

APPLICATION NOTE

AC500 LINE MODE WITH AC500 V3 CPU'S WORKING WITH A HUGE NUMBER OF MODBUS SOCKETS AT SAME TIME



Contents

1	Introduction	3
1.1	Scope of the document	3
1.2	Compatibility	3
1.3	Overview	3
2	Using line Mode.....	4
2.1	Adding Runtime License.....	4
2.2	Configuration in Automation Builder.....	5
2.3	Setting up Modbus Client FB	5
2.3.1	FB with Function Codes 3 (Read) and 16 (Write).....	6
2.3.2	FB with Function Code 23 (Read and Write in one job)	7
2.4	Using a Step chain for calling the instances.....	8
2.4.1	Declaration of the Instances	9
2.4.2	Call of the Instances	9

1 Introduction

1.1 Scope of the document

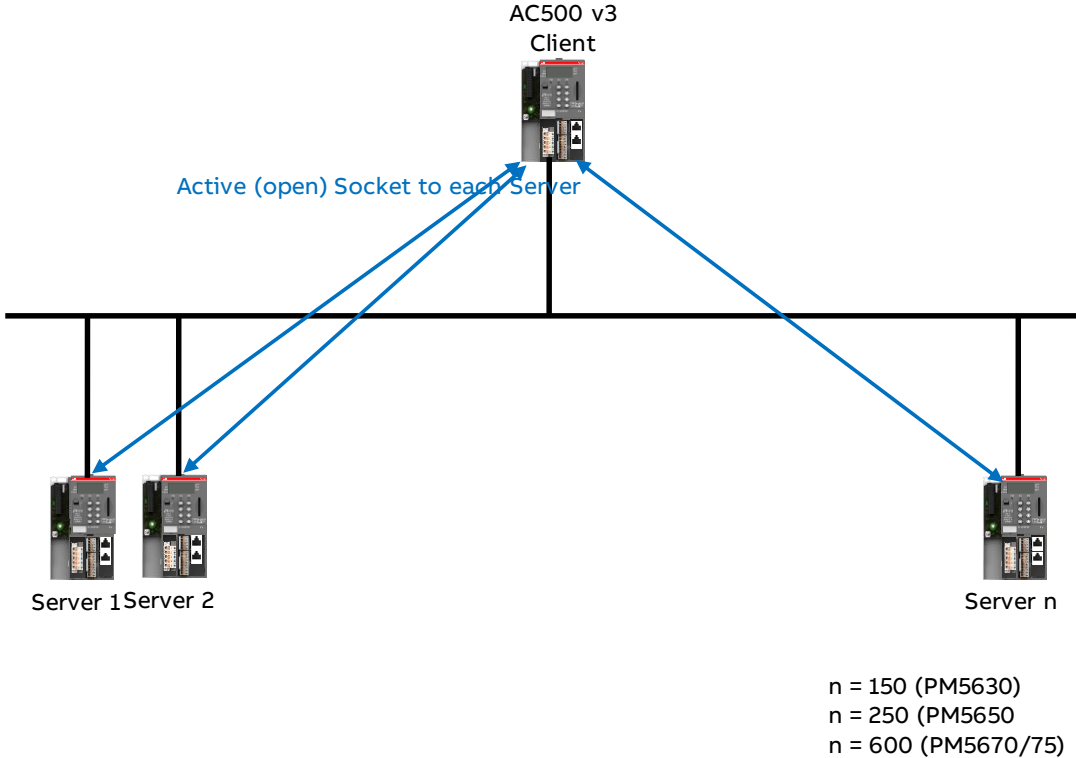
Infrastructure Projects like Tunnel need very often a very high number of Modbus Connections at the same time. To fulfill this Requirement AC500 V3 has the opportunity to increase the number of Modbus Sockets with an additional Runtime License "Line Mode".

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



2 Using line Mode

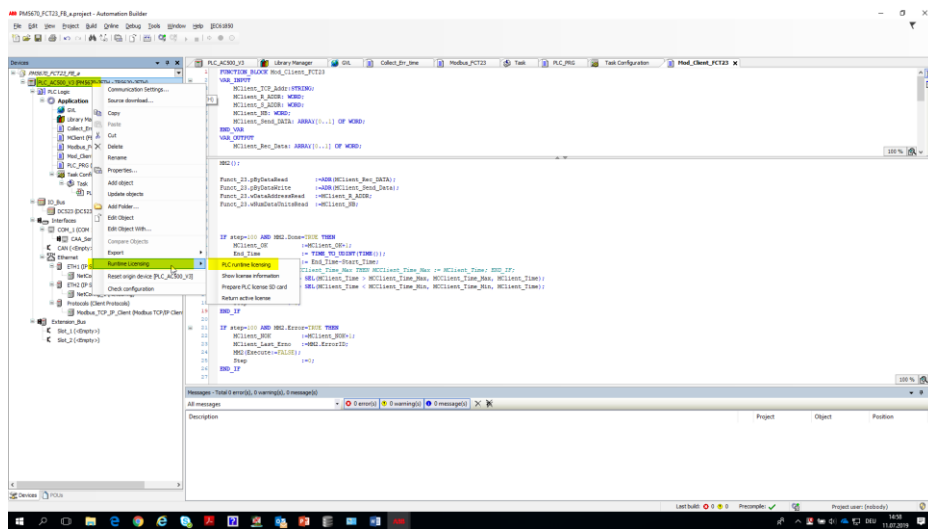
Without using the Line Mode, the AC500 V3 PLC's support the following number of Modbus Sockets

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

PLC Runtime Licensing



This wizard will guide you through the runtime licensing process for:

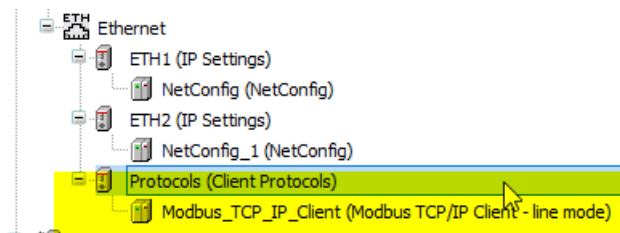
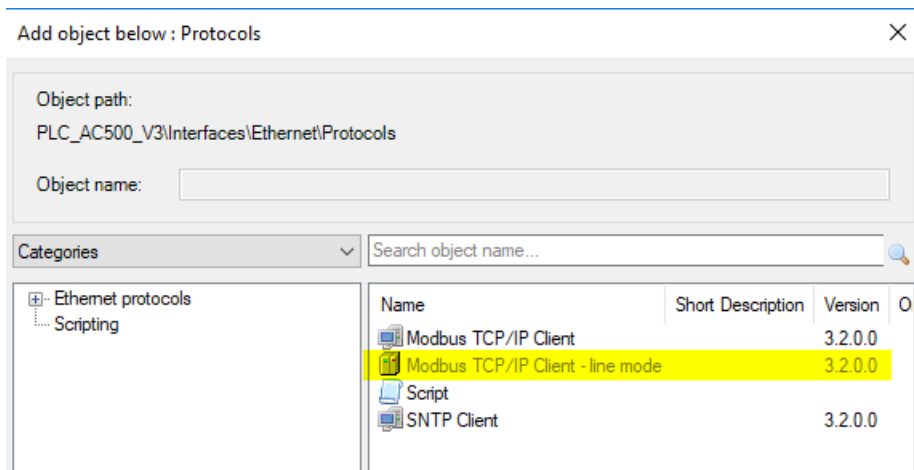
PLC_AC500_V3

Next >

Cancel

2.2 Configuration in Automation Builder

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

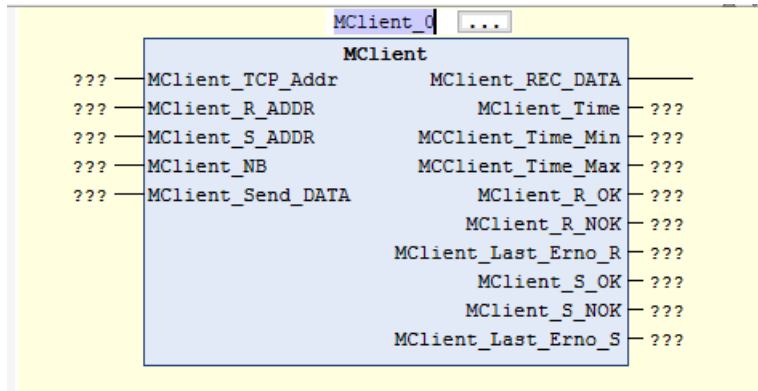


Now the code can be implemented

2.3 Setting up Modbus Client FB

Mostly in such projects there are a lot of Modbus TCP servers with similar functionalities. Therefore 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)



```

1  FUNCTION_BLOCK MClient
2  VAR_INPUT
3      MClient_TCP_Addr:DWORD;
4      MClient_R_ADDR: WORD;
5      MClient_S_ADDR: WORD;
6      MClient_NB: WORD;
7      MClient_Send_DATA: ARRAY[0..1] OF WORD;
8  END_VAR
9  VAR_OUTPUT
10     MClient_REC_DATA: ARRAY [0..1] OF WORD;
11     MClient_Time: UDINT;
12     MCClient_Time_Min: UDINT := 16#FFFFFFF;
13     MCClient_Time_Max: UDINT := 0;
14     MClient_R_OK:WORD;
15     MClient_R_NOK: WORD;
16     MClient_Last_Erno_R: AC500_ModbusTCP.ERROR_ID;
17     MClient_S_OK:WORD;
18     MClient_S_NOK: WORD;
19     MClient_Last_Erno_S: AC500_ModbusTCP.ERROR_ID;
20 END_VAR
21 VAR
22     Step:INT;
23     MM:ModTcpMast2;
24     Start_Time: UDINT;
25     End_Time: UDINT;
26     Rec_data: ARRAY [0..99] OF WORD;
27     Send_data: ARRAY [0..99] OF WORD;
28     MCClient_N: WORD;
29 END_VAR

```

```

1  MM(); // call Modbus master with current inputs
2
3  IF step=0 AND MM.Busy=FALSE THEN // start read data from Modbus TCP server
4      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,
5      KeepAlive:=3000, ByteOrder:=AC500_ModbusTcp.CAA.ENDIANESS.BIG, Port:=502, ResetOnClose:=FALSE);
6      Start_Time := TIME_TO_UDINT(TIME()); // read current time in ms
7      Step:=100; // go to "wait read ready"
8  END_IF
9
10 IF step=100 AND MM.DONE=TRUE THEN // wait for ready read data -> success
11     MClient_REC_DATA[0] :=Rec_data[0];
12     MClient_REC_DATA[1] :=Rec_data[1];
13     MClient_R_OK :=MClient_R_OK+1;
14     MM(EXECUTE:=FALSE);
15     Step :=200; // go to "write data"
16 END_IF
17
18 IF step=100 AND MM.ERROR=TRUE THEN // wait for ready read data -> fail
19     MClient_R_NOK :=MClient_R_NOK+1;
20     MClient_Last_Erno_R :=MM.ErrorID;
21     MM(EXECUTE:=FALSE);
22     Step :=0; // go to start new cycle - "read data"
23 END_IF

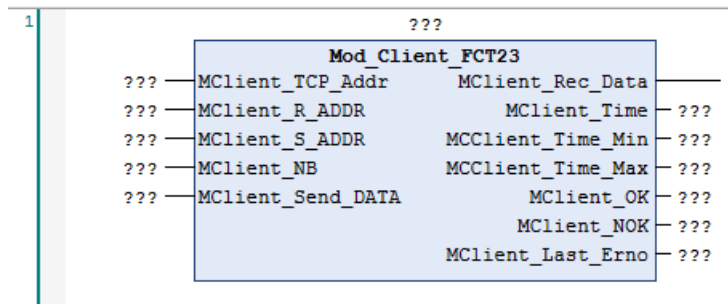
```

```

25 IF step=200 AND MM.Busy=FALSE THEN // start "write data"
26   Send_data[0] := MClient_Send_DATA[0];
27   Send_data[1] := MClient_Send_DATA[1];
28   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,
29     KeepAlive:=3000, ByteOrder:=AC500_ModbusTcp.CAA.ENDIANESS.BIG, Port:=502, ResetOnClose:=FALSE);
30   Step :=300;
31 END_IF
32
33 IF step=300 AND MM.DONE=TRUE THEN // wait for ready write data -> success
34   MClient_S_OK := MClient_S_OK+1;
35   End_Time := TIME_TO_UDINT(TIME());
36   MClient_Time := End_Time-Start_Time;
37   MCClient_Time_Max := SEL(MClient_Time > MCClient_Time_Max, MCClient_Time_Max, MClient_Time);
38   MCClient_Time_Min := SEL(MClient_Time < MCClient_Time_Min, MCClient_Time_Min, MClient_Time);
39   MM(EXECUTE:=FALSE);
40   Step :=0;
41 END_IF
42
43 IF step=300 AND MM.ERROR=TRUE THEN // wait for ready write data -> fail
44   MClient_S_NOK :=MClient_S_NOK+1;
45   MClient_Last_Erno_S :=MM.ErrorID;
46   MM(EXECUTE:=FALSE);
47   Step :=200;
48 END_IF

```

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



```

1 FUNCTION_BLOCK Mod_Client_FCT23
2 VAR_INPUT
3   MClient_TCP_Addr:STRING;
4   MClient_R_ADDR: WORD;
5   MClient_S_ADDR: WORD;
6   MClient_NB: WORD;
7   MClient_Send_DATA: ARRAY[0..1] OF WORD;
8 END_VAR
9 VAR_OUTPUT
10  MClient_Rec_Data: ARRAY[0..1] OF WORD;
11  MClient_Time: UDINT;
12  MCClient_Time_Min: UDINT := 16#FFFFFFFF;
13  MCClient_Time_Max: UDINT := 0;
14  MClient_OK:WORD;
15  MClient_NOK: WORD;
16  MClient_Last_Erno: AC500_ModbusTCP.ERROR_ID;
17 END_VAR
18 VAR
19   MM2: ModTcpMast2;
20   Funct_23: ETH_MOD_FCT23_TYPE;
21   Start_Time: UDINT;
22   step: INT;
23   End_Time: UDINT;
24 END_VAR

```

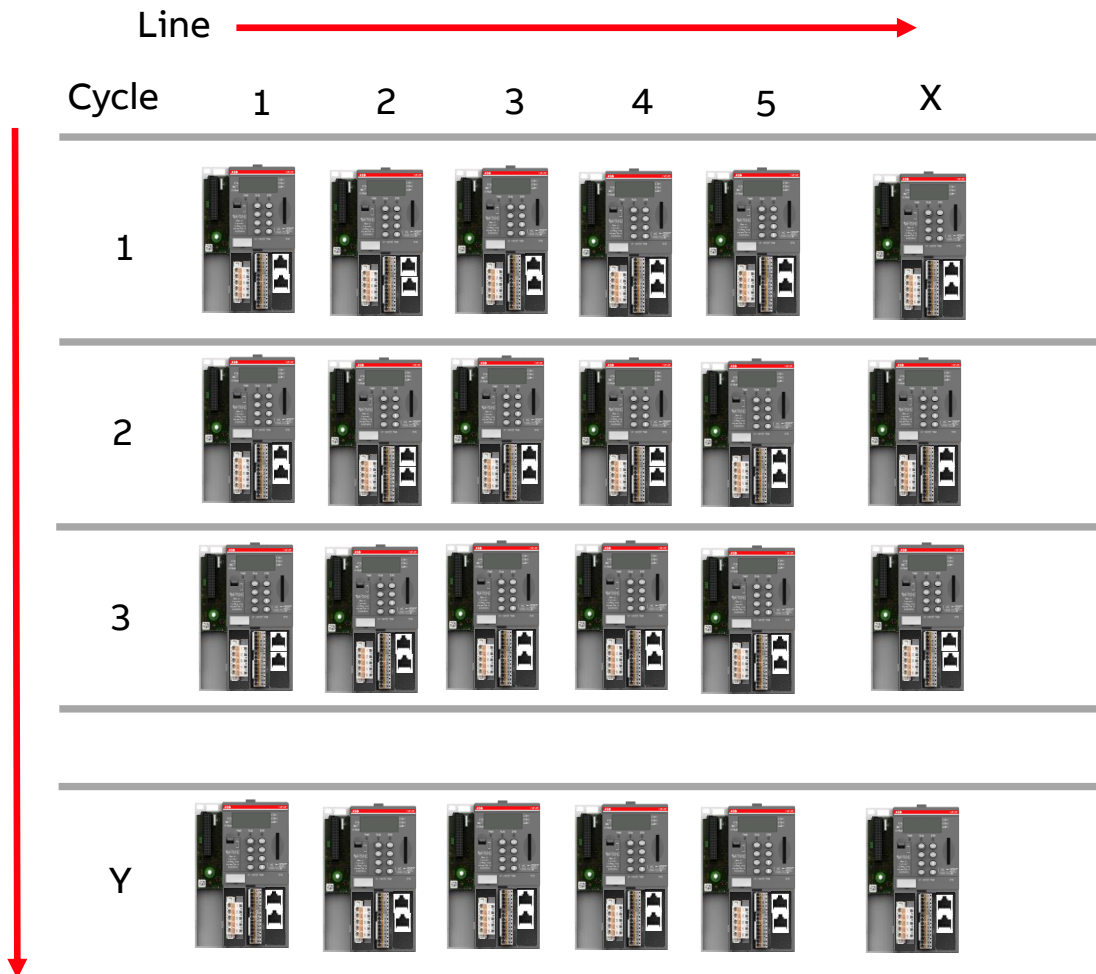
```

1 MM2();
2
3 Funct_23.pByDataRead      :=ADR(MClient_Rec_DATA);
4 Funct_23.pByDataWrite    :=ADR(MClient_Send_Data);
5 Funct_23.wDataAddressRead :=MClient_R_ADDR;
6 Funct_23.wNumDataUnitsRead :=MClient_NB;
7
8
9
10 IF step=100 AND MM2.Done=TRUE THEN
11   MClient_OK              :=MClient_OK+1;
12   End_Time                := TIME_TO_UDINT(TIME());
13   MClient_Time            := End_Time-Start_Time;
14   MClient_Time_Max := SEL(MClient_Time > MClient_Time_Max, MClient_Time_Max, MClient_Time);
15   MClient_Time_Min := SEL(MClient_Time < MClient_Time_Min, MClient_Time_Min, MClient_Time);
16   MM2(Execute:=FALSE);
17   Step                    :=0;
18 END_IF
19
20 IF step=100 AND MM2.Error=TRUE THEN
21   MClient_NOK             :=MClient_NOK+1;
22   MClient_Last_Erno       :=MM2.ErrorID;
23   MM2(Execute:=FALSE);
24   Step                    :=0;
25 END_IF
26
27 IF step=0 AND NOT MM2.Busy THEN
28   MM2(Execute:=TRUE, Eth:=1, IPAdr:=AC500_Ethernet.IP_ADR_STRING_TO_DWORD(MClient_TCP_Adr), UnitID:=0, Fct:=23, Addr:=MClient_S_ADDR, Nb:=MClient_NB,
29   Data:=ADR(Funct_23),
30   RespTimeout:=1000, KeepAlive:=10000, ByteOrder:=AC500_ModbusTcp.CAA.ENDIANESS.BIG, Port:=502, ResetOnClose:=FALSE);
31   Start_Time              := TIME_TO_UDINT(TIME()); // read current time in ms
32   Step                    :=100;
33 END_IF
34

```

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

```

6 IF MBP=0 THEN
7   Start_MB := TIME_TO_UDINT(TIME()); // read current time in ms
8
9 (* CIS22-MODTCP Server - Rack_1 20x *)
10
11 MClient_CIS22_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);
12 MClient_CIS22_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);
13 MClient_CIS22_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);
14 MClient_CIS22_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);
15 MClient_CIS22_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);
16 MClient_CIS22_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);
17 MClient_CIS22_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);
18 MClient_CIS22_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);
19 MClient_CIS22_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);
20 MClient_CIS22_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);
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51 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);
52 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);
53 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);
54 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);
55 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);
56
57 MBP :=100;
58 END_IF
59
60 IF MBP=100 AND Act_Time>=Start_MB+20 THEN
61
62 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);
63 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);
64 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);
65 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);
66 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);
67 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);
68 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);
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136 MClient_CIS22_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);
137 MClient_CIS22_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);
138 MClient_CIS22_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);
139 MClient_CIS22_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);
140 MClient_CIS22_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);
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232 MBP :=0;
233 END_IF
234

```

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

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

We reserve the right to make technical changes or modify the contents of this document without prior notice. With regard to purchase orders, the agreed particulars shall prevail. ABB AG does not accept any responsibility whatsoever for potential errors or possible lack of information in this document.

We reserve all rights in this document and in the subject matter and illustrations contained therein. Any reproduction, disclosure to third parties or utilization of its contents – in whole or in parts – is forbidden without prior written consent of ABB AG.
Copyright© 2019 ABB. All rights reserved