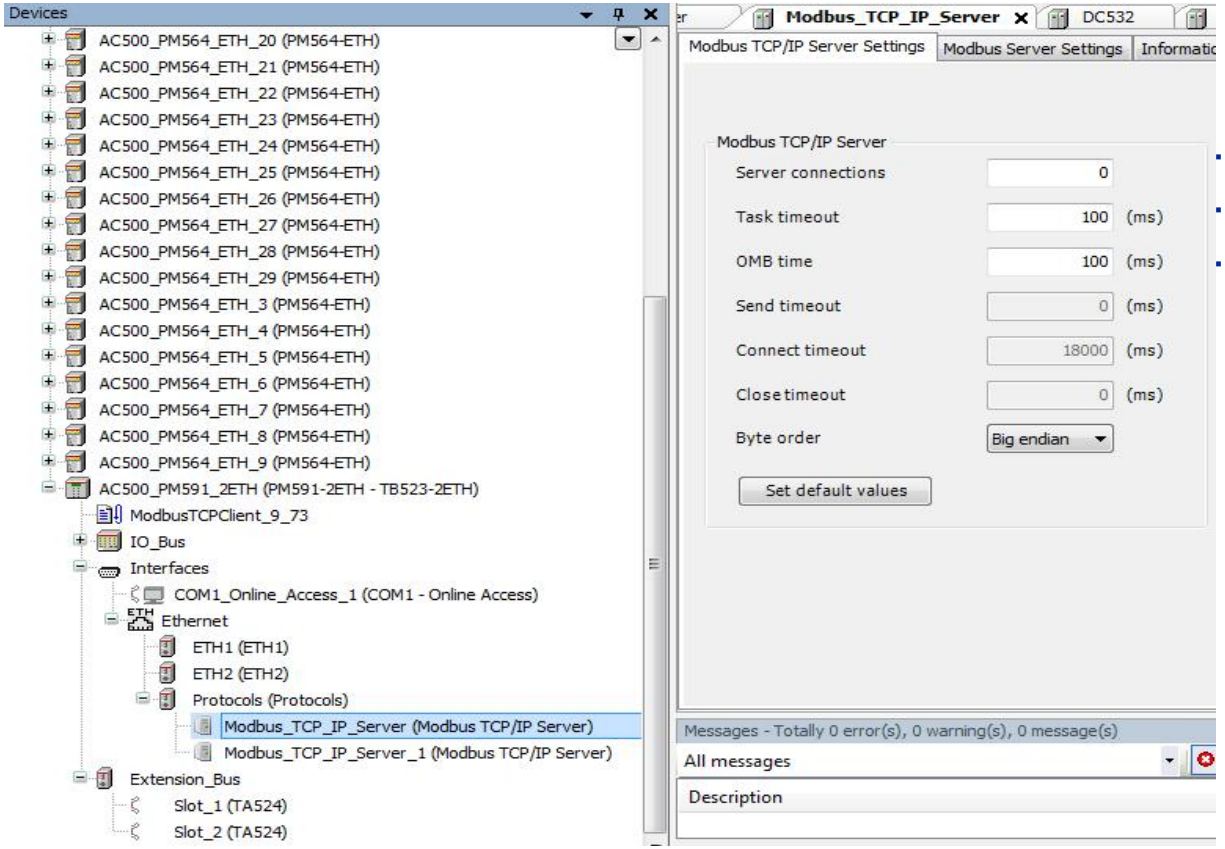
Christian Jambor, Webinar, 2015-03-05

AC500 Ethernet – Modbus TCP
Handling sockets

AC500 Ethernet – Modbus TCP - Handling sockets
Basics: Configuration of Modbus TCP/IP Settings



- Max. number of logical server connections
- Client only: A request is cancelled if no reply
- for Server and Client mode with CPU FW 2.3.x and higher: Determining how long a logical connection remains after a data transfer

AC500 Ethernet – Modbus TCP - Handling sockets
Basics

- Behavior of CPU FW V2.2.x, V2.3.x and FW V2.4.0 with OMB times
  - With CPU FW 2.2.4 the OMB was only **valid for client mode**. In server mode hold CPU the connection.
    - The issue with AC500 FW 2.2.4 in server mode was, in case communication interruptions (e.g. communication via GPRS routers) the connection (socket) sometime does not close. This socket was then blocked for other communication requests. After a few time all sockets could be blocked.
  - With CPU FW 2.3.3 and FW V2.4.0 is the OMB time **valid in server and client mode**.
    - The CPU closes the connection, if there is no data transfer after the OMB time. The benefit is, that the closed sockets are then available for new communication requests.

AC500 Ethernet – Modbus TCP - Handling sockets
Basics

- Factors, which influence the function
  - Number of nodes (Modbus TCP servers)
  - Needed date freshening cycle
  - Used CPU and/or coupler (number sockets)
  - CPU task time
  - CPU Load (Priority of the communication)
  - Reaction times of the servers (Modbus parameter, etc.)

As a result: Procedure for implementation of ModbusTCP client program.

AC500 Ethernet – Modbus TCP - Handling sockets
Basics

- Basic rules
  - If there are less servers than available sockets of the used Modbus TCP client, then one FB ETH_MOD_MAST is employed for every connection.
    - The OMB time of the client must be larger, as the response time of the servers.
    - The OMB time of the server must be larger, as the polling cycle of the client.
  - If there are more servers than available sockets of the used Modbus TCP client, then the FBs ETH_MOD_MAST mast be employed for the communication with several servers.
    - The OMB-time of the client must be small, so that the socket is closed fast and can be employed again for new connection.
    - The control of the communication in the Modbus TCP client program is extreme necessary.

AC500 Ethernet – Modbus TCP - Handling sockets
Basics

CPU and/or coupler type and number of sockets for Modbus TCP:

	PM554-ETH	PM556-ETH	PM564-ETH	PM573-ETH	PM583-ETH	PM590-ETH	PM591-ETH	PM591-2ETH	PM592-ETH	PM595-4ETH-M	PM595-4ETH-F	CM577-ETH	CM597-ETH
till CPU FW V2.3.x	12	-	12	12	12	12	12	-	12	-	-	12	12
CPU FW V2.4.0	12	12	12	12	12	12	12	24	12	24	24	12	12

AC500 Ethernet – Modbus TCP - Handling sockets
Example program: Conditions and targets

- Number of nodes: 49 Modbus TCP servers, which are free available in our Lab
  - rack with 30 PM564-ETH and rack with 19 PM581-ETH
- Date freshening cycle: So quick as possible
- Used CPU: PM591-2ETH with 24 sockets for Modbus TCP
- CPU task time: 10ms
- CPU Load: Observe cpuload in the different tests
- Parameter of the servers:
  - PM564-ETH with 4 server sockets and OMB time 100 ms,
  - PM581-ETH with 9 server sockets
- Further targets:
  - Loss or missing servers may disturb not the date freshening rate.
  - The example should be easy adapt and expand for other applications.

AC500 Ethernet – Modbus TCP - Handling sockets
Example program: Conditions and targets

- rack with

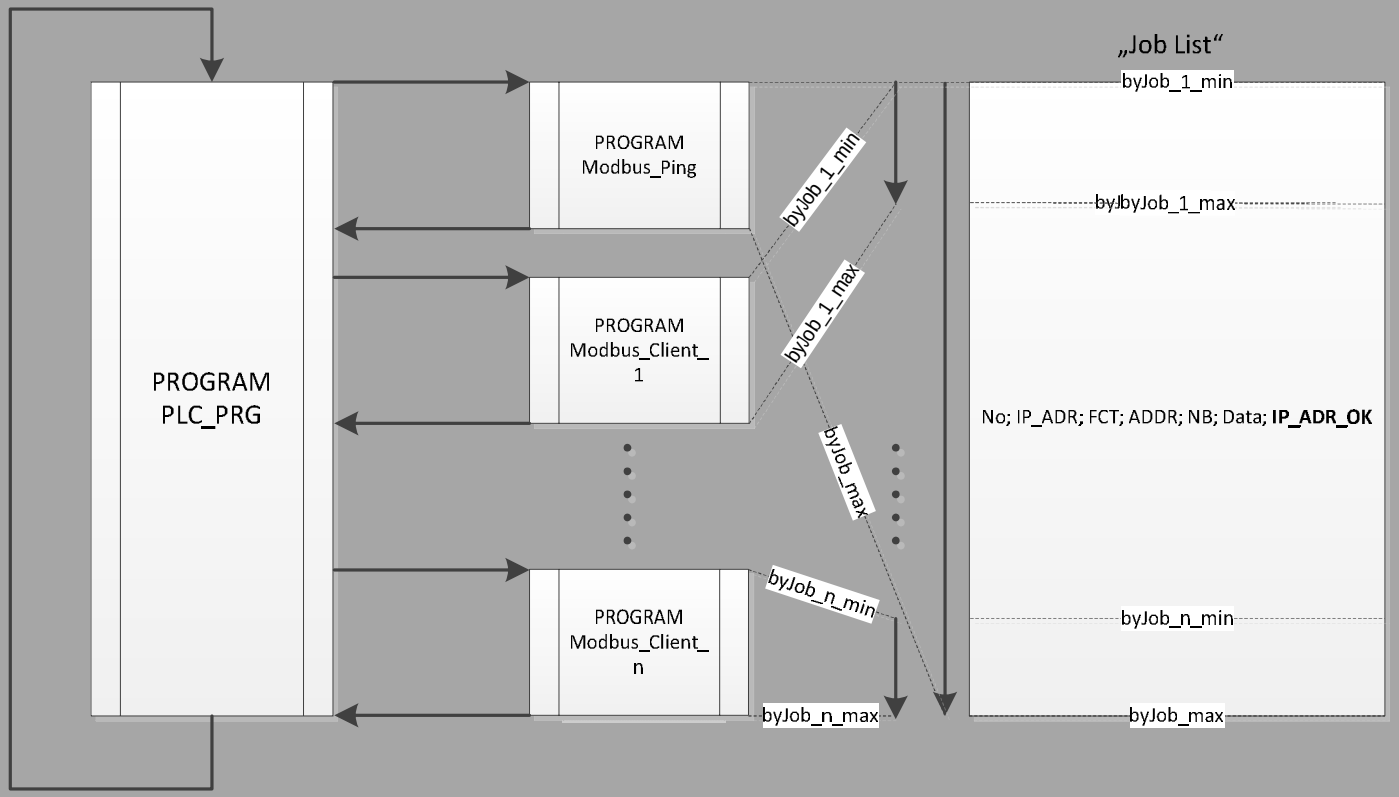
30 PM564-ETH

- rack with

19 PM581-ETH

AC500 Ethernet – Modbus TCP - Handling sockets

Example program: Procedure of Modbus TCP client program

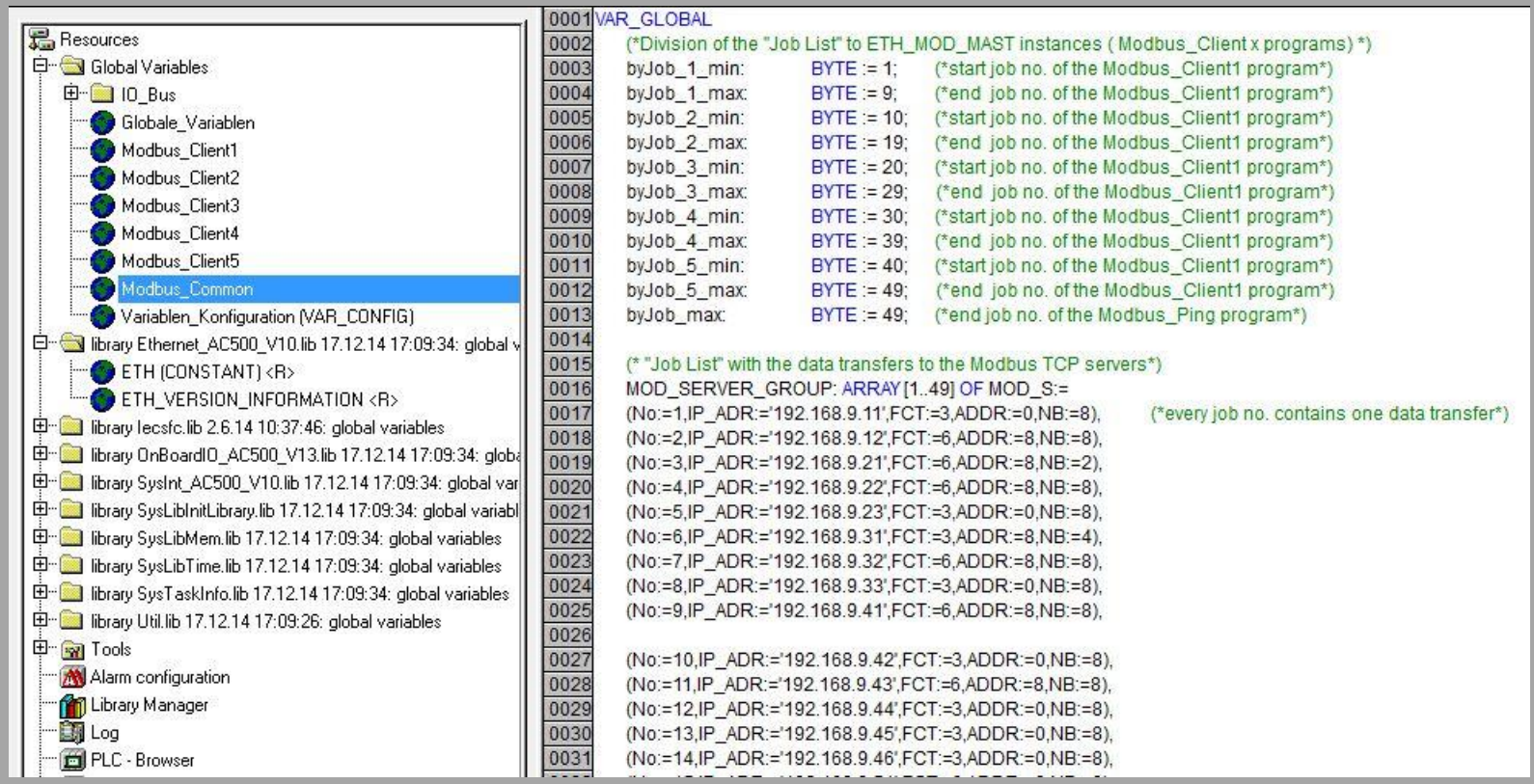


- All of the Modbus TCP data transfers are in a global "Job List".
- Modbus_Ping checks if all nodes in the "Job list" are available.
- Modbus_Client_x executes cyclically the jobs according its section of the "Job List". The data transfer to the node will only be made, if the server is available.

AC500 Ethernet – Modbus TCP - Handling sockets
Example program: The "Job List"

```
TYPE MOD_S :    (*Information about data transfers to one ModbusTCP server, is used in the "Job List".*)
STRUCT
    No:             BYTE;                   (*job number, *)
    IP_ADR:         STRING (15);            (*IP-address of remote ModbusTCP device*)
    FCT:            BYTE;                   (*Modbus function code*)
    IP_ADR_OK:      BOOL;                   (*IP address available, server replies ping *)
    ADDR:           WORD;                   (*operand/register address in remote ModbusTCP device*)
    NB:             WORD;                   (*Number of data to be sent/received*)
    DATA:           ARRAY [0..7] OF WORD;   (*Data container*)
    COM_OK:         WORD;                   (*auxiliary variable for testing, ModbusTCP client counter OK*)
    COM_ERR:        WORD;                   (*auxiliary variable for testing, ModbusTCP client counter not OK*)
    COM_ERNO:       WORD;                   (*auxiliary variable for testing, ModbusTCP client error no.*)
    PINK_OK:        WORD;                   (*auxiliary variable for testing, ETH_ICMP_PING counter OK*)
    PINK_ERR:       WORD;                   (*auxiliary variable for testing, ETH_ICMP_PING counter not OK*)
    PINK_ERNO:      WORD;                   (*auxiliary variable for testing, ETH_ICMP_PING error no.*)
END_STRUCT
END_TYPE
```

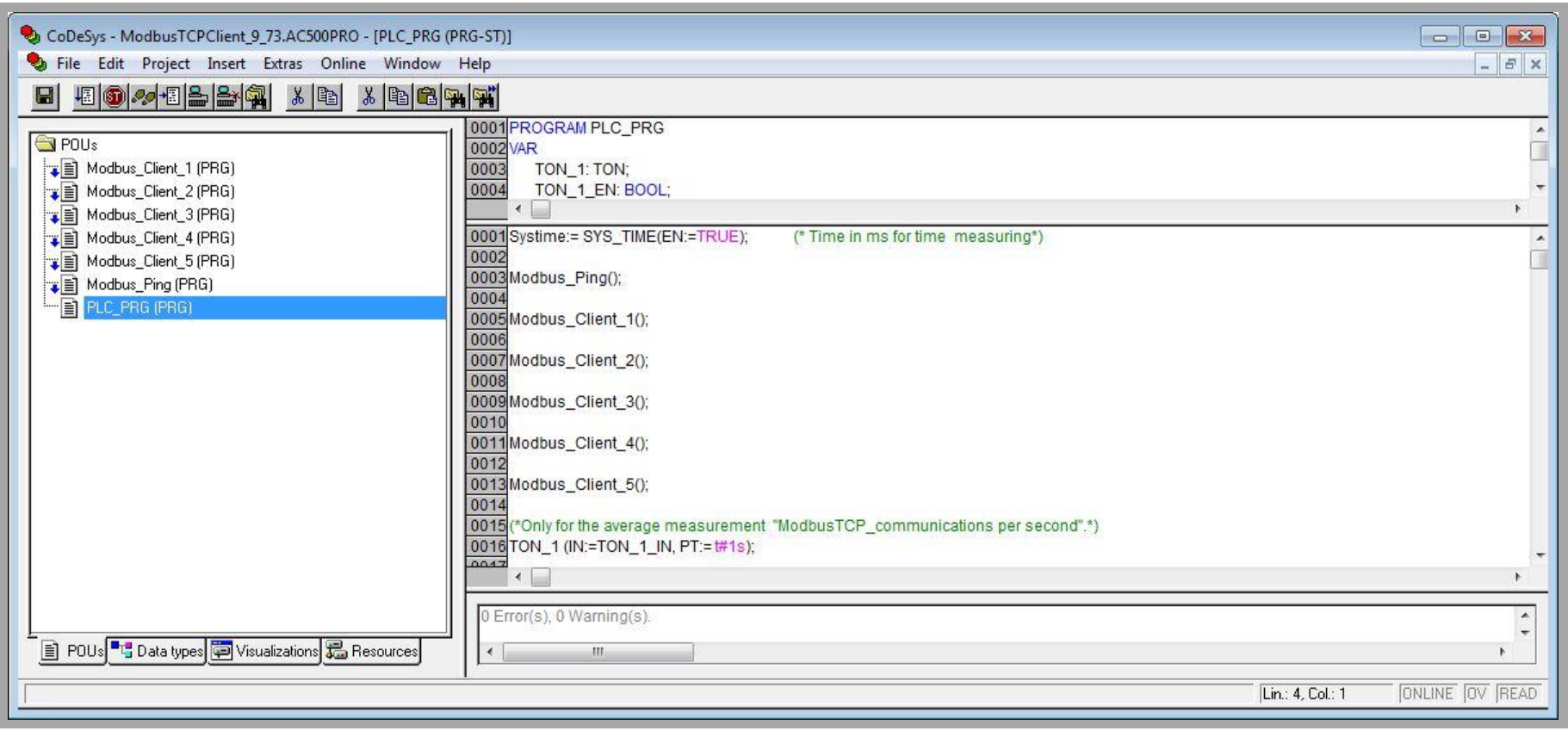
AC500 Ethernet – Modbus TCP - Handling sockets
Example program: Adjustment of the "Job List"




```
0001 VAR_GLOBAL
0002     (*Division of the "Job List" to ETH_MOD_MAST instances ( Modbus_Client x programs) *)
0003     byJob_1_min:        BYTE := 1;     (*start job no. of the Modbus_Client1 program*)
0004     byJob_1_max:        BYTE := 9;     (*end  job no. of the Modbus_Client1 program*)
0005     byJob_2_min:        BYTE := 10;    (*start job no. of the Modbus_Client1 program*)
0006     byJob_2_max:        BYTE := 19;    (*end  job no. of the Modbus_Client1 program*)
0007     byJob_3_min:        BYTE := 20;    (*start job no. of the Modbus_Client1 program*)
0008     byJob_3_max:        BYTE := 29;    (*end  job no. of the Modbus_Client1 program*)
0009     byJob_4_min:        BYTE := 30;    (*start job no. of the Modbus_Client1 program*)
0010     byJob_4_max:        BYTE := 39;    (*end  job no. of the Modbus_Client1 program*)
0011     byJob_5_min:        BYTE := 40;    (*start job no. of the Modbus_Client1 program*)
0012     byJob_5_max:        BYTE := 49;    (*end  job no. of the Modbus_Client1 program*)
0013     byJob_max:          BYTE := 49;    (*end job no. of the Modbus_Ping program*)
0014
0015     (* "Job List" with the data transfers to the Modbus TCP servers*)
0016     MOD_SERVER_GROUP: ARRAY [1..49] OF MOD_S:=
0017     (No:=1,IP_ADR:='192.168.9.11',FCT:=3,ADDR:=0,NB:=8),      (*every job no. contains one data transfer*)
0018     (No:=2,IP_ADR:='192.168.9.12',FCT:=6,ADDR:=8,NB:=8),
0019     (No:=3,IP_ADR:='192.168.9.21',FCT:=6,ADDR:=8,NB:=2),
0020     (No:=4,IP_ADR:='192.168.9.22',FCT:=6,ADDR:=8,NB:=8),
0021     (No:=5,IP_ADR:='192.168.9.23',FCT:=3,ADDR:=0,NB:=8),
0022     (No:=6,IP_ADR:='192.168.9.31',FCT:=3,ADDR:=8,NB:=4),
0023     (No:=7,IP_ADR:='192.168.9.32',FCT:=6,ADDR:=8,NB:=8),
0024     (No:=8,IP_ADR:='192.168.9.33',FCT:=3,ADDR:=0,NB:=8),
0025     (No:=9,IP_ADR:='192.168.9.41',FCT:=6,ADDR:=8,NB:=8),
0026
0027     (No:=10,IP_ADR:='192.168.9.42',FCT:=3,ADDR:=0,NB:=8),
0028     (No:=11,IP_ADR:='192.168.9.43',FCT:=6,ADDR:=8,NB:=8),
0029     (No:=12,IP_ADR:='192.168.9.44',FCT:=3,ADDR:=0,NB:=8),
0030     (No:=13,IP_ADR:='192.168.9.45',FCT:=3,ADDR:=0,NB:=8),
0031     (No:=14,IP_ADR:='192.168.9.46',FCT:=3,ADDR:=0,NB:=8),
```

AC500 Ethernet – Modbus TCP - Handling sockets

Example program: Adjustment for more Modbus_Client_x program



AC500 Ethernet – Modbus TCP - Handling sockets

Example program: Measured values, Visu with instantaneous values

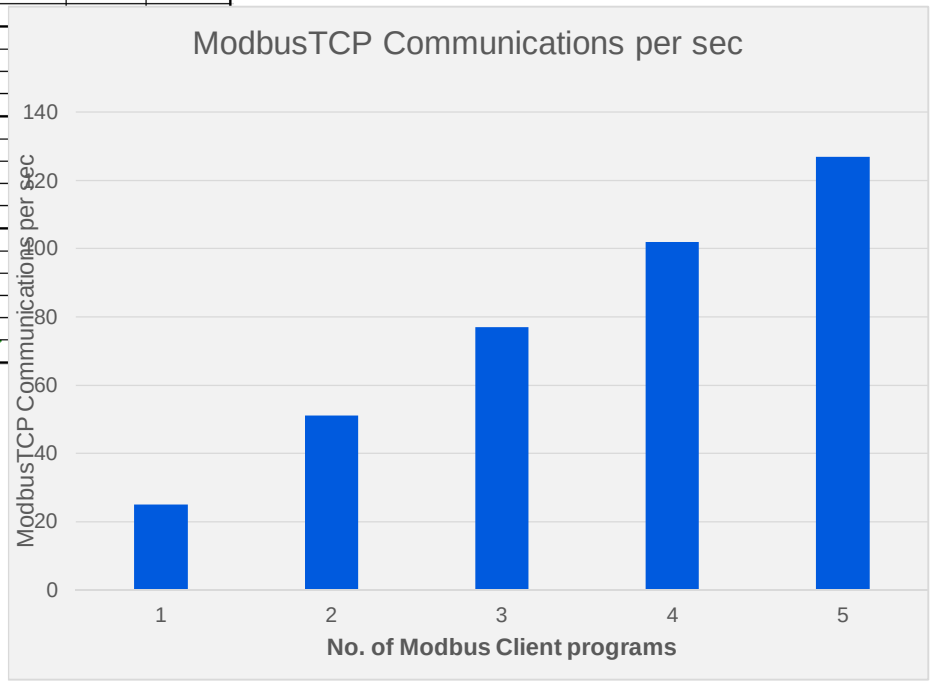
	current Job No.	Average Time / Job
Modbus_Ping	32	20 1/ms
Modbus_Client_1	9	40 1/ms
Modbus_Client_2	16	40 1/ms
Modbus_Client_3	26	40 1/ms
Modbus_Client_4	36	40 1/ms
Modbus_Client_5	46	40 1/ms

ModbusTCP_communications
126 1/s

AC500 Ethernet – Modbus TCP - Handling sockets
Example program: Measured values

No. of Modbus Client	Client CPU FW	Task time	cpuload	Tsk timeout [ms]	OMB time [ms]	Used Sockets 502	ModbusTCP Communications per_sec			Jobs	Job_min	Job_max
1	PM591_2ETH	10 ms	20%	100	100	3 -- 4	25	ETH2	Modbus_Client_5	49	1	49
								ETH1	Modbus_Ping	49		
2	PM591_2ETH	10 ms	24%	100	100	6	50-52	ETH1	Modbus_Client_1	19	1	19
								ETH2	Modbus_Client_5	30	20	49
								ETH1	Modbus_Ping			
3	PM591_2ETH	10 ms	31%	100	100	9	75-78	ETH1	Modbus_Client_1			
								ETH2	Modbus_Client_4			
								ETH2	Modbus_Client_5			
								ETH1	Modbus_Ping			
4	PM591_2ETH	10 ms	31%	100	100	9	100-104	ETH1	Modbus_Client_1			
								ETH2	Modbus_Client_3			
								ETH2	Modbus_Client_4			
								ETH2	Modbus_Client_5			
								ETH1	Modbus_Ping			
5	PM591_2ETH	10 ms	39 - 43%	100	100	14 -- 18	125 -- 129	ETH1	Modbus_Client_1			
								ETH1	Modbus_Client_2			
								ETH2	Modbus_Client_3			
								ETH2	Modbus_Client_4			
								ETH2	Modbus_Client_5			
								ETH1	Modbus_Ping			

Communication rate as a function of no. of Modbus client programs



Power and productivity
for a better world™