.09.119',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_20_1(MClient_TCP_Addr:='192.168.09.120',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);

    MBP :=100;
    END_IF

    IF MBP=100 AND Act_Time>=Start_MB+20 THEN

MClient_eCo_21_1(MClient_TCP_Addr:='192.168.09.121',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_22_1(MClient_TCP_Addr:='192.168.09.122',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_23_1(MClient_TCP_Addr:='192.168.09.123',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_24_1(MClient_TCP_Addr:='192.168.09.124',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_25_1(MClient_TCP_Addr:='192.168.09.125',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_26_1(MClient_TCP_Addr:='192.168.09.126',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_27_1(MClient_TCP_Addr:='192.168.09.127',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
```

After calling the first instances wait for 20ms and call the next

```
MClient_eCo_16_1(MClient_TCP_Addr:='192.168.09.116',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_17_1(MClient_TCP_Addr:='192.168.09.117',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_18_1(MClient_TCP_Addr:='192.168.09.118',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_19_1(MClient_TCP_Addr:='192.168.09.119',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_20_1(MClient_TCP_Addr:='192.168.09.120',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);

    MBP :=100;
    END_IF

    IF MBP=100 AND Act_Time>=Start_MB+20 THEN

MClient_eCo_21_1(MClient_TCP_Addr:='192.168.09.121',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_22_1(MClient_TCP_Addr:='192.168.09.122',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_23_1(MClient_TCP_Addr:='192.168.09.123',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_24_1(MClient_TCP_Addr:='192.168.09.124',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_25_1(MClient_TCP_Addr:='192.168.09.125',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_26_1(MClient_TCP_Addr:='192.168.09.126',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_eCo_27_1(MClient_TCP_Addr:='192.168.09.127',    MClient_R_ADDR:=16#0, MClient_S_ADDR:=16#100, MClient_NB:=1, MClient_Send_DATA:=Send_data);
```

```
MClient_CI522_31_3(MClient_TCP_Addr:='192.168.2.191',   MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_32_3(MClient_TCP_Addr:='192.168.2.192',   MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_33_3(MClient_TCP_Addr:='192.168.2.193',   MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_34_3(MClient_TCP_Addr:='192.168.2.194',   MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);
MClient_CI522_35_3(MClient_TCP_Addr:='192.168.2.195',   MClient_R_ADDR:=16#1000, MClient_S_ADDR:=16#2000, MClient_NB:=1, MClient_Send_DATA:=Send_data);

    MBP :=0;
    END_IF
```

Go back to Start

ABB Automation Products GmbH
Eppelheimer Straße 82
69123 Heidelberg, Germany
Phone: +49 62 21 701 1444
Fax: +49 62 21 701 1382
E-Mail: plc.support@de.abb.com
www.abb.com/plc