Advanced Wind turbine Controller AWC 500

IEC61131-3 programming

DEIF A/S · Frisenborgvej 33 · DK-7800 Skive
Tel.: +45 9614 9614 · Fax: +45 9614 9615
info@deif.com · www.deif.com

Document no.: 4189340738

Revision

Revision	Author	Date	Description
A	SJE	2012-06-26	Initial release
B	LVI	2012-08-14	Frequency input description updated
C	LVI	2012-10-23	CAN telegram debug logging added
D	SJE	2013-06-24	Added IOM5·2 and DIM5·1 module descriptions

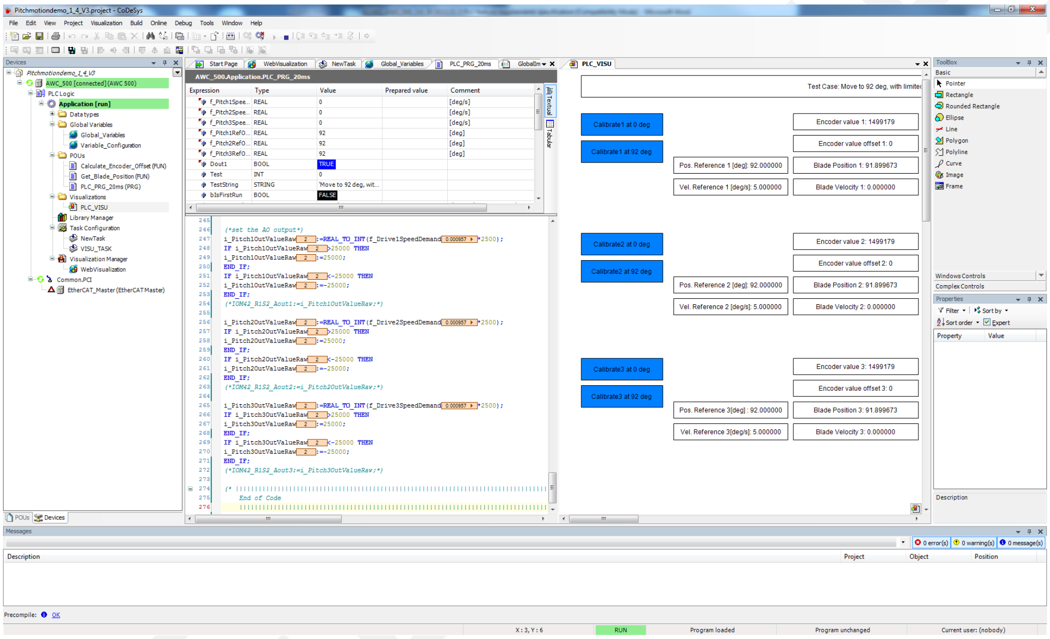
Contents

1 IEC61131-programming guide

The PLC feature is provided by the CoDeSys environment used in other brand PLCs known from the industrial automation. DEIF has chosen to provide a PLC feature based on CoDeSys V3 on its rugged AWC 500 for users convenience.

1.1 Introduction

CoDeSys is the common name for the IEC61131-3 programming system, called CoDeSys IDE. The IDE is installed on your development computer see Figure 1.1.



AWC_500.Application.PLC_PRG_20ms

Expression	Type	Value	Prepared value	Comment
f_Pitch1Spee...	REAL	0		[deg/s]
f_Pitch2Spee...	REAL	0		[deg/s]
f_Pitch3Spee...	REAL	0		[deg/s]
f_Pitch1RefO...	REAL	92		[deg]
f_Pitch2RefO...	REAL	92		[deg]
f_Pitch3RefO...	REAL	92		[deg]
Dout1	BOOL	TRUE		
Test	INT	0		
TestString	STRING	'Move to 92 deg, wit...		
bIsFirstRun	BOOL	FALSE		

```
(*set the AO output*)
i_Pitch1OutValueRaw 2 :=REAL_TO_INT(f_Drive1SpeedDemand 0.000957 *2500);
IF i_Pitch1OutValueRaw 2 >25000 THEN
i_Pitch1OutValueRaw 2 :=25000;
END_IF;
IF i_Pitch1OutValueRaw 2 <-25000 THEN
i_Pitch1OutValueRaw 2 :=-25000;
END_IF;
(*IOM42_R1S2_Aout1:=i_Pitch1OutValueRaw;*)

i_Pitch2OutValueRaw 2 :=REAL_TO_INT(f_Drive2SpeedDemand 0.000957 *2500);
IF i_Pitch2OutValueRaw 2 >25000 THEN
i_Pitch2OutValueRaw 2 :=25000;
END_IF;
IF i_Pitch2OutValueRaw 2 <-25000 THEN
i_Pitch2OutValueRaw 2 :=-25000;
END_IF;
(*IOM42_R1S2_Aout2:=i_Pitch2OutValueRaw;*)

i_Pitch3OutValueRaw 2 :=REAL_TO_INT(f_Drive3SpeedDemand 0.000957 *2500);
IF i_Pitch3OutValueRaw 2 >25000 THEN
i_Pitch3OutValueRaw 2 :=25000;
END_IF;
IF i_Pitch3OutValueRaw 2 <-25000 THEN
i_Pitch3OutValueRaw 2 :=-25000;
END_IF;
(*IOM42_R1S2_Aout3:=i_Pitch3OutValueRaw;*)

(* ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
   End of Code
   ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

Test Case: Move to 92 deg, with limite

Calibrate1 at 0 deg
Calibrate1 at 92 deg
Encoder value 1: 1499179
Encoder value offset 1: 0
Pos. Reference 1 [deg]: 92.000000
Blade Position 1: 91.899673
Vel. Reference 1 [deg/s]: 5.000000
Blade Velocity 1: 0.000000

Calibrate2 at 0 deg
Calibrate2 at 92 deg
Encoder value 2: 1499179
Encoder value offset 2: 0
Pos. Reference 2 [deg]: 92.000000
Blade Position 2: 91.899673
Vel. Reference 2 [deg/s]: 5.000000
Blade Velocity 2: 0.000000

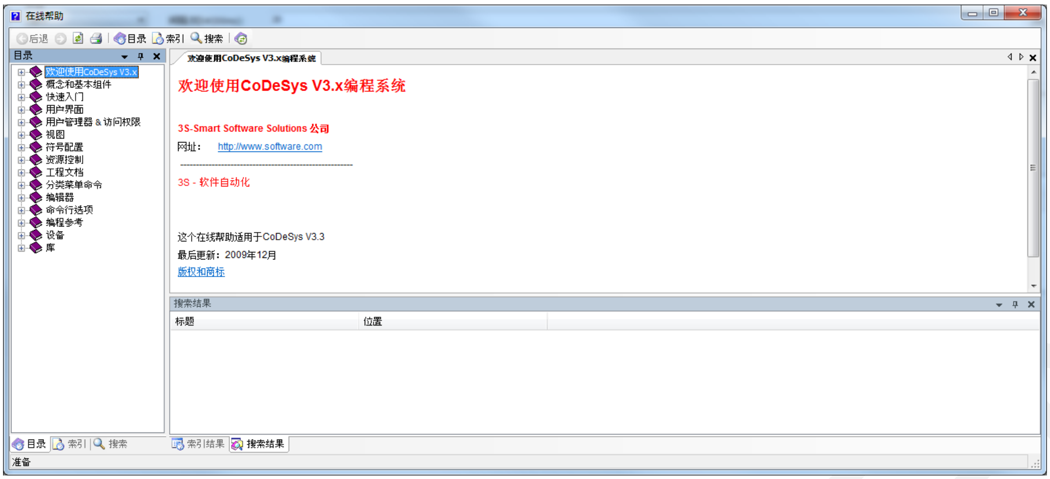
Calibrate3 at 0 deg
Calibrate3 at 92 deg
Encoder value 3: 1499179
Encoder value offset 3: 0
Pos. Reference 3[deg]: 92.000000
Blade Position 3: 91.899673
Vel. Reference 3[deg/s]: 5.000000
Blade Velocity 3: 0.000000

Figure 1.1: The CoDeSysV3 Editor

CoDeSys supports all five programming languages of the IEC 61131-3 programming standard are supported.

- Instruction List
- Sequential Function Chart
- Function Block Diagram
- Structured Text
- Ladder Diagram
- Continuous Function Chart

The entire programming kit including a manual and online assistance is available in English, German and Chinese.



欢迎使用CoDeSys V3.x编程系统

3S-Smart Software Solutions 公司

网址： http://www.software.com

3S - 软件自动化

这个在线帮助适用于CoDeSys V3.3

最后更新：2009年12月

版权和商标

Figure 1.2: Example of the Chinese online help

The following features are included:

- Monitoring of all variables
- Writing and forcing of receipts (sets of variables) into the PLC
- Debugging your complete project (breakpoints, stepping, single cycle, call stack)
- Interrupt-free online changes of POUs and data
- Sampling Trace
- Library management for user defined libraries
- Offline simulation
- Graphic PLC configuration
- OPC Server

The other part the PLC runtime system, called CoDeSys SP (SoftPLC), has DEIF installed on the AWC 500 , so you can run your new or existing programs written with CoDeSys or other CoDeSys based products. When moving from other PLCs.

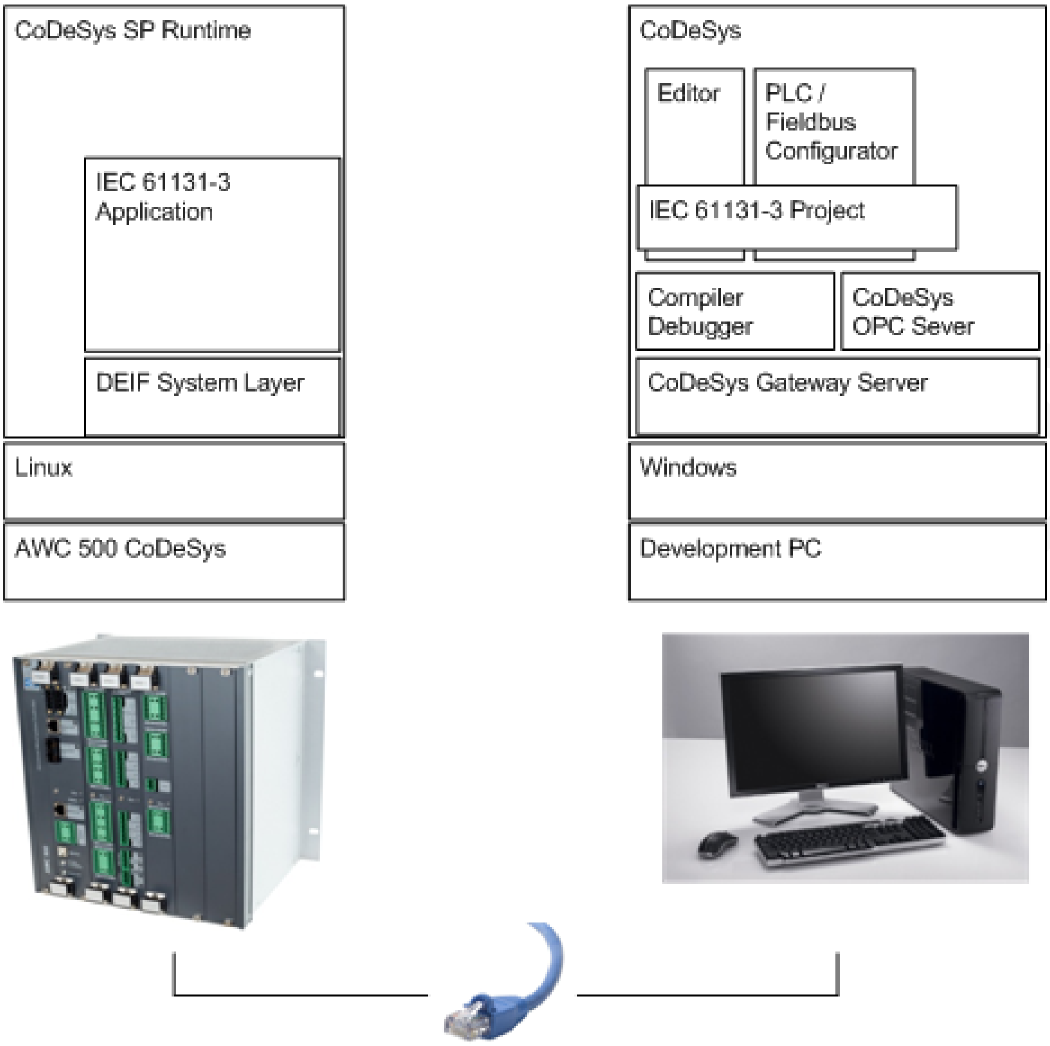
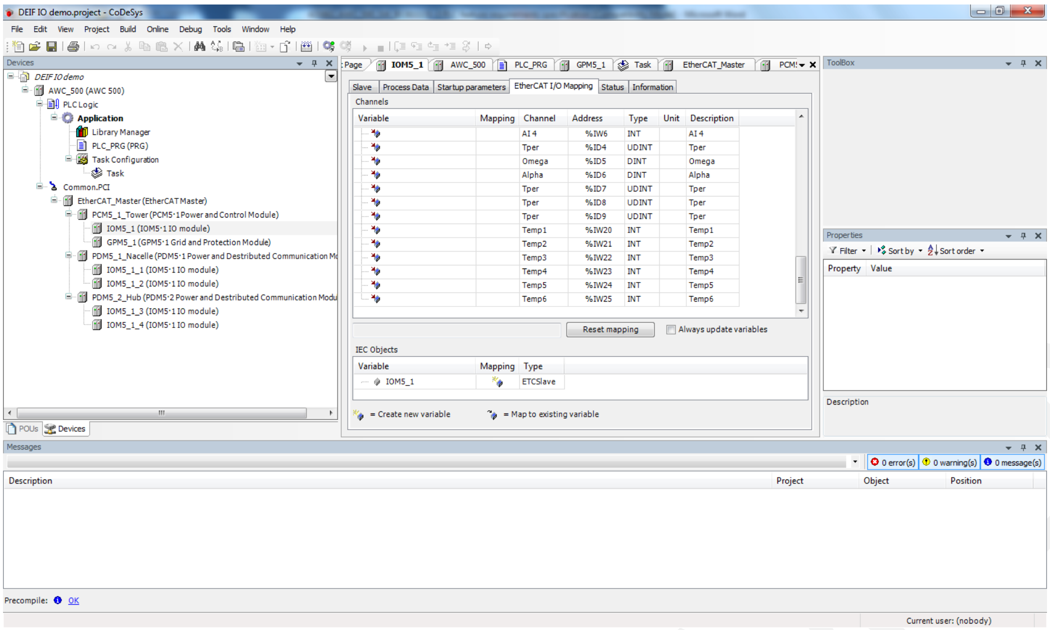


Figure 1.3: DEIF AWC 500 platform with CoDeSys Runtime

Often only small changes related to a transition is expected. Examples on things to change in the transition from a other PLC's to AWC 500 will most likely focus on that DEIF does not have matching library functions, e.g. communication drivers. The HMI is directly compatible other time the entire HMI requires rework. PLC programs written in CoDeSys V2 can be imported in CoDeSys V3, minor conversions are required. These conversion tools are build-in CoDeSys V3 to guide the user through this. DEIF also assist client getting started with using the AWC 500 and audits the clients PLC code. Figure 1.4 displays an example turbine application.


DEIF IO demo.project - CoDeSys

Devices

- DEIF IO demo
  - AWC_500 (AWC 500)
    - PLC Logic
      - Application
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - Task
    - Common.PCI
      - EtherCAT_Master (EtherCAT Master)
        - PCM5_1_Tower (PCM5-1 Power and Control Module)
          - IOM5_1 (IOM5-1 IO module)
          - GPM5_1 (GPM5-1 Grid and Protection Module)
        - PDM5_1_Nacelle (PDM5-1 Power and Destributed Communication Mo
          - IOM5_1_1 (IOM5-1 IO module)
          - IOM5_1_2 (IOM5-1 IO module)
        - PDM5_2_Hub (PDM5-2 Power and Destributed Communication Modu
          - IOM5_1_3 (IOM5-1 IO module)
          - IOM5_1_4 (IOM5-1 IO module)

IOM5_1 — EtherCAT I/O Mapping

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		AI 4	%IW6	INT		AI 4
		Tper	%ID4	UDINT		Tper
		Omega	%ID5	DINT		Omega
		Alpha	%ID6	DINT		Alpha
		Tper	%ID7	UDINT		Tper
		Tper	%ID8	UDINT		Tper
		Tper	%ID9	UDINT		Tper
		Temp1	%IW20	INT		Temp1
		Temp2	%IW21	INT		Temp2
		Temp3	%IW22	INT		Temp3
		Temp4	%IW23	INT		Temp4
		Temp5	%IW24	INT		Temp5
		Temp6	%IW25	INT		Temp6

□ Always update variables

IEC Objects

Variable	Mapping	Type
IOM5_1		ETCSlave

= Create new variable = Map to existing variable

Messages: 0 error(s) 0 warning(s) 0 message(s)

Precompile: OK

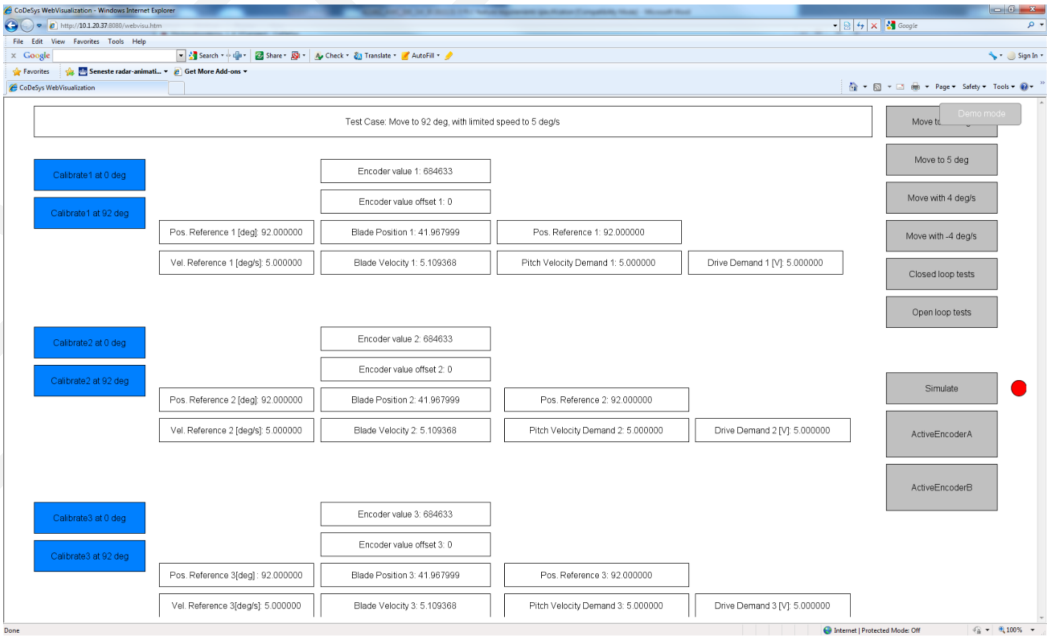
Current user: (nobody)

Figure 1.4: Example configuration for Wind Turbine with Tower base, Nacelle and Hub (via EtherCAT)

AWC 500 freatures:

- Distributed IO configuration matching common turbine configurations
- Automatic startup. The PLC runtime is able to start(restart) without user interaction.
- Boot projects. Possible to make CoDeSys boot projects, that is application projects that start up automatically.
- Web visualisation. CoDeSys web server implementation can be accessed on http:/[ip]:8080/webvisu.htm
- SD card access.
- Retain / persistent variables.

Figure 1.5 presents an Web visualisation example.


CoDeSys WebVisualization - Windows Internet Explorer

http://10.1.20.37:8080/webvisu.htm

Test Case: Move to 92 deg, with limited speed to 5 deg/s

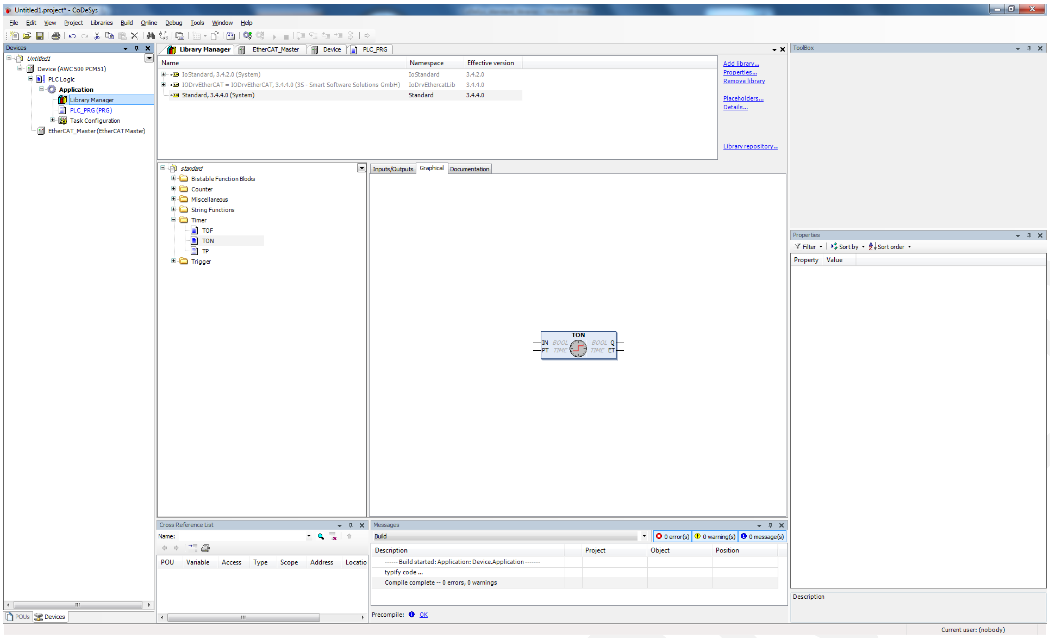
Encoder value 1: 684633
Encoder value offset 1: 0
Pos. Reference 1 [deg]: 92.000000
Blade Position 1: 41.967999
Pos. Reference 1: 92.000000
Vel. Reference 1 [deg/s]: 5.000000
Blade Velocity 1: 5.109368
Pitch Velocity Demand 1: 5.000000
Drive Demand 1 [V]: 5.000000

Encoder value 2: 684633
Encoder value offset 2: 0
Pos. Reference 2 [deg]: 92.000000
Blade Position 2: 41.967999
Pos. Reference 2: 92.000000
Vel. Reference 2 [deg/s]: 5.000000
Blade Velocity 2: 5.109368
Pitch Velocity Demand 2: 5.000000
Drive Demand 2 [V]: 5.000000

Encoder value 3: 684633
Encoder value offset 3: 0
Pos. Reference 3[deg] : 92.000000
Blade Position 3: 41.967999
Pos. Reference 3: 92.000000
Vel. Reference 3[deg/s]: 5.000000
Blade Velocity 3: 5.109368
Pitch Velocity Demand 3: 5.000000
Drive Demand 3 [V]: 5.000000

Figure 1.5: Example HMI for Pitch Motion Controller.

1.2 CoDeSys Standard libraries



Name	Namespace	Effective version
IoStandard, 3.4.2.0 (System)	IoStandard	3.4.2.0
IODrvEtherCAT = IODrvEtherCAT, 3.4.4.0 (3S - Smart Software Solutions GmbH)	IoDrvEthercatLib	3.4.4.0
Standard, 3.4.4.0 (System)	Standard	3.4.4.0

Description	Project	Object	Position
------ Build started: Application: Device.Application -------			
typify code ...			
Compile complete -- 0 errors, 0 warnings			

Figure 1.6: CoDeSys Library manager

The following libraries are provided with CoDeSys V3:

Library	Functionality
SysCom.library	Serial synchronous communication with a target device
SysComAsync.library	Serial asynchronous communication with a target device
SysCpuHandling.library	IEC function call, test and reset of bits
SysDir.library	Handling of a file system on the target device (synchron)
SysDirAsync.library	Handling of a file system on the target device (asynchronous)
SysEvent.library	Synchronisation and controlling the processing between two (IEC-) tasks
SysFile.library	Handling of a file system (synchronous accesses) on the target device
SysFileAsync.library	Handling of a file system (asynchronous accesses) on the target device
SysInt.library	Applying an interrupt handler on a function
SysMem.library	Memory management
SysPci.library	Access on PCI-cards connected to the system
SysPort.library	Communication with external hardware modules via their port addresses, e.g. realtime clock, graphic controller etc. (synchronous)
SysPortAsync.library	Communication with external hardware modules via their port addresses, e.g. realtime clock, graphic controller etc. (asynchronous)
SysProcess.library	Process handling on a single-processing target system
SysSem.library	Creating and using semaphores for task synchronization
SysSemProcess.library	Using semaphores for process synchronisation
SysShm.library	Creating and accessing a Shared Memory

Table 1.1: CoDeSys V3 libraries (1 of 2)

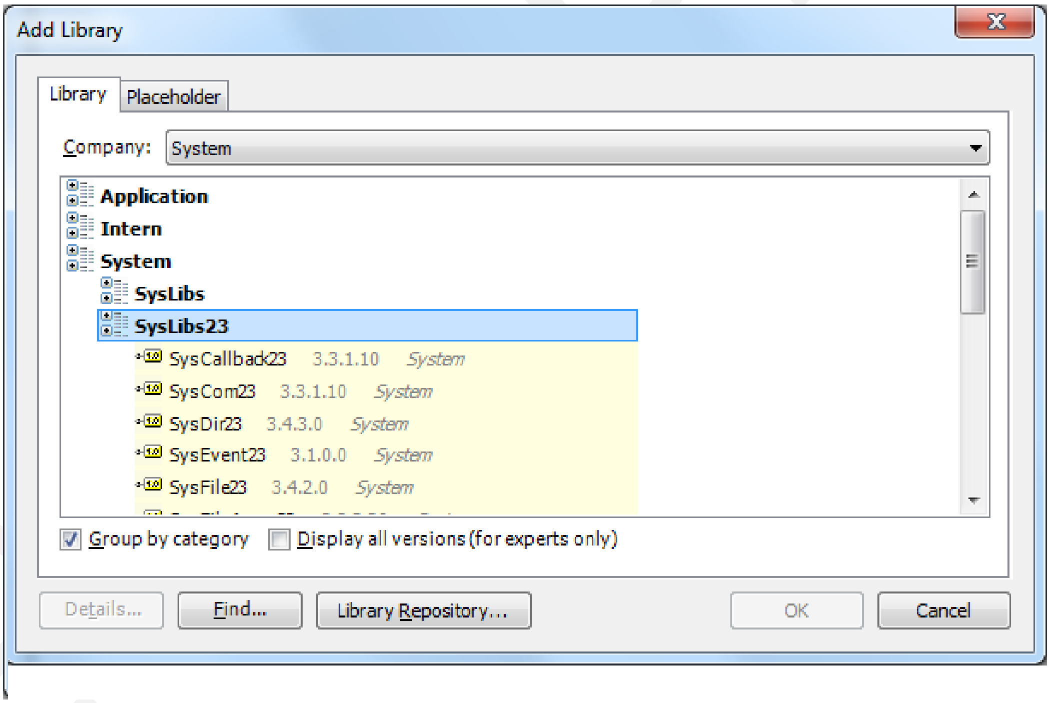
Library	Functionality
SysSocket.library	Access on sockets for the communication via TCP/IP and UDP (synchronous)
SysSocketAsync.library	Asynchronous access on sockets for the communication via TCP/IP and UDP (asynchronous)
SysTask.library	Task management (see also SysIECTasks.library)
SysTime.library	Additional functions for reading the realtime clock of the computer (see also SysRtc.library); is needed in addition to the SysTaskInfo.library for displaying the task time evaluation in the CoDeSys Task Configuration
SysTimer.library	Implementing a timer for triggering an event of calling a function
SysTypes.library	Platform comprehensive constants and file types for the runtime system

Table 1.2: CoDeSys V3 libraries (2 of 2)

Detailed documentation of each library can be found under
`C:\Program Files\3S CoDeSys\CoDeSys\Documentation\en`

1.2.1 CoDeSys V2 compliant libraries

Under Add Library →SysLibs23 a number of libraries exists that helps ensuring that existing PLC projects that have been running under CoDeSys version 2, executes under CoDeSys V3. Include these in the project if required or if you are familiar with the use:



Company: System

- Application
- Intern
- System
  - SysLibs
  - SysLibs23
    - SysCallback23 3.3.1.10 System
    - SysCom23 3.3.1.10 System
    - SysDir23 3.4.3.0 System
    - SysEvent23 3.1.0.0 System
    - SysFile23 3.4.2.0 System

☑ Group by category □ Display all versions (for experts only)

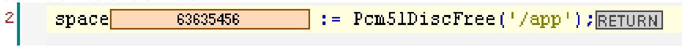
Figure 1.7: CoDeSys V2 compliant libraries

1.3 DEIF Libraries

1.3.1 CmpPcm51Clib.library

Disk space monitoring from CoDeSys.

Pcm51DiskFree: The function Pcm51DiskFree will return how many bytes are free on a mounted device.



```
space 63635456 := Pcm51DiscFree('/app');RETURN
```

`/app` : Root of the application file system
`/mmc` : Root of the MMC/SDCARD file system
`/tmp` : Root of the /tmp (RAM)

Pcm51Exit: Exits the CoDeSys runtime. It will be restarted by the Operating System

Pcm51Reboot: Reboot the PCM51 Operating System

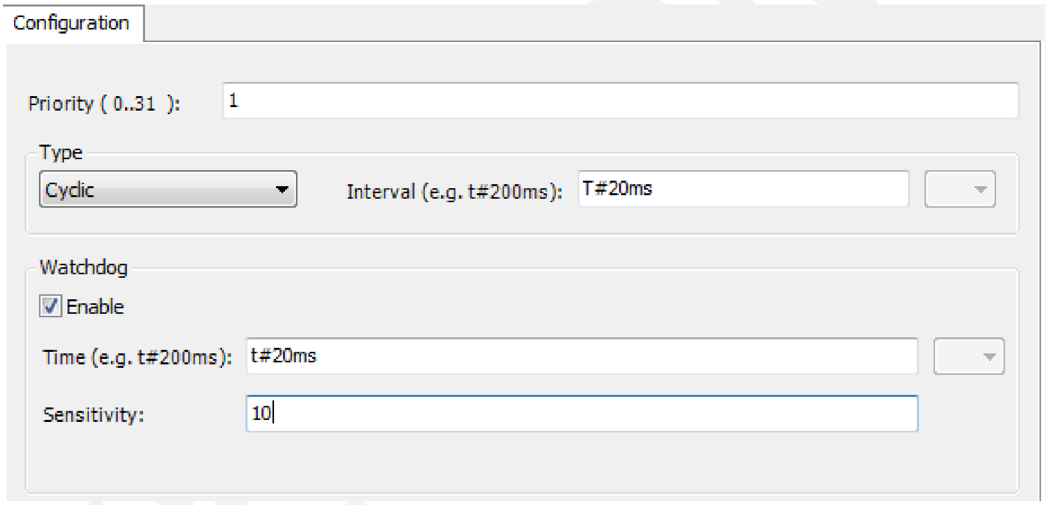
1.3.2 Writing CoDeSys log

CmpLog can be used to write to the CoDeSys log. Example:

```
result := CmpLog.LogAdd2( STD_LOGGER, 16#1000, LogClass.LOG_INFO, 0, 0 ,
                          'This␣is␣added␣to␣the␣CoDeSys␣log' );
```

1.4 Watchdog

To control handling on CPU overload (a task taking too long time execute in a certain program) the Watchdog can be enabled using the following steps:



Priority (0..31): 1

Type: Cyclic

Interval (e.g. t#200ms): T#20ms

Watchdog

☑ Enable

Time (e.g. t#200ms): t#20ms

Sensitivity: 10

Figure 1.8: Enabling watchdog

In Task Configuration, see Figure 1.8

1. Select Watchdog: Enable
2. Set Time: t#20ms
3. Sensitivity: 10

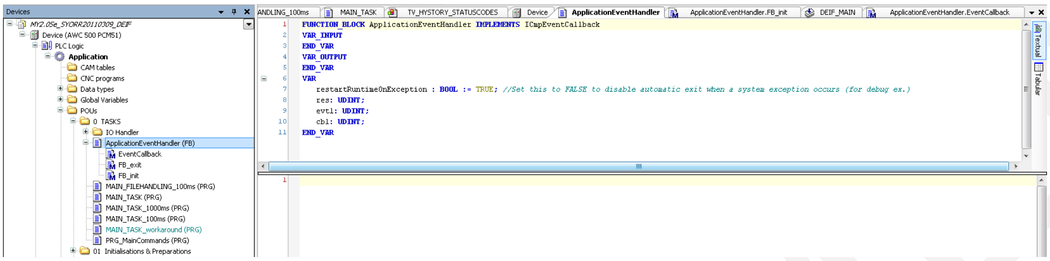
The CoDeSys will then cause a Watchdog exception, and the state of the Digital output will be set to default state. Initialization code can be protected against the watchdog with:
After declaring an appropriate variable for the handle of the task (of type RTS_IEC_HANDLE), `hIecTask : RTS_IEC_HANDLE;` the disabling (and succeeding reenabling) can be handled by employing the interface functions in the following manner:

```
hIecTask := IecTaskGetCurrent(0);
IecTaskDisableWatchdog(hIecTask);
... // Code that is protected against watchdog
IecTaskEnableWatchdog(hIecTask);
```

1.4.1 Support for runtime reboot upon exception error

This goes for both FPU, Watchdog, Segmentation fault and more. Once a fault occurs the runtime will reboot and be up and running within 5 seconds. All implementation is location in the "ApplicationEventHandler" function block in the "0 TASKS" folder. The functions block is instantiated in the MAIN_TASK(PRG) (WatchdogExceptions: ApplicationEventHandler;) and you may choose to disable the auto runtime reboot functionality by setting the restartRuntimeOnException to FALSE.

Add the FUNCTION BLOCK ApplicationEventHandler IMPLEMENTS IcmpEventCallback to the project.



```
FUNCTION_BLOCK ApplicationEventHandler IMPLEMENTS ICmpEventCallback
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
   restartRuntimeOnException : BOOL := TRUE; //Set this to FALSE to disable automatic exit when a system exception occurs (for debug ex.)
   res: UDINT;
   evtl: UDINT;
   cbl: UDINT;
END_VAR
```

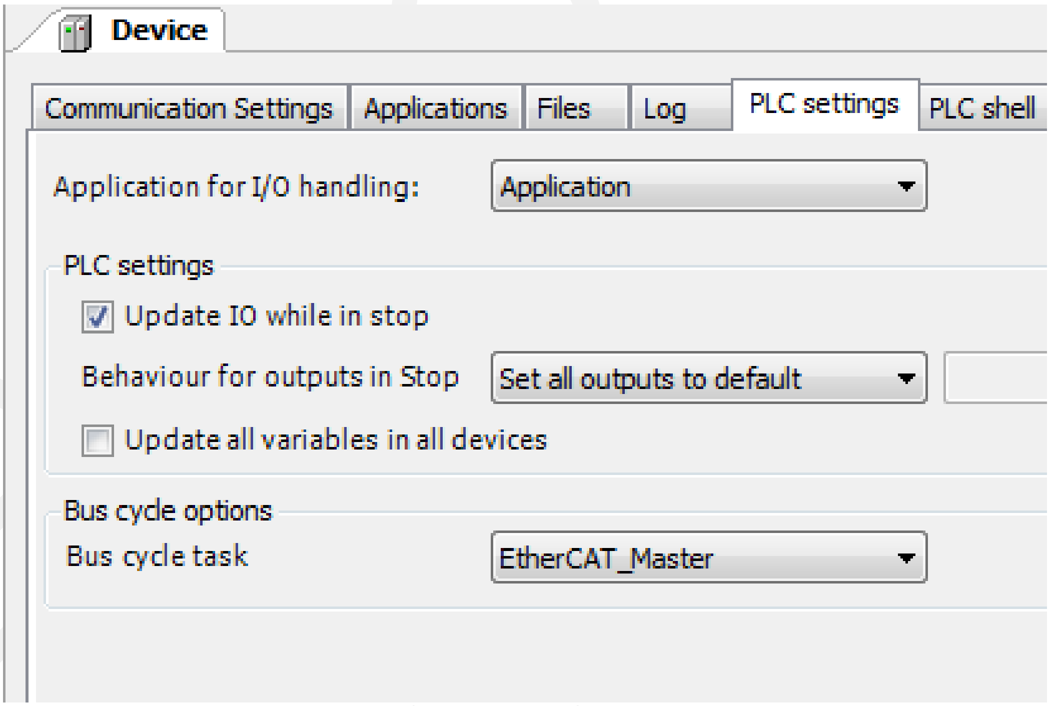
Figure 1.9: The ApplicationEventHandler function block

1.5 Default output state

Digital Output can be set to default state on exceptions:

In Device→PLC Settings, see Figure 1.8

1. Enable "Update IO while in stop"
2. Set "Behaviour for outputs in Stop" to "Set all outputs to default".



Application for I/O handling: Application

PLC settings

☑ Update IO while in stop

Behaviour for outputs in Stop: Set all outputs to default

☐ Update all variables in all devices

Bus cycle options

Bus cycle task: EtherCAT_Master

Figure 1.10: Setting Digital Output default state

1.6 Hardware monitoring

Functionblock	Parameter/Property	Results
IoDrvEtherCAT	NumberActiveSlaves	This property returns the number of the actually connected slaves. If StartConfigWithLessDevice is TRUE, then the number of real devices can be determined. Example: **VAR** startWithLessDevice: **BOOL**; **END_VAR** **Program**: startWithLessDevice := EtherCAT_Master.StartConfigWithLessDevice;

IoDrvEtherCAT	xConfigFinished	If this parameter is TRUE, the transfer of all configuration parameters has been finished successfully. The communication is running.
		Example:
		VAR
		    configFinished: **BOOL**;
		END_VAR
		Program:
		EtherCAT_Master();
		configFinished := EtherCAT_Master.xConfigFinished;

IoDrvEtherCAT	xError	This output will get TRUE if an error is detected during the start of the EtherCAT stack or if during operation the communication with the slaves gets interrupted because no more messages can be received (for example due to a cable break). The error cause can be made out with the help of the logging list resp. via the error string. Thereby "Ethercat_Master.LastMessage" is the instance of the master.
		Example:
		VAR
		    error: **BOOL**;
		END_VAR
		Program:
		EtherCAT_Master();
		EtherCAT_Master.xError;

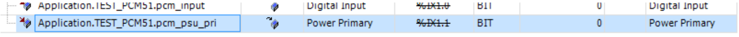
IoDrvEtherCAT	LastMessage	This property returns a string with the latest message of the EtherCAT Stacks. If the start has been done successfully, "All slaves done" should be returned. The string is the same as used for the diagnostic message shown in the EtherCAT Master device editor in online mode
		Example:
		VAR
		    lastMesage: STRING(255);
		END_VAR
		Program:
		lastMesage := EtherCAT_Master.LastMessage;

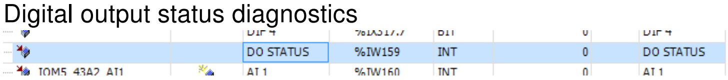
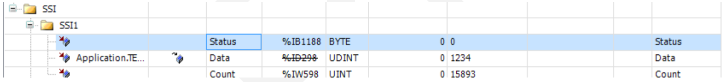
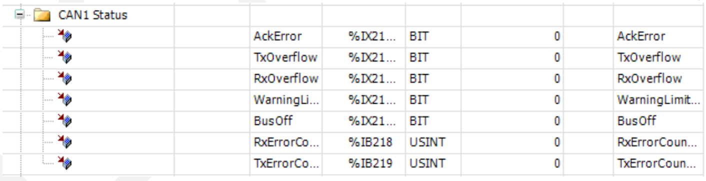
Functionblock	Parameter/Property	Results

ETCSlave	VendorID	After the start of the EtherCAT stack, this property returns the Vendor ID read from the device.
		Example:
		`VAR`
		`    productID: DWORD;`
		`END_VAR`
		`Program:`
		`vendorID := PCM51.VendorID;`

ETCSlave	ProductID	After the start of the EtherCAT stack this property returns the Product ID read from the device.
		Example:
		`VAR`
		`    serialID: DWORD;`
		`END_VAR`
		`Program:`
		`serialID:= PCM51. SerialID;`

ETCSlave	wState	The current state of the slave is returned. Possible values:
		`0: ETC_SLAVE_BOOT`
		`1: ETC_SLAVE_INIT`
		`2: ETC_SLAVE_PREOPERATIONAL`
		`4: ETC_SLAVE_SAVEOPERATIONAL`
		`8: ETC_SLAVE_OPERATIONAL`
		Example: Declaration:
		`pSlave: POINTER TO ETCSlave;`
		Program:
		`pSlave := Ethercat_Master.FirstSlave;`
		`WHILE pSlave <> 0 DO`
		`pSlave^();`
		`IF pSlave^.wState =`
		`    ETC_SLAVE_STATE.ETC_SLAVE_OPERATIONAL THEN`
		`;`
		`END_IF`
		`pSlave := pSlave^.NextInstance;`
		`END_WHILE`

Input / Output		
PCM51 PDM51 PDM52	Power Primary	This signal whether the Primary Power on the rack is OK.  Application.TEST_PCM51.pcm_input Digital Input ~~%IX1.0~~ BIT 0 Digital Input Application.TEST_PCM51.pcm_psu_pri Power Primary ~~%IX1.1~~ BIT 0 Power Primary

Functionblock	Parameter/Property	Results
IOM51	DO STATUS	Digital output status diagnostics  DIP 4 %IX317.7 BIT 0 DIP 4 DO STATUS %IW159 INT 0 DO STATUS IOM5_43A2_AI1 AI 1 %IW160 INT 0 AI 1 Bit 0 indicate the status of the digital outputs 0: An error is indicated if one of the following conditions is filled: • Supply voltage is below 8V • Supply voltage is above 37V • Error is reported on one of the digital output drivers. (Over-current or over-temperature) After power-up the digital output (EnableDO) has to be enabled before error-bit is reset. If an over-current or over-temperature status has occurred, the digital outputs (EnableDO) have to be disabled and then enabled again to clear the error state. 1 : Digital Outputs are OK Bit 1 indicates state of IOM calibration. 0 : NOT calibrated, 1 : Calibrated
IFM51	SSI Status	This signal should be 0 if things are OK.  SSI SSI1 Status %IB1188 BYTE 0 0 Status Application.TE... Data ~~%ID298~~ UDINT 0 1234 Data Count %IW598 UINT 0 15893 Count Bit0: high if SSI line error Bit1: high if Frame error
IFM51	CAN Status	Monitor the status of the CAN bus via the following inputs:  CAN1 Status AckError %IX21... BIT 0 AckError TxOverflow %IX21... BIT 0 TxOverflow RxOverflow %IX21... BIT 0 RxOverflow WarningLi... %IX21... BIT 0 WarningLimit... BusOff %IX21... BIT 0 BusOff RxErrorCo... %IB218 USINT 0 RxErrorCoun... TxErrorCo... %IB219 USINT 0 TxErrorCoun...

<table>
<tr><th colspan="3">CANopen</th></tr>
<tr><th>Functionblock</th><th>Parameter/Property</th><th>Results</th></tr>
<tr><td>CANopen Manager</td><td>stkGetInfo</td><td>
<b>ENUM STK_STATE</b>
<table>
<tr><th>Name</th><th>Type</th><th>Inherited from</th></tr>
<tr><td><b>NO_STATE</b></td><td>INT</td><td></td></tr>
<tr><td><b>INITALISATION</b></td><td>INT</td><td></td></tr>
<tr><td><b>RESET_APPLICATION</b></td><td>INT</td><td></td></tr>
<tr><td><b>RESET_COMMUNICATION</b></td><td>INT</td><td></td></tr>
<tr><td><b>PRE_OPERATIONAL</b></td><td>INT</td><td></td></tr>
<tr><td><b>OPERATIONAL</b></td><td>INT</td><td></td></tr>
<tr><td><b>STOPPED</b></td><td>INT</td><td></td></tr>
</table>
Declaration:
<pre>
    getstate   : CS.PROC_STATE;
    oError   : CS.ERROR;

    (* Information from GetInfo *)
    infoStk   : CS.STK_INFO;
    infoNet   : CS.NET_INFO;
    infoError : CS.ERROR;

    (* Work variables *)
    getdataexcecute : CS.PROC_CMD;
</pre>
Program:
<pre>
(* Function call for getting canopen stack info *)
CANopen_Manager.StkGetInfo(stkGetInfo => getstate,
    eCmd := getdataexcecute, pStkInfo := ADR(infoStk),
    pNetInfo := ADR(infoNet), eError => infoError);

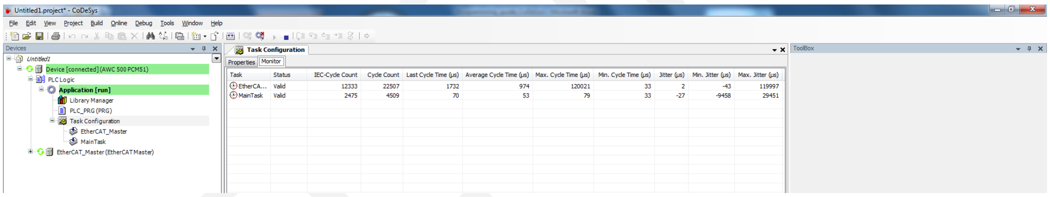
(* Get data for the current canopen stack on
'CANopen_Manager' *)
IF getdata = TRUE THEN
    getdataexcecute := CS.PROC_CMD.EXECUTE;
    getdata := FALSE;
END_IF

(* handles stopping the modules on error or OK *)
IF(getstate <> CS.PROC_STATE.BUSY) THEN
   getdataexcecute := CS.PROC_CMD.NONE;
   IF (infoStk.eState = CS.STK_STATE.OPERATIONAL)
      (* stack is operational *)
   END_IF
END_IF
</pre>
</td></tr>
</table>

<table>
<tr><th colspan="3">CANopen</th></tr>
<tr><th>Functionblock</th><th>Parameter/Property</th><th>Results</th></tr>
<tr><td>CANRemote-Device</td><td>nCanOpenState</td><td>CIA405.DEVICE_STATE the states:
<table>
<tr><th>Name</th><th>Type</th></tr>
<tr><td><b>INIT</b></td><td>INT</td></tr>
<tr><td><b>RESET_COMM</b></td><td>INT</td></tr>
<tr><td><b>RESET_APP</b></td><td>INT</td></tr>
<tr><td><b>PRE_OPERATIONAL</b></td><td>INT</td></tr>
<tr><td><b>STOPPED</b></td><td>INT</td></tr>
<tr><td><b>OPERATIONAL</b></td><td>INT</td></tr>
<tr><td><b>UNKNOWN</b></td><td>INT</td></tr>
<tr><td><b>NOT_AVAIL</b></td><td>INT</td></tr>
</table>
Example:
<pre>
PowerConverter ();
State := PowerConverter.CANOpenState
IF (state = CIA405.DEVICE_STATE.OPERATIONAL) THEN
  (* Power converter is Operational *)
  bValidData_CAN_Converter := TRUE;
END_IF
</pre></td></tr>
</table>

1.7 Monitoring programming load

In CoDeSys you can see the task cycle times and statistics under Task→Monitoring:



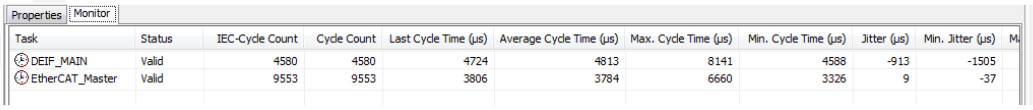
Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)	Min. Jitter (µs)	Max. Jitter (µs)
EtherCA...	Valid	12333	22507	1732	974	120021	33	2	-43	119997
MainTask	Valid	2475	4509	70	53	79	33	-27	-9458	29451

Figure 1.11: Monitoring tasks

Else you can measure execution time of each subsystem with time, E.g.:

```
start:=Time();
PRG_Subsystem1();
PRG_Subsystem2();
PRG_Subsystem3();
diff:=TIME()-start;
```

Important for ensuring Realtime operation is that none of the Cycle times exceed the Task interval e.g. that Last Cycle Time 4724 us is less than the Task Interval 20000 us, and does not exceed it.



Task	Status	IEC-Cycle Count	Cycle Count	Last Cycle Time (µs)	Average Cycle Time (µs)	Max. Cycle Time (µs)	Min. Cycle Time (µs)	Jitter (µs)	Min. Jitter (µs)	M
DEIF_MAIN	Valid	4580	4580	4724	4813	8141	4588	-913	-1505	
EtherCAT_Master	Valid	9553	9553	3806	3784	6660	3326	9	-37	

Figure 1.12: Monitoring tasks

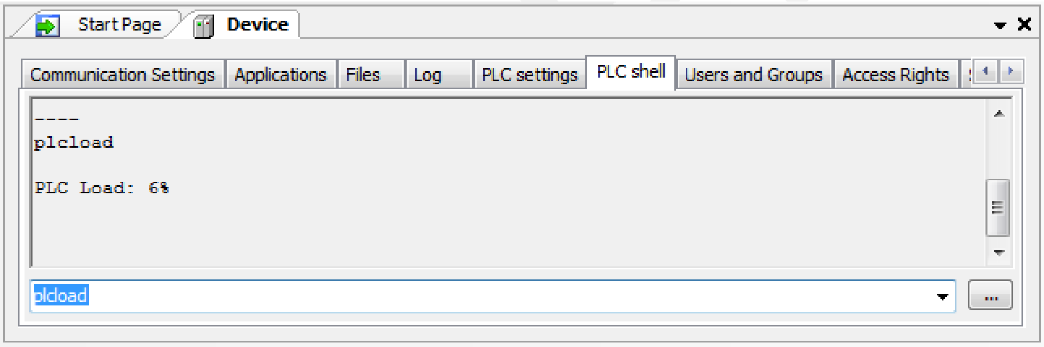
Two methods for measuring the cycle time via CoDeSys Task configuration or via Systemtime.

```
PROGRAM MAIN_20ms
VAR
(* CoDeSys Task cycle time configuration and measurement *)
    Result          : ISysTypes.RTS_IEC_RESULT;
    hTask          : ISysTypes.RTS_IEC_HANDLE;
    pTaskInfo2     : POINTER TO CmpIecTask.Task_Info2;
    dwInterval  : DWORD;
    dwCycleTime  : DWORD;
    dwMinCycleTime  : DWORD;
    dwMaxCycleTime  : DWORD;
    dwAvgCycleTime  : DWORD;

(* Execution time measurement *)
    start : SysTimeCore.SYSTIME;
    end : SysTimeCore.SYSTIME;
    cycletime : SysTimeCore.SYSTIME;
END_VAR
(* START OF FUNCTION_BLOCK / PROGRAM *)
SysTimeCore.SysTimeGetUs(pUsTime := start); (* Start time *)

(* CoDeSys Task cycle time configuration and measurement *)
hTask := CmpIecTask.IecTaskGetCurrent(pResult:=ADR(Result));
IF hTask <> ISysTypes.RTS_INVALID_HANDLE THEN
    pTaskInfo2 := CmpIecTask.IecTaskGetInfo3(hIecTask:=hTask, pResult:=ADR(Result));
    dwInterval:= pTaskInfo2^.dwInterval;
    gdwTaskIntervalDEIF_MAIN_20ms_usec:= dwInterval;
    dwCycleTime := pTaskInfo2^.dwCycleTime;
    gdwCycleTime_DEIF_MAIN := dwCycleTime;
    dwMinCycleTime := pTaskInfo2^.dwMinCycleTime;
    dwMaxCycleTime := pTaskInfo2^.dwMaxCycleTime;
    dwAvgCycleTime := pTaskInfo2^.dwAverageCycleTime;
END_IF

(lineN)
(lineN+1)
...
(lineN+?)


(* Task measurement *)
SysTimeCore.SysTimeGetUs(pUsTime := end); (* Measure end time *)
cycletime := end - start; (* Calculate Cycle time *)

(* END OF FUNCTION_BLOCK / PROGRAM *)


Task load is defined as CycleTime/TaskInterval. This can be cal:
PROGRAM MonitorTasks
VAR
    bReset : BOOL;
END_VAR

(* START OF FUNCTION_BLOCK / PROGRAM *)
(* Cycle-Times *)
gdwCycleTime_DEIF_MAIN_min := MIN(gdwCycleTime_DEIF_MAIN_min, gdwCycleTime_DEIF_MAIN);
gdwCycleTime_DEIF_MAIN_max := MAX(gdwCycleTime_DEIF_MAIN_max, gdwCycleTime_DEIF_MAIN);

(* Cycle-Exceed *)
IF gdwTaskIntervalDEIF_MAIN_20ms_usec <> 0 THEN
    IF gdwCycleTime_DEIF_MAIN > gdwTaskIntervalDEIF_MAIN_20ms_usec THEN
        gbDEIF_MAIN_CycleTimeExceeded := TRUE;
        guiDEIF_MAIN_CycleTimeExceedCounter := guiDEIF_MAIN_CycleTimeExceedCounter + 1;
    END_IF
```

```
END_IF

(* Reset *)
IF bReset THEN
    guiDEIF_MAIN_CycleTimeExceedCounter := 0;
    gdwCycleTime_DEIF_MAIN_min := 1000000;
    gdwCycleTime_DEIF_MAIN_max := 0;
    gbDEIF_MAIN_CycleTimeExceeded := FALSE;
    bReset := FALSE;
END_IF

(* Task Load *)
IF gdwTaskIntervalDEIF_MAIN_20ms_usec <> 0 THEN
    (*Calculate task load in [%] *)
    grTaskLoad_20ms_Percent := LIMIT(0, DWORD_TO_REAL(gdwCycleTime_DEIF_MAIN) /
                                  DWORD_TO_REAL(gdwTaskIntervalDEIF_MAIN_20ms_usec) *
                                  100.0, 100);
END_IF
(* END OF FUNCTION_BLOCK / PROGRAM *)
```

1.7.1 Monitoring load via CoDeSys PLC shell

Another indication about the PLC load can be found via the CoDeSys PLC shell (Device->PLC shell).



```
----
plcload

PLC Load: 6%
```

plcload

Figure 1.13: Example of PLC load via CoDeSys PLC shell

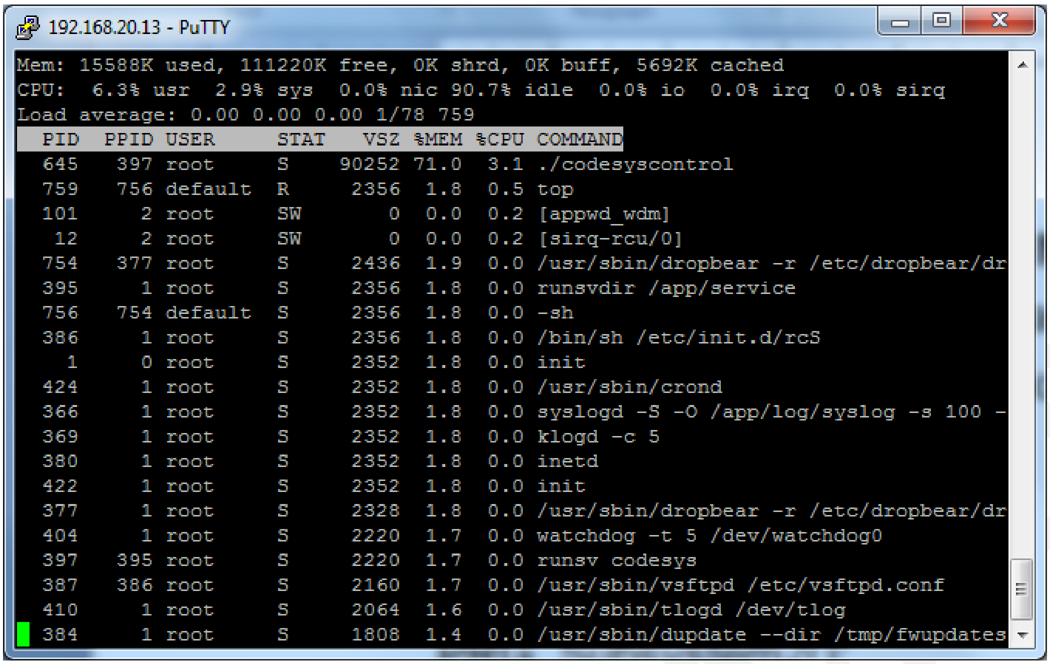
1.7.2 Monitoring Processor load via variable

The above PLC load measurement can be monitored with

```
VAR
            Result          : ISysTypes.RTS_IEC_RESULT;
                        ProcessorLoad_Percent : REAL;
END_VAR

(* Processor Load *)
ProcessorLoad_Percent := UDINT_TO_REAL( SchedGetProcessorLoad(pResult:=ADR(Result)) );
```

1.7.3 Monitoring load via Linux

You can log in to the unit and via the SSH and run the command `top` see the CPU load.



```
Mem: 15588K used, 111220K free, 0K shrd, 0K buff, 5692K cached
CPU:  6.3% usr  2.9% sys  0.0% nic 90.7% idle  0.0% io  0.0% irq  0.0% sirq
Load average: 0.00 0.00 0.00 1/78 759
  PID  PPID USER     STAT   VSZ %MEM %CPU COMMAND
  645   397 root     S     90252 71.0  3.1 ./codesyscontrol
  759   756 default  R      2356  1.8  0.5 top
  101     2 root     SW        0  0.0  0.2 [appwd_wdm]
   12     2 root     SW        0  0.0  0.2 [sirq-rcu/0]
  754   377 root     S      2436  1.9  0.0 /usr/sbin/dropbear -r /etc/dropbear/dr
  395     1 root     S      2356  1.8  0.0 runsvdir /app/service
  756   754 default  S      2356  1.8  0.0 -sh
  386     1 root     S      2356  1.8  0.0 /bin/sh /etc/init.d/rcS
    1     0 root     S      2352  1.8  0.0 init
  424     1 root     S      2352  1.8  0.0 /usr/sbin/crond
  366     1 root     S      2352  1.8  0.0 syslogd -S -O /app/log/syslog -s 100 -
  369     1 root     S      2352  1.8  0.0 klogd -c 5
  380     1 root     S      2352  1.8  0.0 inetd
  422     1 root     S      2352  1.8  0.0 init
  377     1 root     S      2328  1.8  0.0 /usr/sbin/dropbear -r /etc/dropbear/dr
  404     1 root     S      2220  1.7  0.0 watchdog -t 5 /dev/watchdog0
  397   395 root     S      2220  1.7  0.0 runsv codesys
  387   386 root     S      2160  1.7  0.0 /usr/sbin/vsftpd /etc/vsftpd.conf
  410     1 root     S      2064  1.6  0.0 /usr/sbin/tlogd /dev/tlog
  384     1 root     S      1808  1.4  0.0 /usr/sbin/dupdate --dir /tmp/fwupdates
```

Figure 1.14: Example of top output

1.7.4 Description of the "top" command

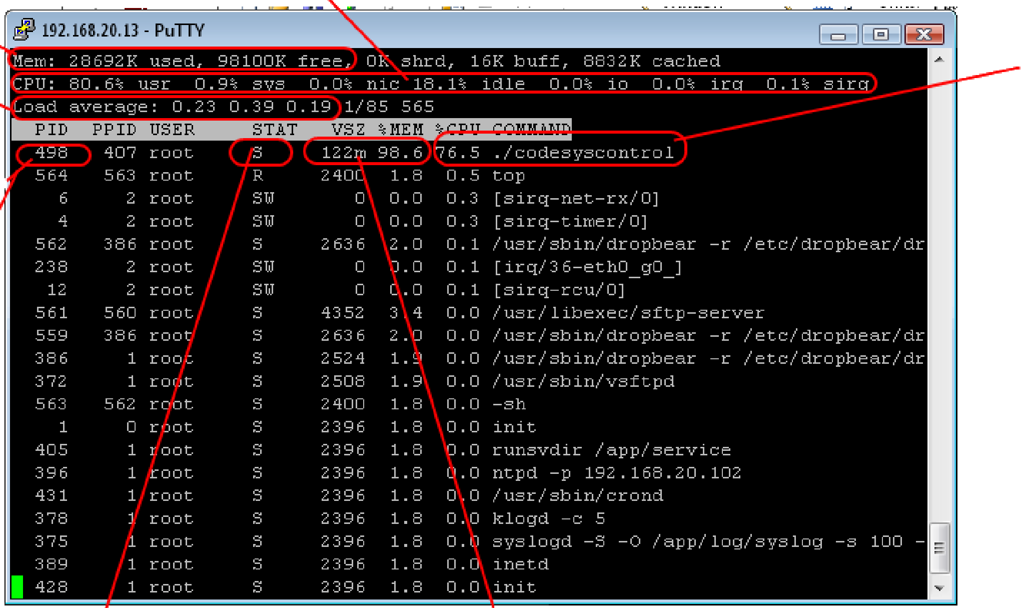
Description of the linux 'top' command

This CPU line are describes were the CPU is used not how much.
usr = user space (ex. codesys), sys, nic, io, irq = linux/system/network, idle = idle (not doing anything).
This line will/should always be 98-100% even if the CPU load is only 20%

Memory Usage
Total memory usage in AWC500
Here 28MB are used and 98MB are free

Load Average (CPU)
This is the linux way of measuring CPU load if load average is below 1.00 it means that the CPU can keep up.
The numbers displayed are:
<5min> <10min> <15min>
measured load average

Normally a load average below 0.7 is considered acceptable load. Load average may spike during startup of linux but settle within 5 minuttes.



```
Mem: 28692K used, 98100K free, 0K shrd, 16K buff, 8832K cached
CPU: 80.6% usr  0.9% sys  0.0% nic 18.1% idle  0.0% io  0.0% irq  0.1% sirq
Load average: 0.23 0.39 0.19 1/85 565
  PID  PPID USER     STAT   VSZ %MEM %CPU COMMAND
  498   407 root     S     122m 98.6 76.5 ./codesyscontrol
  564   563 root     R     2400  1.8  0.5 top
    6     2 root     SW       0  0.0  0.3 [sirq-net-rx/0]
    4     2 root     SW       0  0.0  0.3 [sirq-timer/0]
  562   386 root     S     2636  2.0  0.1 /usr/sbin/dropbear -r /etc/dropbear/dr
  238     2 root     SW       0  0.0  0.1 [irq/36-eth0_g0_]
   12     2 root     SW       0  0.0  0.1 [sirq-rcu/0]
  561   560 root     S     4352  3.4  0.0 /usr/libexec/sftp-server
  559   386 root     S     2636  2.0  0.0 /usr/sbin/dropbear -r /etc/dropbear/dr
  386     1 root     S     2524  1.9  0.0 /usr/sbin/dropbear -r /etc/dropbear/dr
  372     1 root     S     2508  1.9  0.0 /usr/sbin/vsftpd
  563   562 root     S     2400  1.8  0.0 -sh
    1     0 root     S     2396  1.8  0.0 init
  405     1 root     S     2396  1.8  0.0 runsvdir /app/service
  396     1 root     S     2396  1.8  0.0 ntpd -p 192.168.20.102
  431     1 root     S     2396  1.8  0.0 /usr/sbin/crond
  378     1 root     S     2396  1.8  0.0 klogd -c 5
  375     1 root     S     2396  1.8  0.0 syslogd -S -O /app/log/syslog -s 100 -
  389     1 root     S     2396  1.8  0.0 inetd
  428     1 root     S     2396  1.8  0.0 init
```

The total CPU load of each thread. The total load is all the CPU % added together
A better way to measure CPU load in linux is the load average.

Process ID this can be used to kill a process.
ex: 'kill -9 498' for restarting codesys in this example.
The process ID will change on each startup.

The status of the task.
R = Running
S = Sleeping (normal mode, waiting for interrupt or cyclic task tick)
W = Waiting (normal mode, waiting for event)
Z = Zombi mode, task did not exit correctly

How much of the Total memory is used by each task.
Ex, codesys is uring 98.6% of the 28692K used in this example
The VSZ value is not important since it deals in virual memory usage.

1.8 EtherCAT

Please note the CoDeSys Runtime must be stopped before EtherCAT Scan is possible.

1.9 Persistent variables

The AWC 500 has 128kB non-volatile memory, the parameters are saved instantly on assignment, so on the event of a power down no further actions need to be taken for storing the variables.

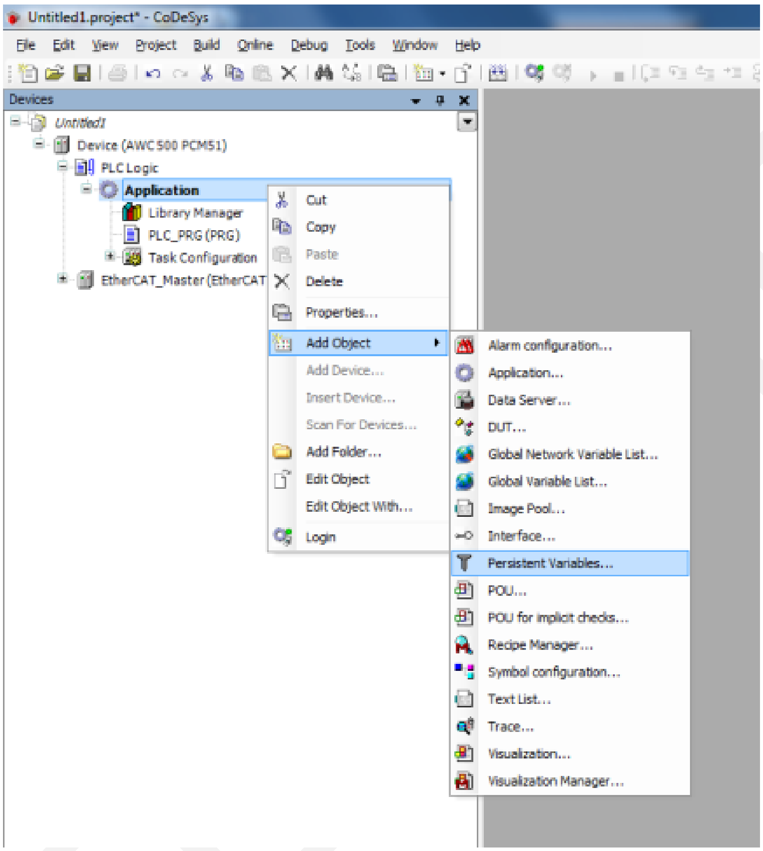
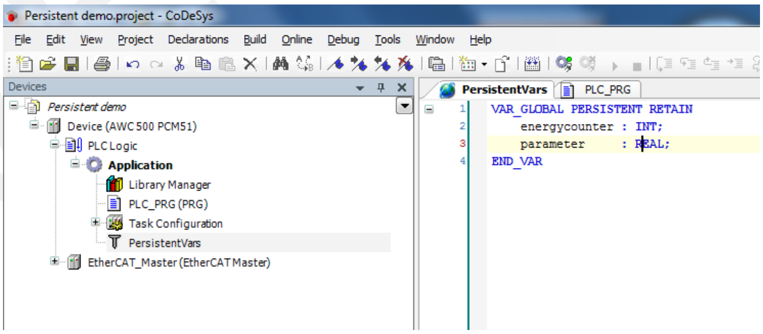


Figure 1.15: Adding persistent variables to the project

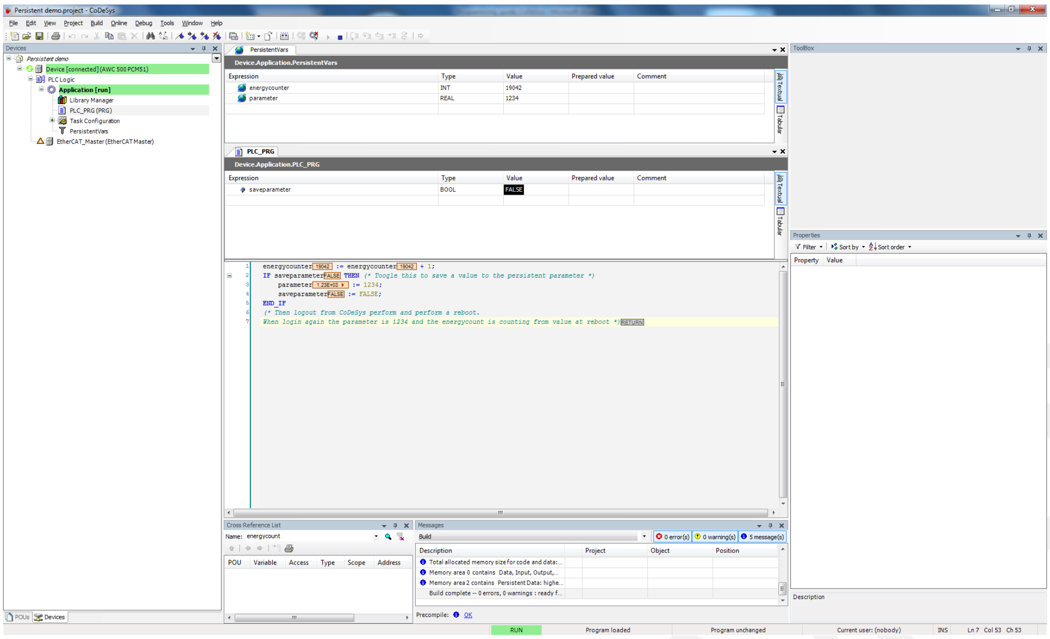


```
VAR_GLOBAL PERSISTENT RETAIN
    energycounter : INT;
    parameter     : REAL;
END_VAR
```

Figure 1.16: Modifying global variables to become persistant variables

Or simply add "PERSISTENT RETAIN" to the global variables VAR declaration.

This simple test shows the persistent variables:



Device.Application.PersistentVars

Expression	Type	Value	Prepared value	Comment
energycounter	INT	19042		
parameter	REAL	1234		

Device.Application.PLC_PRG

Expression	Type	Value	Prepared value	Comment
saveparameter	BOOL	FALSE		

```
energycounter 19042 := energycounter 19042 + 1;
IF saveparameter FALSE THEN (* Toogle this to save a value to the persistent parameter *)
    parameter 1.23E+03 := 1234;
    saveparameter FALSE := FALSE;
END_IF
(* Then logout from CoDeSys perform and perform a reboot.
When login again the parameter is 1234 and the energycount is counting from value at reboot *)RETURN
```

Cross Reference List — Name: energycount

POU	Variable	Access	Type	Scope	Address

Messages — Build — 0 error(s) 0 warning(s) 5 message(s)

Description	Project	Object	Position
Total allocated memory size for code and data:...			
Memory area 0 contains Data, Input, Output,...			
Memory area 2 contains Persistent Data: highe...			
Build complete -- 0 errors, 0 warnings : ready f...			

Precompile: OK

RUN Program loaded Program unchanged

Figure 1.17: Testing persistant variables

It is recommended to read default parameters from a text file on the flash instead e.g.:

```
rMyVar:= DEIFReadLREALParameter('Parameter.txt', 'MyVariable');
DEIFWriteLREALParameters('Parameter.txt', rMyVar);
```

The following presents the format of the Parameter.txt file:

```
[MyVariable = 1.232]
```

1.10 Handling Division by zero

You should always check if the divisor is zero before making a division, and else signal a flag and skip the division.

However some build in features are available on AWC 400 and AWC 500 :

If you define functions in your project with the names CheckDivByte, CheckDivWord, CheckDivDWord and CheckDivReal, CheckDivLReal, you can use them to check the value of the divisor if you use the operator DIV, for example to avoid a division by 0. The functions must have the above listed names.
AWC 500 : Use the Add Object → POUs for implicit checks

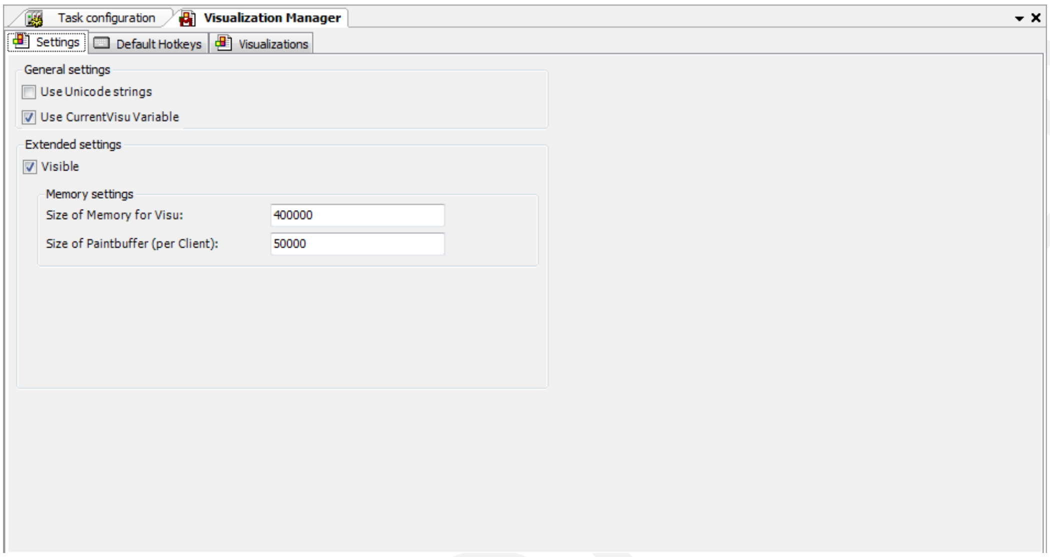
```
FUNCTION CheckDivLReal : REAL
VAR_INPUT
divisor:REAL;
END_VAR

IF divisor = 0 THEN
CheckDivLReal:=1;
(* Adding the following line allows to idenfity in which POU division by zero occurs.*)
    (* Set gCurrentPOU in beginning of each POU. *)
(* gDivisionByZeroOccuredAt:=gCurrentPOU; *)

ELSE
CheckDivLReal:=divisor;
END_IF;
```

1.11 Creating dynamic language switching in HMI

All static texts added to visualization during development of a project are automatically added to a Global Text list with an incremented ID, under a "default" language. These can only be edited in the visualization.
Additional languages can be added to this list as new columns.

1.12 Extending visualisation memory

When using much visualization on one page, the memory can be extended.



General settings

□ Use Unicode strings

☑ Use CurrentVisu Variable

Extended settings

☑ Visible

Memory settings

Size of Memory for Visu: 400000

Size of Paintbuffer (per Client): 50000

Figure 1.18: Extending visualisation memory

1.13 Distributing applications

Binary boot project package of files can be created to distribute CoDeSys application projects. The files can then be uploaded via FTP/SFTP remotely. The advantage is that binary files and not the source PLC project is distribued.

1.13.1 Creating a boot project (for remote upload)

Start with the working PLC project in CoDeSys.
Go to Online→Create Boot Application

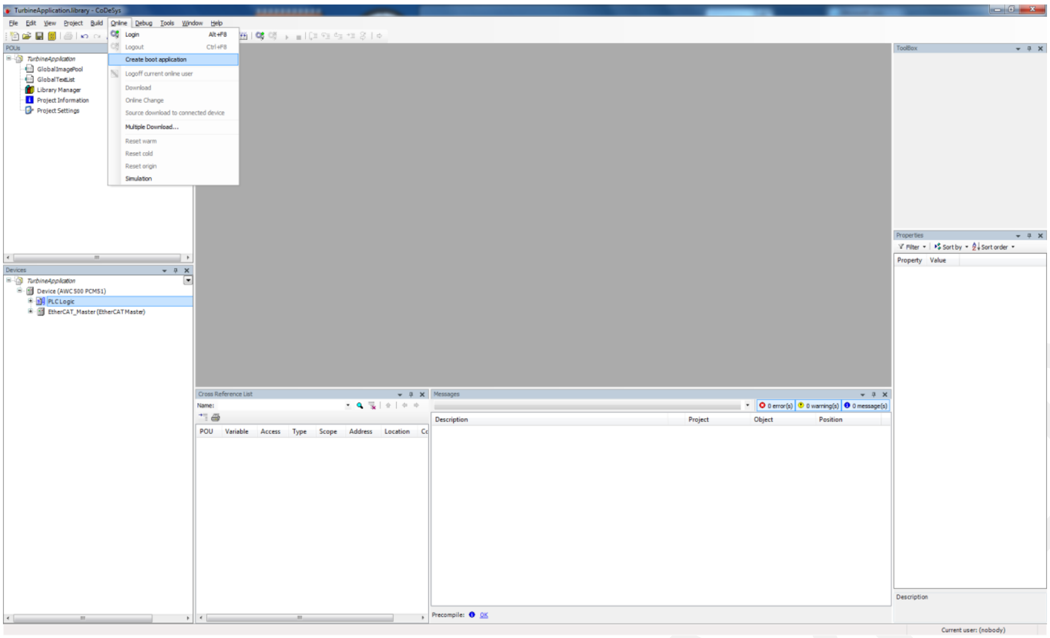
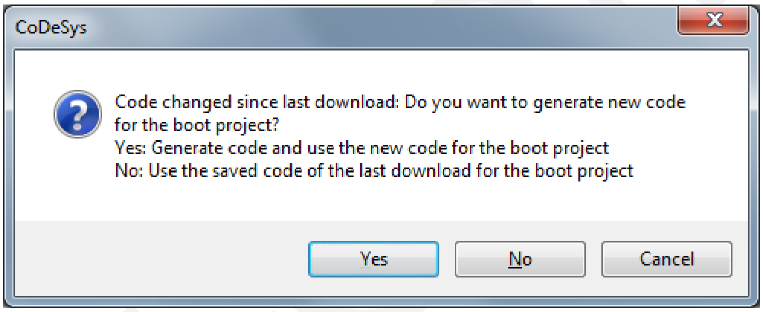
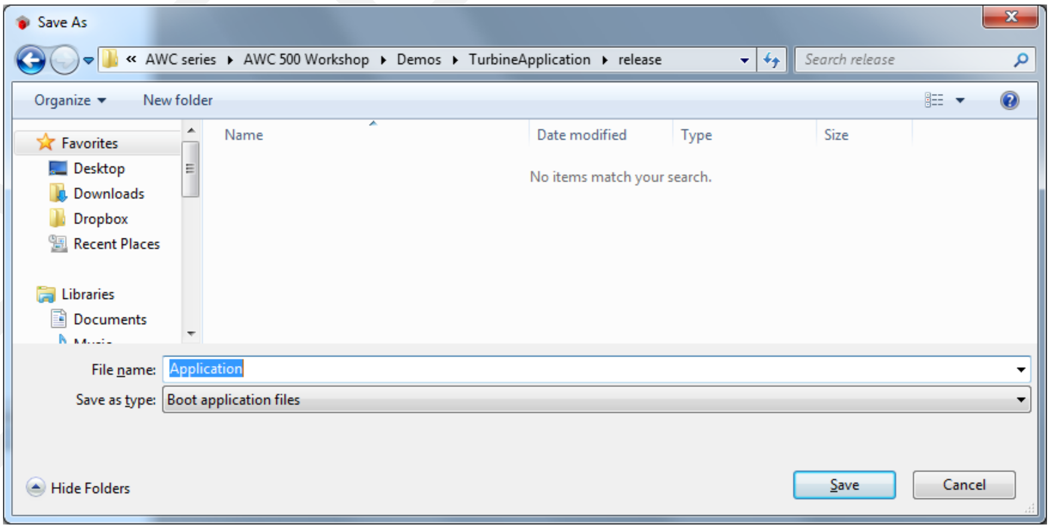


Figure 1.19: Create Boot Application



Code changed since last download: Do you want to generate new code for the boot project?
Yes: Generate code and use the new code for the boot project
No: Use the saved code of the last download for the boot project

Figure 1.20: Select Yes to generate new code.

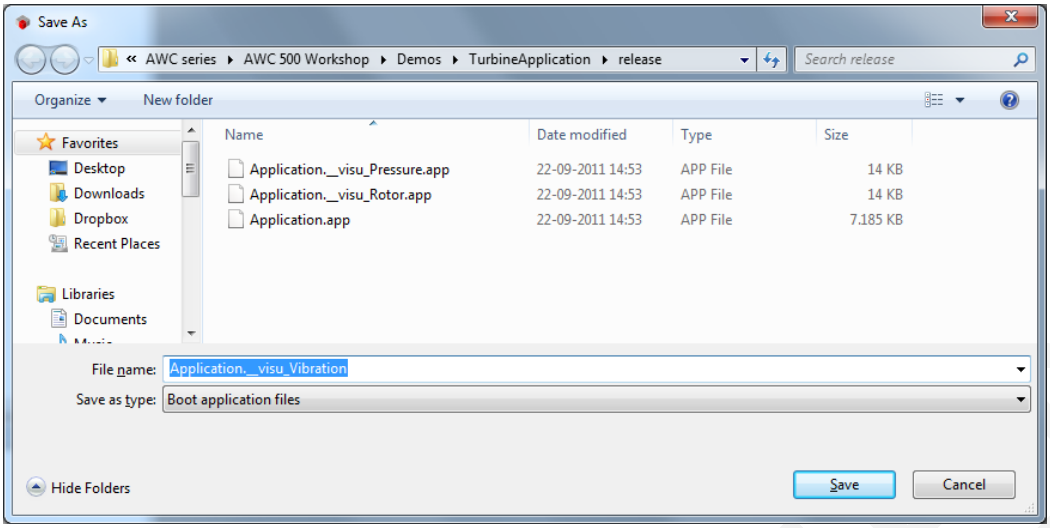


File name: Application

Save as type: Boot application files

Figure 1.21: Here the project is named "Application"

Additional files are also saved for visualization.



Name	Date modified	Type	Size
Application._visu_Pressure.app	22-09-2011 14:53	APP File	14 KB
Application._visu_Rotor.app	22-09-2011 14:53	APP File	14 KB
Application.app	22-09-2011 14:53	APP File	7.185 KB

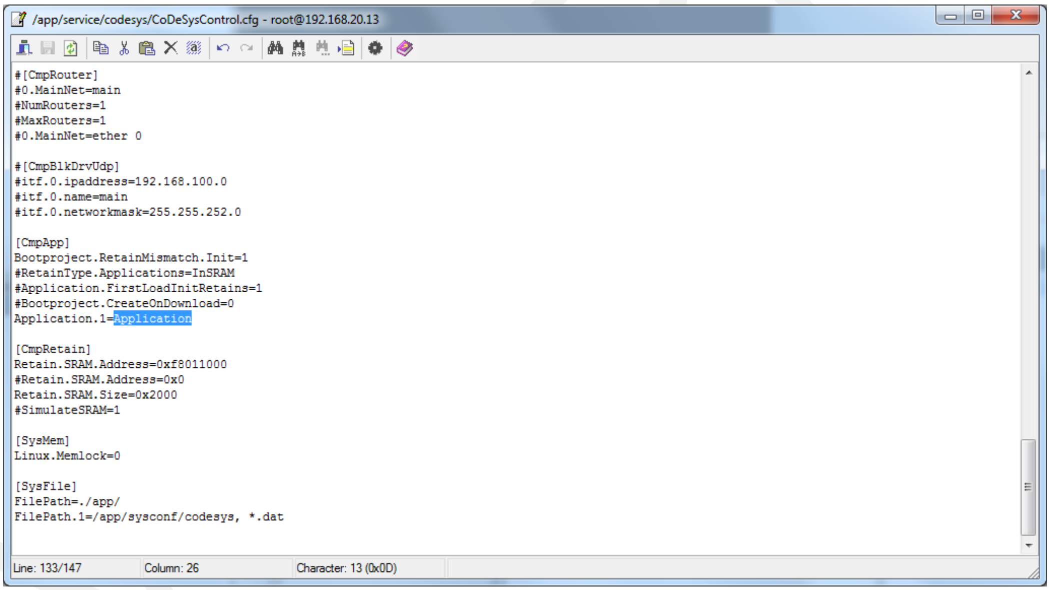
File name: Application._visu_Vibration

Save as type: Boot application files

Figure 1.22: Save all the proposed files

Changing boot application name

If the generated application is named something else than Application, it must be changed in CoDeSysControl.cfg located in /app/service/codesys/ and rename "Application" under PLC Logic in CoDeSys.



/app/service/codesys/CoDeSysControl.cfg - root@192.168.20.13

```
#[CmpRouter]
#0.MainNet=main
#NumRouters=1
#MaxRouters=1
#0.MainNet=ether 0

#[CmpBlkDrvUdp]
#itf.0.ipaddress=192.168.100.0
#itf.0.name=main
#itf.0.networkmask=255.255.252.0

[CmpApp]
Bootproject.RetainMismatch.Init=1
#RetainType.Applications=InSRAM
#Application.FirstLoadInitRetains=1
#Bootproject.CreateOnDownload=0
Application.1=Application

[CmpRetain]
Retain.SRAM.Address=0xf8011000
#Retain.SRAM.Address=0x0
Retain.SRAM.Size=0x2000
#SimulateSRAM=1

[SysMem]
Linux.Memlock=0

[SysFile]
FilePath=./app/
FilePath.1=/app/sysconf/codesys, *.dat
```

Line: 133/147 Column: 26 Character: 13 (0x0D)

Figure 1.23: Edit application name in CoDeSysControl.cfg

If the PLC application name is changed and used as a Boot application, then the application name must be changed in the /app/service/codesys/CoDeSys.cfg file also,E.g.:

```
[CmpApp]
Bootproject.RetainMismatch.Init=1
RetainType.Applications=InSRAM
#RetainType.Applications=OnPowerfail
#Application.FirstLoadInitRetains=1
#Bootproject.CreateOnDownload=0
Application.1= MY_APPLICATION_NAME
```

Change `MY_APPLICATION_NAME` to the specific name used in CoDeSys.

```
Application.1= Application
```

Please note by default "Application" is used as name for the application.

Instead of using the node id or hostname to identify the AWC 500 , then an additional Nodename can be set.

Add the lines:

```
[SysTarget]
NodeName=MY_NODE_NAME
```

Eg:

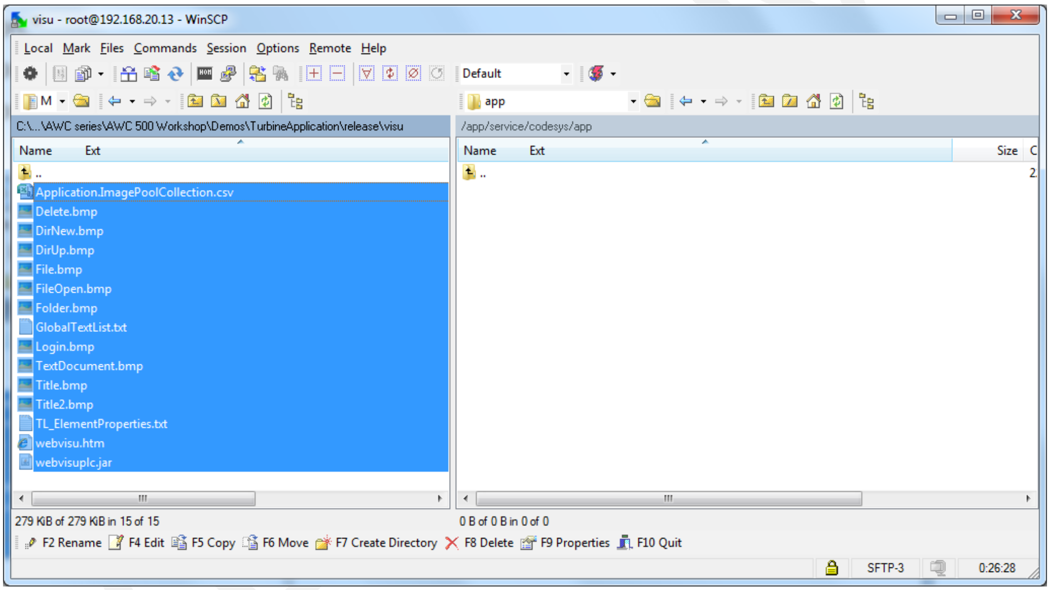
```
[SysTarget]
NodeName=awc500pcm-test-rack-c
```

Note: Do no change other settings in the codesys.cfg file, as it may harm the AWC 500 unit.

1.13.2 Generated boot project package

The generated boot project package now looks like:

`\release\visu`	Visualization files.
`\release\Application.*`	Visualization files
`\release\Application.app`	Boot project application
`\release\Application.crc`	Check sum file


visu - root@192.168.20.13 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
M
app
Name Ext
Name Ext Size
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:26:28

C:\...\AWC series\AWC 500 Workshop\Demos\TurbineApplication\release\visu

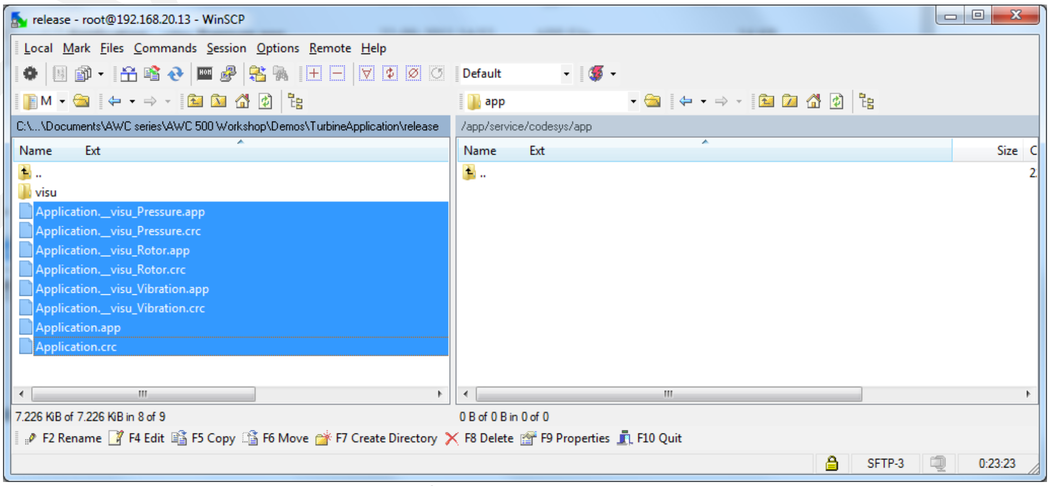
/app/service/codesys/app

..
Application.ImagePoolCollection.csv
Delete.bmp
DirNew.bmp
DirUp.bmp
File.bmp
FileOpen.bmp
Folder.bmp
GlobalTextList.txt
Login.bmp
TextDocument.bmp
Title.bmp
Title2.bmp
TL_ElementProperties.txt
webvisu.htm
webvisuplc.jar

279 KiB of 279 KiB in 15 of 15

0 B of 0 B in 0 of 0

Figure 1.24: Contents of the "visu" folder


release - root@192.168.20.13 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
M
app
Name Ext
Name Ext Size
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:23:23

C:\...\Documents\AWC series\AWC 500 Workshop\Demos\TurbineApplication\release

/app/service/codesys/app

..
visu
Application._visu_Pressure.app
Application._visu_Pressure.crc
Application._visu_Rotor.app
Application._visu_Rotor.crc
Application._visu_Vibration.app
Application._visu_Vibration.crc
Application.app
Application.crc

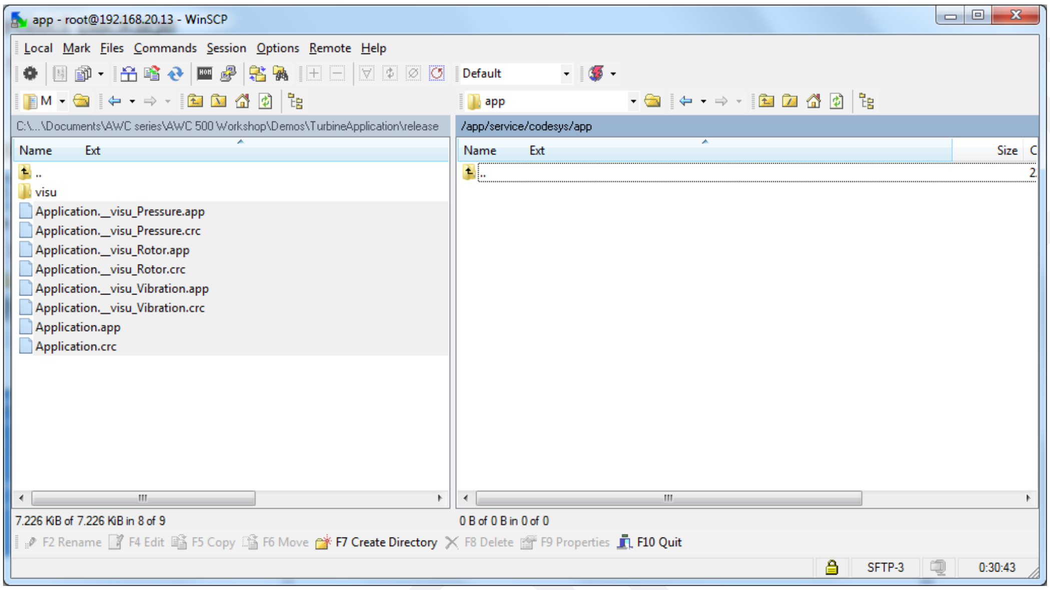
7.226 KiB of 7.226 KiB in 8 of 9

0 B of 0 B in 0 of 0

Figure 1.25: Contents of the "release" folder

1.13.3 Remote upload procedure

1. Stop existing turbine application to pause/idling state.
2. Login via SFTP to the AWC 500 .
3. Upload the boot project package
4. Delete all files under `/app/service/codesys/app` (right pane Figure 1.26)
5. Upload all files (right pane Figure 1.26) under `\release` to `/app/service/codesys/app`


app - root@192.168.20.13 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
M
app
Name Ext
Name Ext Size
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:30:43

C:\...\Documents\AWC series\AWC 500 Workshop\Demos\TurbineApplication\release

```
..
visu
Application._visu_Pressure.app
Application._visu_Pressure.crc
Application._visu_Rotor.app
Application._visu_Rotor.crc
Application._visu_Vibration.app
Application._visu_Vibration.crc
Application.app
Application.crc
```

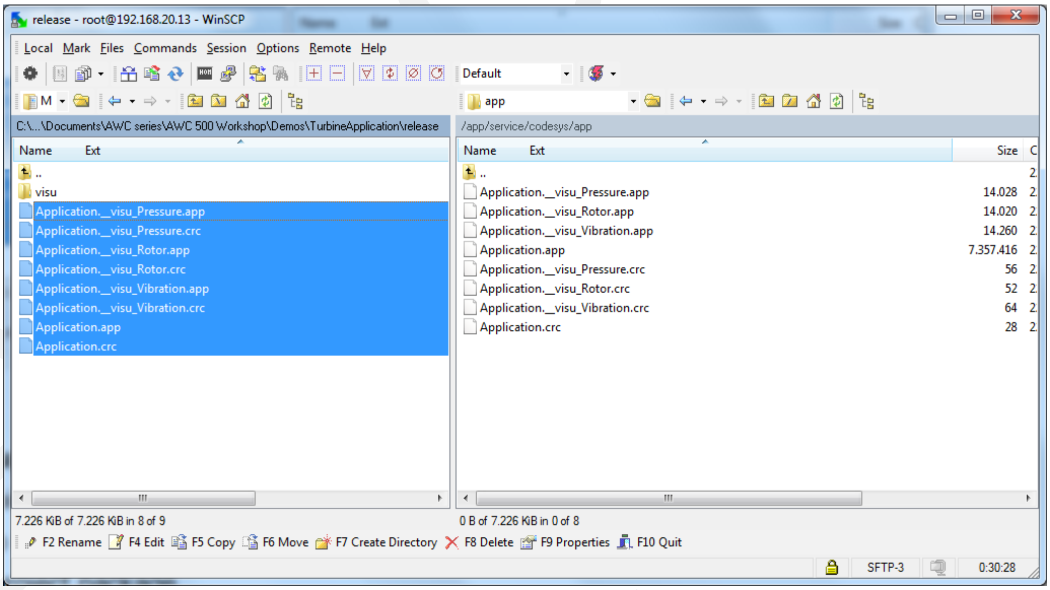
7.226 KiB of 7.226 KiB in 8 of 9

/app/service/codesys/app

```
..
```

0 B of 0 B in 0 of 0

Figure 1.26: Remote upload procedure


release - root@192.168.20.13 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
M
app
Name Ext
Name Ext Size
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3
0:30:28

C:\...\Documents\AWC series\AWC 500 Workshop\Demos\TurbineApplication\release

```
..
visu
Application._visu_Pressure.app
Application._visu_Pressure.crc
Application._visu_Rotor.app
Application._visu_Rotor.crc
Application._visu_Vibration.app
Application._visu_Vibration.crc
Application.app
Application.crc
```

7.226 KiB of 7.226 KiB in 8 of 9

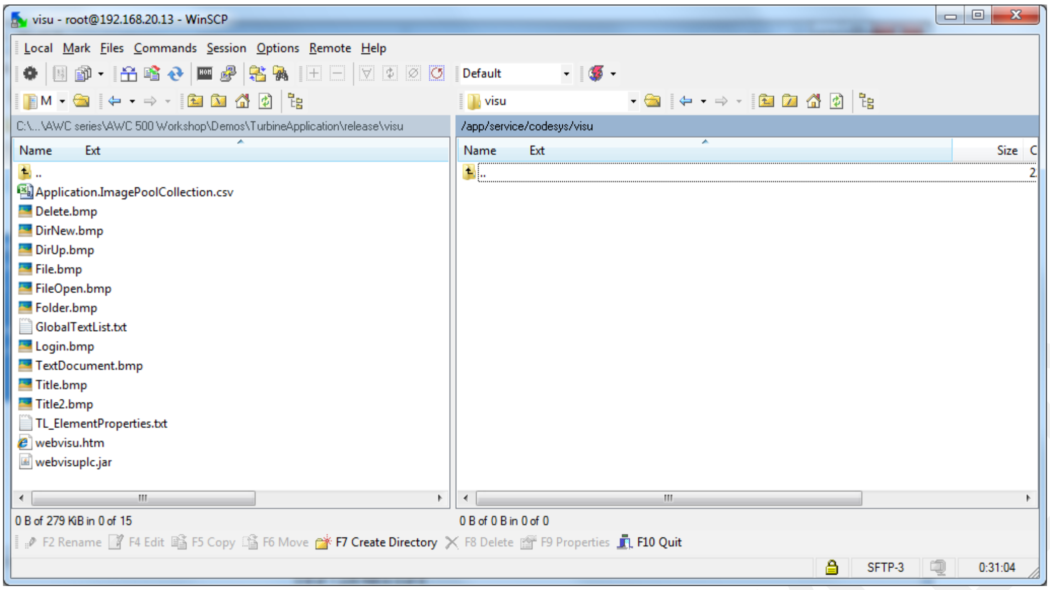
/app/service/codesys/app

```
..
Application._visu_Pressure.app       14.028
Application._visu_Rotor.app          14.020
Application._visu_Vibration.app      14.260
Application.app                   7.357.416
Application._visu_Pressure.crc           56
Application._visu_Rotor.crc              52
Application._visu_Vibration.crc          64
Application.crc                          28
```

0 B of 7.226 KiB in 0 of 8

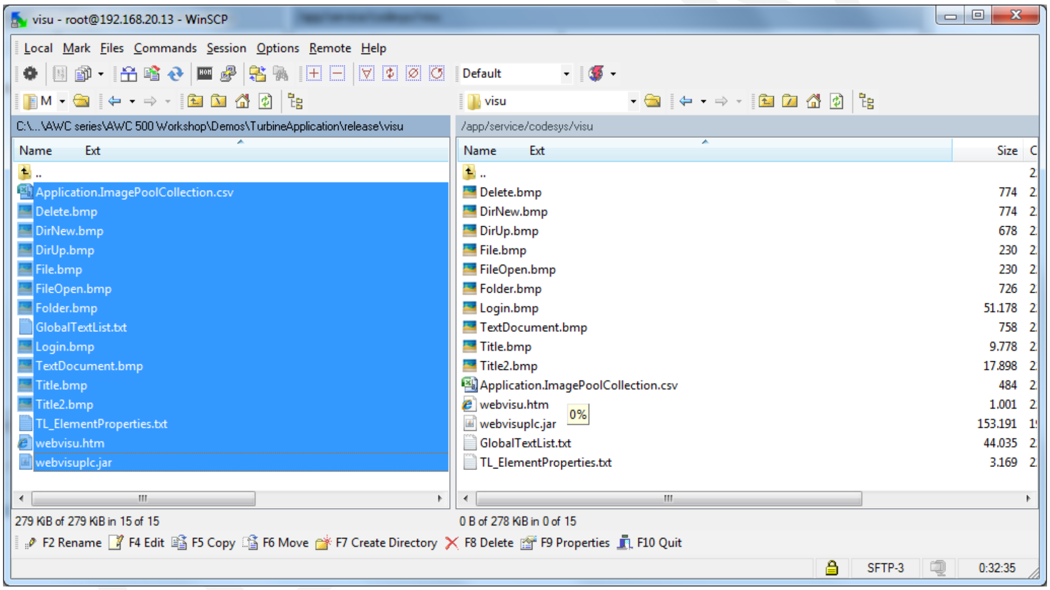
Figure 1.27: Uploaded files

6. On the AWC 500 change folder to `/app/service/codesys/visu`
7. Delete all existing files
8. Upload the new visualization files from `\release\visu` to `/app/service/codesys/visu`, see Figure 1.28


visu - root@192.168.20.13 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
M
visu
Name Ext
Name Ext Size
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3

```
C:\...\AWC series\AWC 500 Workshop\Demos\TurbineApplication\release\visu
..
Application.ImagePoolCollection.csv
Delete.bmp
DirNew.bmp
DirUp.bmp
File.bmp
FileOpen.bmp
Folder.bmp
GlobalTextList.txt
Login.bmp
TextDocument.bmp
Title.bmp
Title2.bmp
TL_ElementProperties.txt
webvisu.htm
webvisuplc.jar
0 B of 279 KiB in 0 of 15

/app/service/codesys/visu
..
0 B of 0 B in 0 of 0

0:31:04
```

Figure 1.28: Remote upload procedure, "visu" folder


visu - root@192.168.20.13 - WinSCP
Local Mark Files Commands Session Options Remote Help
Default
M
visu
Name Ext
Name Ext Size
F2 Rename F4 Edit F5 Copy F6 Move F7 Create Directory F8 Delete F9 Properties F10 Quit
SFTP-3

```
C:\...\AWC series\AWC 500 Workshop\Demos\TurbineApplication\release\visu
..
Application.ImagePoolCollection.csv
Delete.bmp
DirNew.bmp
DirUp.bmp
File.bmp
FileOpen.bmp
Folder.bmp
GlobalTextList.txt
Login.bmp
TextDocument.bmp
Title.bmp
Title2.bmp
TL_ElementProperties.txt
webvisu.htm
webvisuplc.jar
279 KiB of 279 KiB in 15 of 15

/app/service/codesys/visu
..
Delete.bmp                              774
DirNew.bmp                              774
DirUp.bmp                               678
File.bmp                                230
FileOpen.bmp                            230
Folder.bmp                              726
Login.bmp                            51.178
TextDocument.bmp                        758
Title.bmp                             9.778
Title2.bmp                           17.898
Application.ImagePoolCollection.csv     484
webvisu.htm                           1.001
webvisuplc.jar      0%              153.191
GlobalTextList.txt                   44.035
TL_ElementProperties.txt              3.169
0 B of 278 KiB in 0 of 15

0:32:35
```

Figure 1.29: Uploaded files

9. Finally do a remote reboot via SSH

2 AWC 500 module configurations

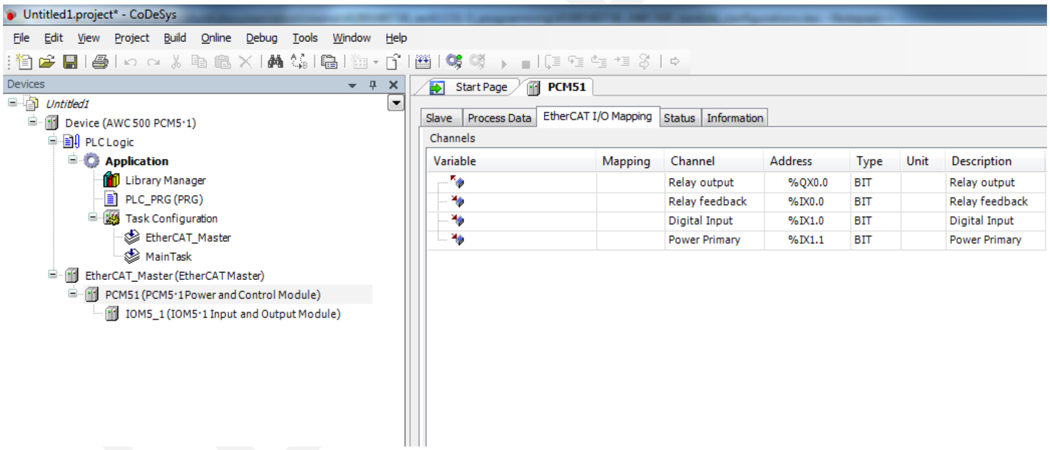
For each AWC 500 module physical inputs and outputs are linked under the "EtherCAT I/O Mapping" pane. Additional for each module start-up parameters can be set, to configure the module.

2.1 PCM5·1 – Power and Control Module

2.1.1 PCM5·1 – Terminal Description

Terminal		CoDeSys Process Variable	Terminal		CoDeSys Process Variable
6	Power supply Primary		10	Power supply Secondary	
7	Power supply Primary		11	Power supply Secondary	
8	In	Digital Input	12	Out	Relay output
9	In		13	Out	

2.1.2 PCM5·1 – EtherCAT I/O Mapping



Untitled1
- Device (AWC 500 PCM5·1)
  - PLC Logic
    - Application
      - Library Manager
      - PLC_PRG (PRG)
      - Task Configuration
        - EtherCAT_Master
        - MainTask
  - EtherCAT_Master (EtherCAT Master)
    - PCM51 (PCM5·1 Power and Control Module)
      - IOM5_1 (IOM5·1 Input and Output Module)

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		Relay output	%QX0.0	BIT		Relay output
		Relay feedback	%IX0.0	BIT		Relay feedback
		Digital Input	%IX1.0	BIT		Digital Input
		Power Primary	%IX1.1	BIT		Power Primary

Figure 2.1: PCM5·1 – EtherCAT I/O Mapping

Channel	IEC Address	Type	Description
Relay out	`%QX*`	BIT	Relay out
Relay feedback	`%IX*`	BIT	Relay feedback
Digital Input	`%IX*`	BIT	Digital Input
Power Primary	`%IX*`	BIT	Power Primary

2.1.3 PCM5·1 – Startup parameters

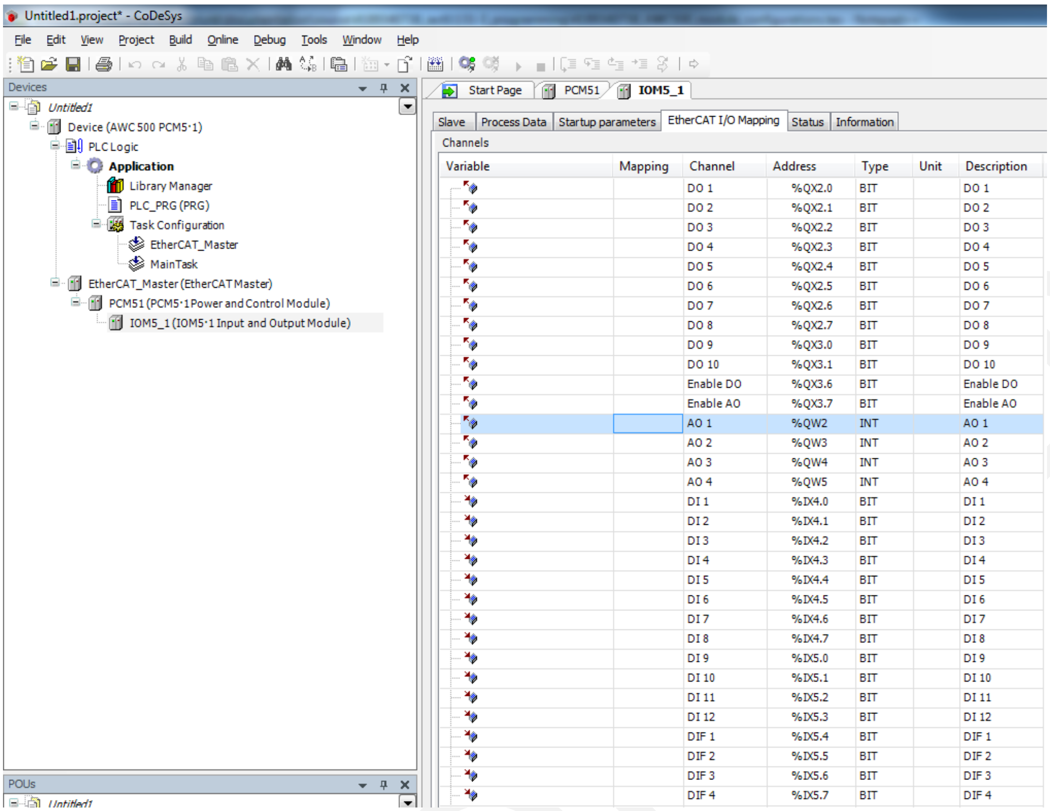
Currently none.

2.2 IOM5.1 – Input and Output Module

2.2.1 IOM5.1 – Terminal Description

Terminals		CODESYS Process Variable	Desciption
1–4	T1	Temp1	Temperature 1
5–8	T1	Temp2	Temperature 2
9–12	T1	Temp3	Temperature 3
41–44	T1	Temp4	Temperature 4
45–48	T1	Temp5	Temperature 5
49–42	T1	Temp6	Temperature 6
13–15	AI1	AI 1	Analogue Input 1. 13–14 using -20. . . 20 mA input, 14–15 using -10. . . 10 V input
16–18	AI2	AI 2	Analogue Input 2. 16–17 using -20. . . 20 mA input, 17–18 using -10. . . 10 V input
53–55	AI3	AI 3	Analogue Input 3. 53–54 using -20. . . 20 mA input, 54–55 using -10. . . 10 V input
56–58	AI4	AI 4	Analogue Input 4. 56–57 using -20. . . 20 mA input, 57–58 using -10. . . 10 V input
19–20	AO1	AO 1	Analogue Output -20. . . 20 mA
21–22	AO2	AO 2	Analogue Output -20. . . 20 mA
59–60	AO3	AO 3	Analogue Output -20. . . 20 mA
61–62	AO4	AO 4	Analogue Output -20. . . 20 mA
23	DI1	DI 1	Digital input 1
24	DI2	DI 2	Digital input 2
25	DI3	DI 3	Digital input 3
26	DI4	DI 4	Digital input 4
27	DI5	DI 5	Digital input 5
28	DI6	DI 6	Digital input 6
29	DI COM		Digital common input reference supply (DI1–DI12). 24 V for NPN input signal, GND for PNP input signal. Note: Terminal 29 and 69 are internally connected
63	DI7	DI 7	Digital input 7
64	DI8	DI 8	Digital input 8
65	DI9	DI 9	Digital input 9
66	DI10	DI 10	Digital input 10
67	DI11	DI 11	Digital input 11
68	DI12	DI 12	Digital input 12
69	DI COM		Digital common input reference supply (DI1–DI12). 24 V for NPN input signal, GND for PNP input signal. Note: Terminal 29 and 69 are internally connected
30–31	FI1	DIF 1 Tper1	Frequency input 1, NPN or PNP coupling Digital input 13, 30+, 31-
32–33	FI2	DIF 2 Tper2	Frequency input 2, NPN or PNP coupling Digital input 14, 32+, 33-
70–71	FI3	DIF 3 Tper3	Frequency input 3, NPN or PNP coupling Digital input 15, 70+, 71-
72–73	FI4	DIF 4 Tper4	Frequency input 4, NPN or PNP coupling Digital input 16, 72+, 73-
34	DO SUP+		24 V digital output supply. Note: Terminals 34 and 74 are internally connected
35	DO1	DO1	Digital Output 1
36	DO2	DO2	Digital Output 2
37	DO3	DO3	Digital Output 3
38	DO4	DO4	Digital Output 4
39	DO5	DO5	Digital Output 5
40	DO SUP-		GND digital output supply. Note: Terminals 40 and 80 are internally connected
74	DO SUP+		24 V digital output supply. Note: Terminals 34 and 74 are internally connected
75	DO6	DO6	Digital Output 6
76	DO7	DO7	Digital Output 7
77	DO8	DO8	Digital Output 8
78	DO9	DO9	Digital Output 9
79	DO10	DO10	Digital Output 10
80	DO SUP-		GND digital output supply. Note: Terminals 40 and 80 are internally connected

2.2.2 IOM5·1 –EtherCAT I/O Mapping



Variable	Mapping	Channel	Address	Type	Unit	Description
		DO 1	%QX2.0	BIT		DO 1
		DO 2	%QX2.1	BIT		DO 2
		DO 3	%QX2.2	BIT		DO 3
		DO 4	%QX2.3	BIT		DO 4
		DO 5	%QX2.4	BIT		DO 5
		DO 6	%QX2.5	BIT		DO 6
		DO 7	%QX2.6	BIT		DO 7
		DO 8	%QX2.7	BIT		DO 8
		DO 9	%QX3.0	BIT		DO 9
		DO 10	%QX3.1	BIT		DO 10
		Enable DO	%QX3.6	BIT		Enable DO
		Enable AO	%QX3.7	BIT		Enable AO
		AO 1	%QW2	INT		AO 1
		AO 2	%QW3	INT		AO 2
		AO 3	%QW4	INT		AO 3
		AO 4	%QW5	INT		AO 4
		DI 1	%IX4.0	BIT		DI 1
		DI 2	%IX4.1	BIT		DI 2
		DI 3	%IX4.2	BIT		DI 3
		DI 4	%IX4.3	BIT		DI 4
		DI 5	%IX4.4	BIT		DI 5
		DI 6	%IX4.5	BIT		DI 6
		DI 7	%IX4.6	BIT		DI 7
		DI 8	%IX4.7	BIT		DI 8
		DI 9	%IX5.0	BIT		DI 9
		DI 10	%IX5.1	BIT		DI 10
		DI 11	%IX5.2	BIT		DI 11
		DI 12	%IX5.3	BIT		DI 12
		DIF 1	%IX5.4	BIT		DIF 1
		DIF 2	%IX5.5	BIT		DIF 2
		DIF 3	%IX5.6	BIT		DIF 3
		DIF 4	%IX5.7	BIT		DIF 4

Figure 2.2: IOM5.1 – EtherCAT I/O Mapping

Channel	IEC Address	Type	Value range	Description
Digital Inputs				
DI 1. . . 12	`%IX*`	BIT	0/1	Filtered state of digital inputs 1. . . 12
DIF 1. . . 4	`%IX*`	BIT	0/1	Filtered state of digital inputs 13. . . 16
DO Status	`%IB*`	BYTE	0. . . 3	**Digital output status diagnostics** **Bit 0** indicate the status of the digital outputs. 0: An error is indicated if one of the following conditions is filled: • Supply voltage is below 8 V • Supply voltage is above 37 V • Error is reported on one of the digital output drivers. (Over-current or over-temperature) After power-up the digital output (**EnableDO**) has to be enabled before error-bit is reset. If an over-current or over-temperature status has occurred, the digital outputs (**EnableDO**) have to be disabled and then enabled again to clear the error state. 1 : Digital Outputs are OK **Bit 1** indicates state of IOM calibration. 0 : NOT calibrated, 1 : Calibrated

The digital inputs do not require any further configuration.

Channel	IEC Address	Type	Value range	Description
Digital Frequency Inputs				
Tper1...4	`%ID*`	UDINT	0...4294967295	Number of 800ns counts for one set of pulses (see: number of pulses in Digital Frequency Input configuration in Module Parameters)
Freqcount1...4	`%ID*`	UDINT	0...4294967295	Value in Freqcount is incremented each time a new pulse is detected.
Omega1	`%ID*`	DINT	±2147483647	Actual rotational speed normalized to $\pm 2^{31}$ corresponds to +/-1047.197551 rad/s
Alpha1	`%ID*`	DINT	±2147483647	Actual rotational acceleration normalized to $\pm 2^{31}$ = +/-1047.197551 rad/s^2

The Freqcount channel should be used to verify if the eg. the generator shaft is moving or not. If Freqcount is not increased for a time corresponding to the minimum speed of the shaft then the Tper channel should be dismissed and the RPM set to 0. For frequency input 1 on each IMO5.1 two additional values are offered directly: Omega1 offers the actual rotational speed and Alpha1 offers the acceleration. They can for example be used to detect or filter any physically impossible/unplanned speeds, accelerations or de-accelerations.

Channel	IEC Address	Type	Value range	Description
Analog Inputs				
AI 1...4	`%IW*`	INT	-25000...+25000	Actual reading (3-samples average) of analog input channel (±10 V or ±20 mA)

The Analog inputs type is configured with startup parameters Analog input 1...4 config.

Channel	IEC Address	Type	Value range	Description
Temperature Inputs				
Temp1...6	`%IW*`	INT	-25000...+25000	Actual temperature channel 1...6 average of last second's samples **Temperature diagnostics:** -32768 : Indicates a shortcircuit sensor 32767 : indicates an open sensor-input 0: indicates 0.0 °C -500...10000 (PT100) : indicates the actual temperature in 0.1 °C units

The Temperature input type is configured with startup parameters Temperature 1...6 config.

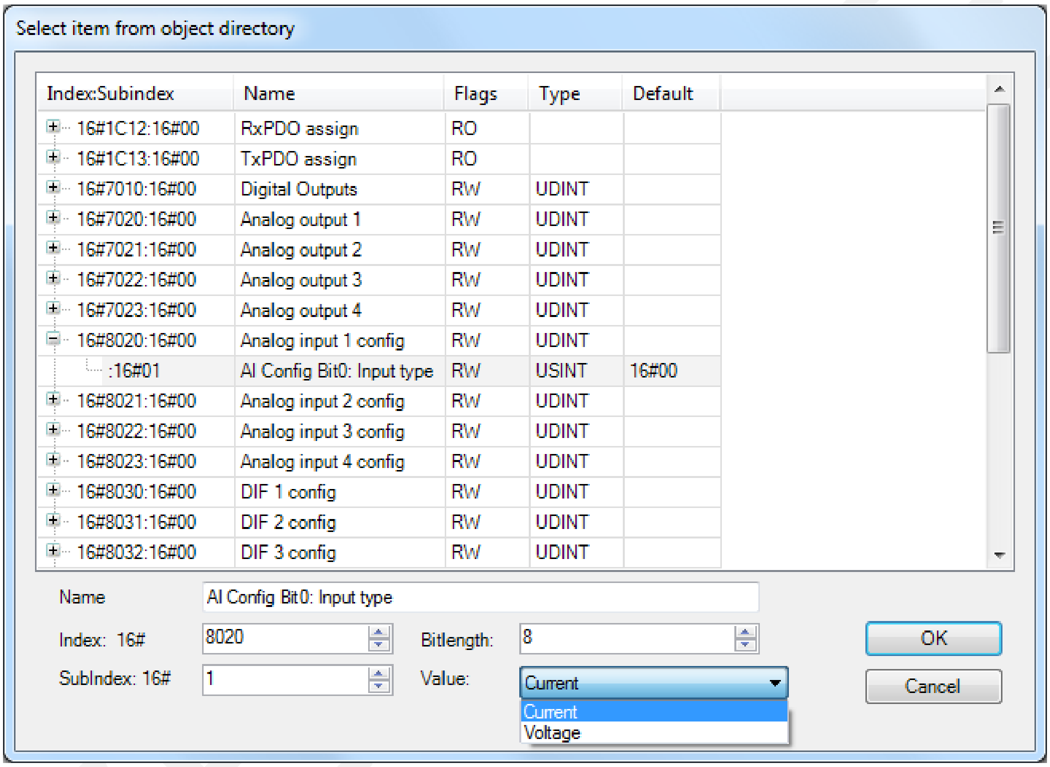
Channel	IEC Address	Type	Value range	Description
Digital outputs				
DO 1...10	`%QX*`	Bit	0/1	0 = sink(default), 1= source
Enable DO	`%QX*`	Bit		Enables digital outputs(default disabled)

Please note each cycle EnableDO must be set TRUE via a variable assiged TRUE.
eg. IOM51_EnableDO := TRUE; to enable the digital output.

Channel	IEC Address	Type	Value range	Description
Analogue output				
AO 1...4	%QW*	INT	-25000...+25000	New setpoint for Analogue Output channel 1...4 -25000: sets the output to -20 mA sink 0 :sets the Output as close as possible to 0 mA +25000 sets the Output to +20 mA
Enable AO	%QX*	BIT		Enables analogue outputs(default disabled)

2.2.3 IOM5·1 – Startup parameters

Analog input 1...4 config



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#1C12:16#00	RxPDO assign	RO		
16#1C13:16#00	TxPDO assign	RO		
16#7010:16#00	Digital Outputs	RW	UDINT	
16#7020:16#00	Analog output 1	RW	UDINT	
16#7021:16#00	Analog output 2	RW	UDINT	
16#7022:16#00	Analog output 3	RW	UDINT	
16#7023:16#00	Analog output 4	RW	UDINT	
16#8020:16#00	Analog input 1 config	RW	UDINT	
:16#01	AI Config Bit0: Input type	RW	USINT	16#00
16#8021:16#00	Analog input 2 config	RW	UDINT	
16#8022:16#00	Analog input 3 config	RW	UDINT	
16#8023:16#00	Analog input 4 config	RW	UDINT	
16#8030:16#00	DIF 1 config	RW	UDINT	
16#8031:16#00	DIF 2 config	RW	UDINT	
16#8032:16#00	DIF 3 config	RW	UDINT	

Name: AI Config Bit0: Input type

Index: 16# 8020

Bitlength: 8

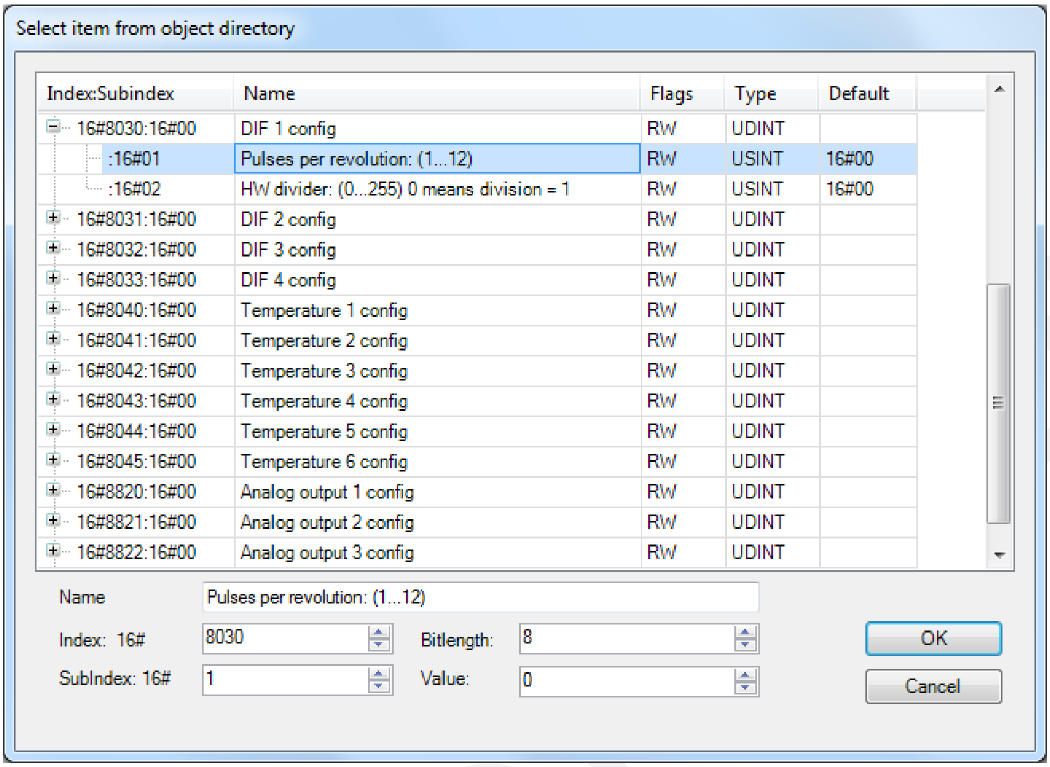
SubIndex: 16# 1

Value: Current

Figure 2.3: Analog input 1...4 config

Channel	Type	Value range	Description
Analog input 1...4 config			
AI Config Bit0: Input type	USINT	0...1	Selects input type: Current(0):-20...+20mA Voltage(1): -10...+10V

DIF 1. . . 4 config



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
⊟ 16#8030:16#00	DIF 1 config	RW	UDINT	
:16#01	Pulses per revolution: (1...12)	RW	USINT	16#00
:16#02	HW divider: (0...255) 0 means division = 1	RW	USINT	16#00
⊞ 16#8031:16#00	DIF 2 config	RW	UDINT	
⊞ 16#8032:16#00	DIF 3 config	RW	UDINT	
⊞ 16#8033:16#00	DIF 4 config	RW	UDINT	
⊞ 16#8040:16#00	Temperature 1 config	RW	UDINT	
⊞ 16#8041:16#00	Temperature 2 config	RW	UDINT	
⊞ 16#8042:16#00	Temperature 3 config	RW	UDINT	
⊞ 16#8043:16#00	Temperature 4 config	RW	UDINT	
⊞ 16#8044:16#00	Temperature 5 config	RW	UDINT	
⊞ 16#8045:16#00	Temperature 6 config	RW	UDINT	
⊞ 16#8820:16#00	Analog output 1 config	RW	UDINT	
⊞ 16#8821:16#00	Analog output 2 config	RW	UDINT	
⊞ 16#8822:16#00	Analog output 3 config	RW	UDINT	

Name: Pulses per revolution: (1...12)

Index: 16# 8030 Bitlength: 8

SubIndex: 16# 1 Value: 0

Figure 2.4: DIF 1 config

Channel	Type	Value range	Description
DIF 1. . . 4 config			
Pulses per revolution: (1. . . 12)	BYTE	1. . . 12 129(1+128). . . 140(12+128)	Number of pulses per revolution for frequency channel 1. . . 4 Bit 7: 0 = HW divider is disabled, 1 = HW divider is enabled
HW divider: (0. . . 255) 0 means division = 1	BYTE	0. . . 255	HW divider setting of frequency input channel 1 to 4 0 : the divider is disabled

The working principal is illustrated on the figure below:

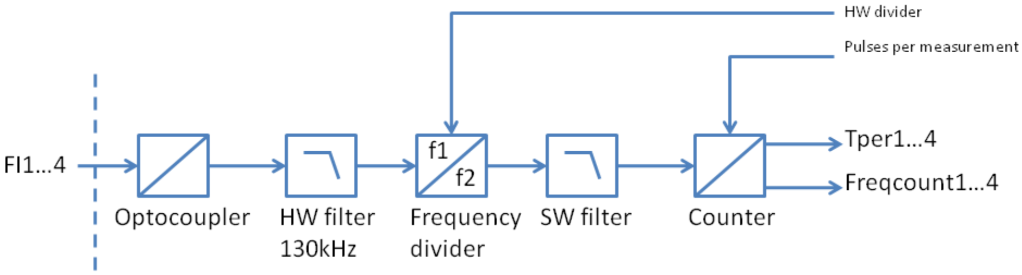


Figure 2.5: Frequency input signal patch

Optocoupler : The highspeed optocoupler reacts on input voltages above 9V.
HW filter : Filters unwanted signals.
HW divider : The HW divider divides down the input frequency to a signal below 1kHz.
SW Filter : Removes all pulses (positive or negative) of the signal after the divider <200us.
Capturer : At a low to high transaction an internal free-running counter is read and stored. Tper1. . . 4 is calculated and Freqcount1. . . 4 increased each pulse or on each N number of pulses.

See below examples on how to use the parameters based on different sensor types.

IOM5·1 – Frequency Input conversion example 2048 pulse encoder

In this example two physical speed measurements, rotor and generator speed are to be captured:
Rotor speed settings:

The physical range is 0..20 [RPM]

At 20 RPM the pulses per seconds are then :

f = 20 [RPM] / 60 * 2048 (encoder pulses per revolution) = 682 [Hz]

This is less than 1 kHz thus we can use the parameters

Pulses per revolution: (1...12): 1

(Meaning it measure between every one pulse)

HW divider: (0...255): 0

Generator speed settings:

Physical range is ca. 0..2000 [RPM]

At 2000 RPM the pulses per seconds are then :

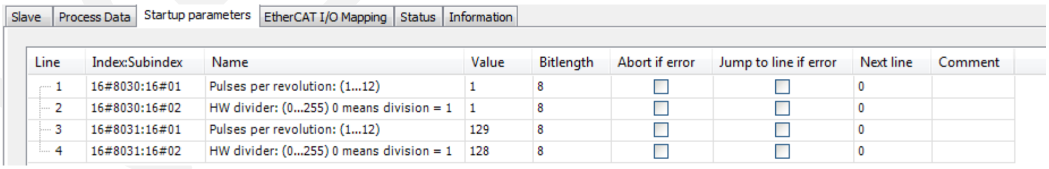
f = 2000 [RPM] / 60 * 2048 (encoder pulses per revolution) = 68200 [Hz]

This is higher than 1 kHz thus we need to use the HW divider settings - easiest is to use highest settings for HW divider:

Pulses per revolution: (1. . . 12) : 129 (1+128) as adding 128 to enable the HW divider.

(Meaning it measure between every one pulse)

HW divider: (0. . . 255) : 128 (Divides the input frequency up to 125 kHz by 128)



Line	Index:Subindex	Name	Value	Bitlength	Abort if error	Jump to line if error	Next line	Comment
1	16#8030:16#01	Pulses per revolution: (1...12)	1	8	□	□	0	
2	16#8030:16#02	HW divider: (0...255) 0 means division = 1	1	8	□	□	0	
3	16#8031:16#01	Pulses per revolution: (1...12)	129	8	□	□	0	
4	16#8031:16#02	HW divider: (0...255) 0 means division = 1	128	8	□	□	0	

Figure 2.6: Startup parameters

The Encoder pulses per revolution(here 2048 encoder pulses per revolution) are changed in the code in the conversion from measured frequency to rotational speed [rpm] calculation.

The Frequency counter input TperX from the IOM 5.1 card returns a count of 800 ns. To convert this to a frequency [Hz] or a rotational speed in [RPM], the calculation follows :

To get a frequency [Hz] the formular is :

```
(* PPR : Pulses per revolution: (1...12)*)
(* HWD : HW divider: (0...255) *)
f [Hz] = 1 / ( TperX * 800 * 10^-9) * PPR * HWD
```

or

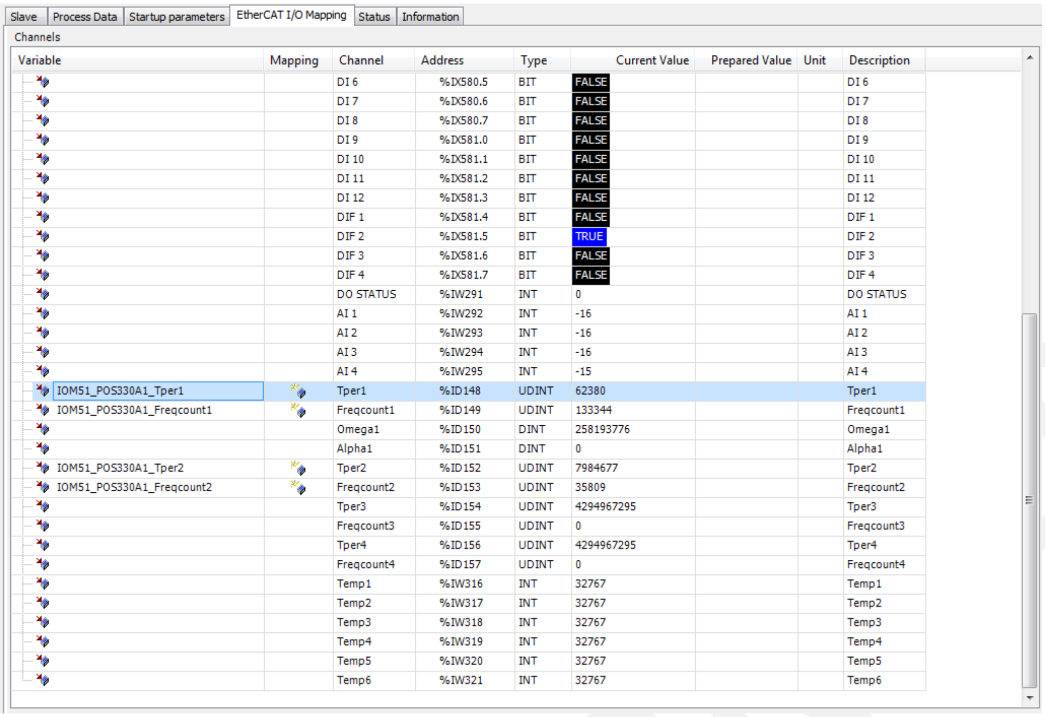
```
(* PPR : Pulses per revolution: (1...12)*)
(* HWD : HW divider: (0...255) *)
f [Hz] = 1250000.0 / TperX * PPR * HWD
```

To get a [rpm] then multiply with 60:

```
Omega [RPM] = f * 60 / EPPR (* EPPR = encoder pulses per revolution *)
```

Program example:

```
VAR
        rRotorSpeed_Hz                  : LREAL;
        rRotorSpeed_RPM                 : LREAL;
        udiPrevFreqcount1               : UDINT;
        uiRotorTimeOutCnt               : UINT;

        rGeneratorSpeed_Hz              : LREAL;
        rGeneratorSpeed_RPM             : LREAL;
        udiPrevFreqcount2               : UDINT;
        uiGeneratorTimeOutCnt   : UINT;
END_VAR
(* Rotor speed *)
IF IOM51_Freqcount1 <> udiPrevFreqcount1 THEN
        IF IOM51_Tper1 <> 16#FFFFFFFF AND IOM51_Tper1 <> 16#00000000 THEN
        (*      PPR: Pulses per revolution 1...12:      1       Measure Tper over each pulse    *)
        (*      HWD: HW divider 0...255:                        1   Divides the input frequency up to 125 kHz by 1 *)
        rRotorSpeed_Hz := 1250000.0 / UDINT_TO_LREAL(IOM51_Tper1) * 1.0 (*PPR*) * 1.0 (*HWD*); (*[Hz]*)
        rRotorSpeed_RPM := 60.0 * rRotorSpeed_Hz / 2048.0 (* encoder pulses per revolution *); (*[RPM]*)
        END_IF
        uiRotorTimeOutCnt := 50;   (* Reset timeout for pulses, 50 * 0.020 sec = 1 sec *)
ELSE
        IF uiRotorTimeOutCnt > 0 THEN
                uiRotorTimeOutCnt := uiRotorTimeOutCnt-1;
        ELSE
                rRotorSpeed_RPM := 0.0;
        END_IF
END_IF
udiPrevFreqcount1 := IOM51_Freqcount1;
(* Generator speed *)
IF IOM51_Freqcount2 <> udiPrevFreqcount2 THEN
     IF IOM51_Tper2 <> 16#FFFFFFFF AND IOM51_Tper2 <> 16#00000000 THEN
    (* PPR: Pulses per revolution 1...12:   129  1 + 128, measure Tper over each pulse,and adding 128 to enable the HW divider. *)
    (* HWD: HW divider 0...255:                  128  Divides the input frequency up to 125 kHz by 128 *)
    rGeneratorSpeed_Hz := 1250000.0 / UDINT_TO_LREAL(IOM51_Tper2) * 1.0 (*PPR*) * 128.0 (*HWD*); (*[Hz]*)
    rGeneratorSpeed_RPM:= 60.0 * rGeneratorSpeed_Hz / 2048.0 (* encoder pulses per revolution *); (*[RPM]*)
    END_IF
    uiGeneratorTimeOutCnt := 50;    (* Reset timeout for pulses, 50 * 0.020 sec = 1 sec *)
ELSE
    IF uiGeneratorTimeOutCnt > 0 THEN
        uiGeneratorTimeOutCnt := uiGeneratorTimeOutCnt-1;
    ELSE
        rGeneratorSpeed_RPM := 0.0;
    END_IF
END_IF
udiPrevFreqcount2 := IOM51_Freqcount2;
```



Variable	Mapping	Channel	Address	Type	Current Value	Prepared Value	Unit	Description
		DI 6	%IX580.5	BIT	FALSE			DI 6
		DI 7	%IX580.6	BIT	FALSE			DI 7
		DI 8	%IX580.7	BIT	FALSE			DI 8
		DI 9	%IX581.0	BIT	FALSE			DI 9
		DI 10	%IX581.1	BIT	FALSE			DI 10
		DI 11	%IX581.2	BIT	FALSE			DI 11
		DI 12	%IX581.3	BIT	FALSE			DI 12
		DIF 1	%IX581.4	BIT	FALSE			DIF 1
		DIF 2	%IX581.5	BIT	TRUE			DIF 2
		DIF 3	%IX581.6	BIT	FALSE			DIF 3
		DIF 4	%IX581.7	BIT	FALSE			DIF 4
		DO STATUS	%IW291	INT	0			DO STATUS
		AI 1	%IW292	INT	-16			AI 1
		AI 2	%IW293	INT	-16			AI 2
		AI 3	%IW294	INT	-16			AI 3
		AI 4	%IW295	INT	-15			AI 4
IOM51_POS330A1_Tper1		Tper1	%ID148	UDINT	62380			Tper1
IOM51_POS330A1_Freqcount1		Freqcount1	%ID149	UDINT	133344			Freqcount1
		Omega1	%ID150	DINT	258193776			Omega1
		Alpha1	%ID151	DINT	0			Alpha1
IOM51_POS330A1_Tper2		Tper2	%ID152	UDINT	7984677			Tper2
IOM51_POS330A1_Freqcount2		Freqcount2	%ID153	UDINT	35809			Freqcount2
		Tper3	%ID154	UDINT	4294967295			Tper3
		Freqcount3	%ID155	UDINT	0			Freqcount3
		Tper4	%ID156	UDINT	4294967295			Tper4
		Freqcount4	%ID157	UDINT	0			Freqcount4
		Temp1	%IW316	INT	32767			Temp1
		Temp2	%IW317	INT	32767			Temp2
		Temp3	%IW318	INT	32767			Temp3
		Temp4	%IW319	INT	32767			Temp4
		Temp5	%IW320	INT	32767			Temp5
		Temp6	%IW321	INT	32767			Temp6

Figure 2.7: Startup parameters

Recommended setting - when using encoder with 2048 pulses per revolution Number of pulses (1...12) : 129 (1 + 128) HW divider (0...255) : 128

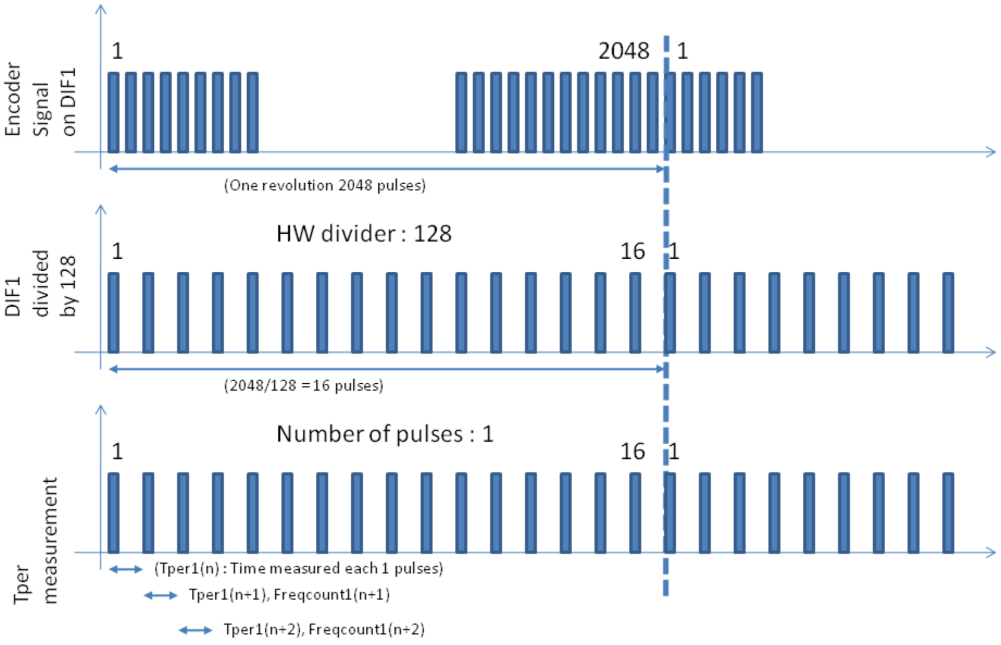


Figure 2.8: Wire setup

IOM – Frequency Input conversion example for inductive sensor based speed detection with 8 holes per revolution

Number of pulses (1...12) : 129 (1 + 128) HW divider (0...255) : 0 (or 1)

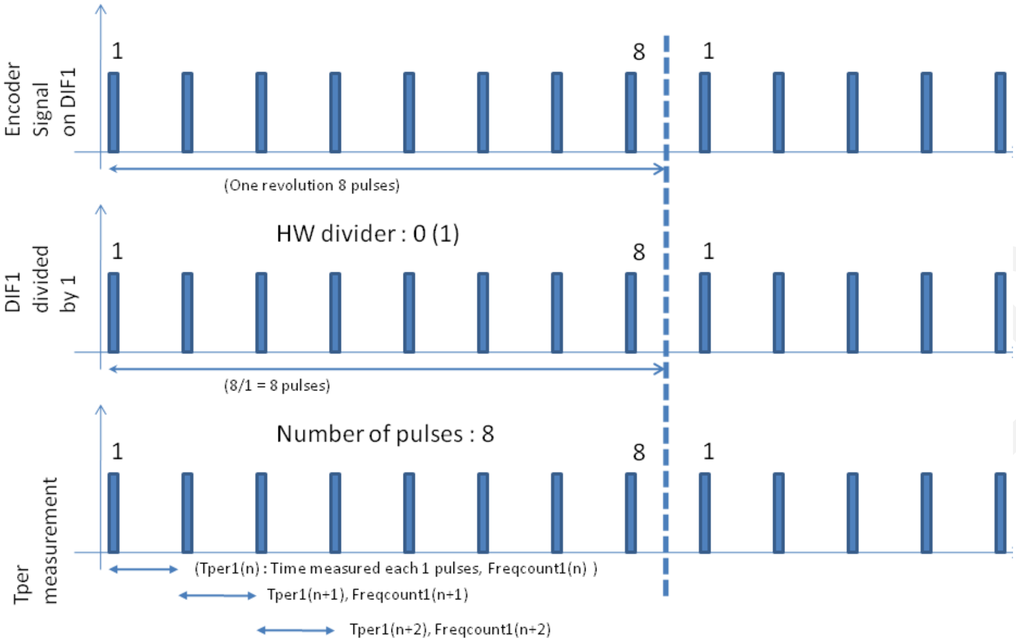

Encoder Signal on DIF1
1
8
1
(One revolution 8 pulses)
DIF1 divided by 1
HW divider : 0 (1)
1
8
1
(8/1 = 8 pulses)
Tper measurement
Number of pulses : 8
1
8
1
(Tper1(n) : Time measured each 1 pulses, Freqcount1(n))
Tper1(n+1), Freqcount1(n+1)
Tper1(n+2), Freqcount1(n+2)

Figure 2.9: Wire setup

The Tper is updated and Freqcount is increased by 1, at each detected pulse.

Here the recommended settings or inductive sensor based speed detection with 8 holes per revolution are to change the "Number of pulses" to 8. The advantage by setting the update with the detection by each 8 holes, is that Tper is updated each revolution. Number of pulses (1...12) : 136 (8 + 128) HW divider (0...255) : 0 (or 1)

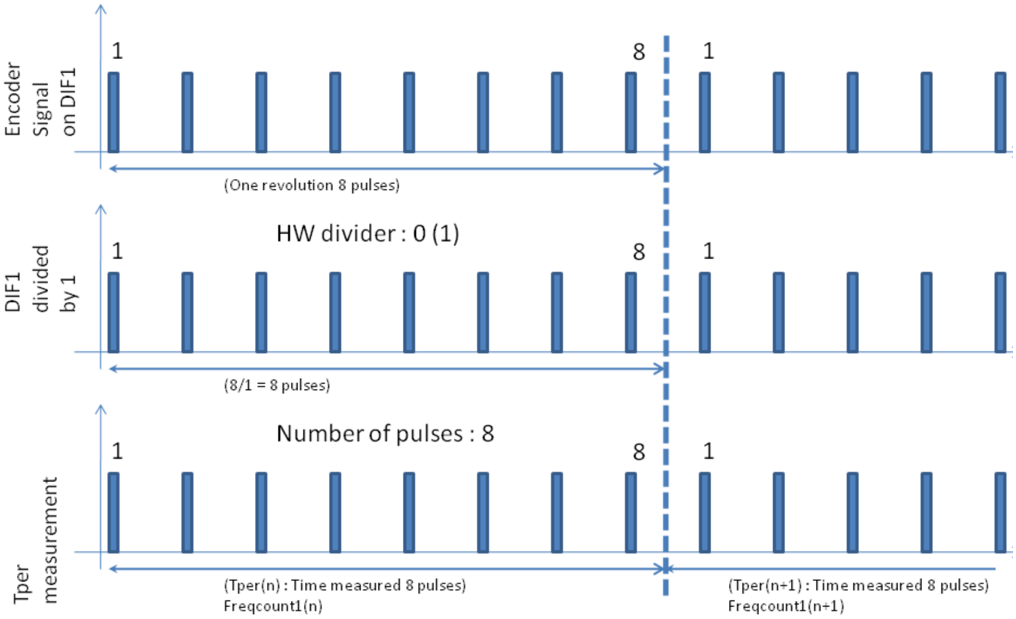
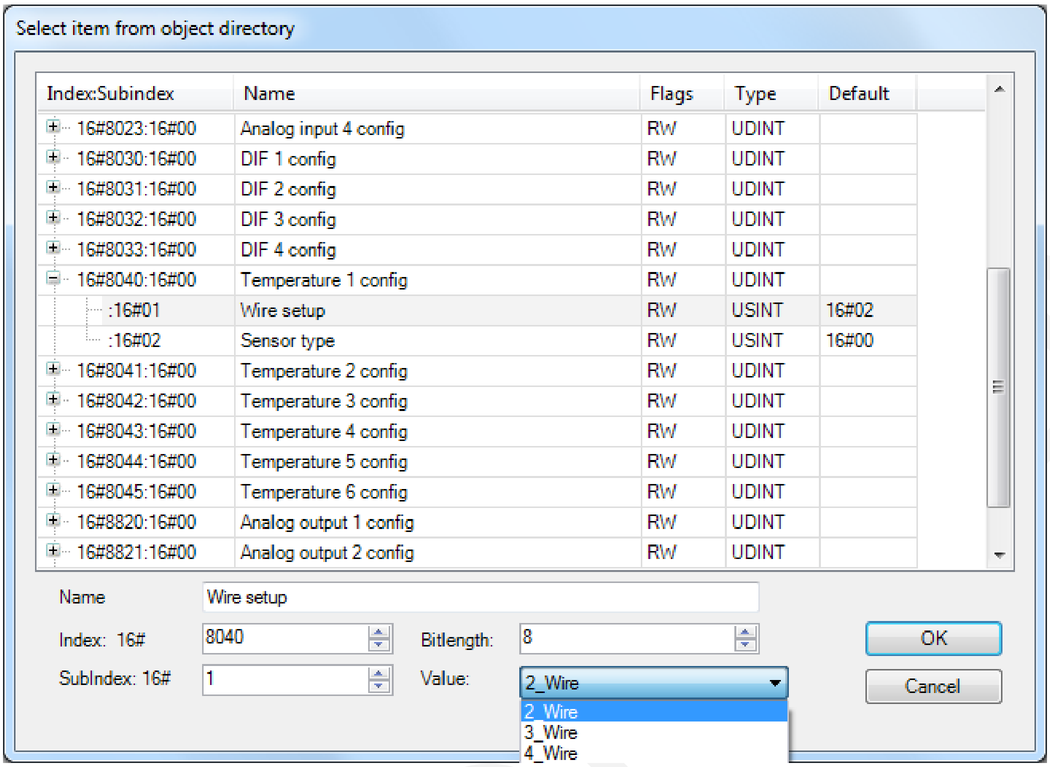


Figure 2.10: Wire setup

Temperature 1. . . 6 config



Select item from object directory

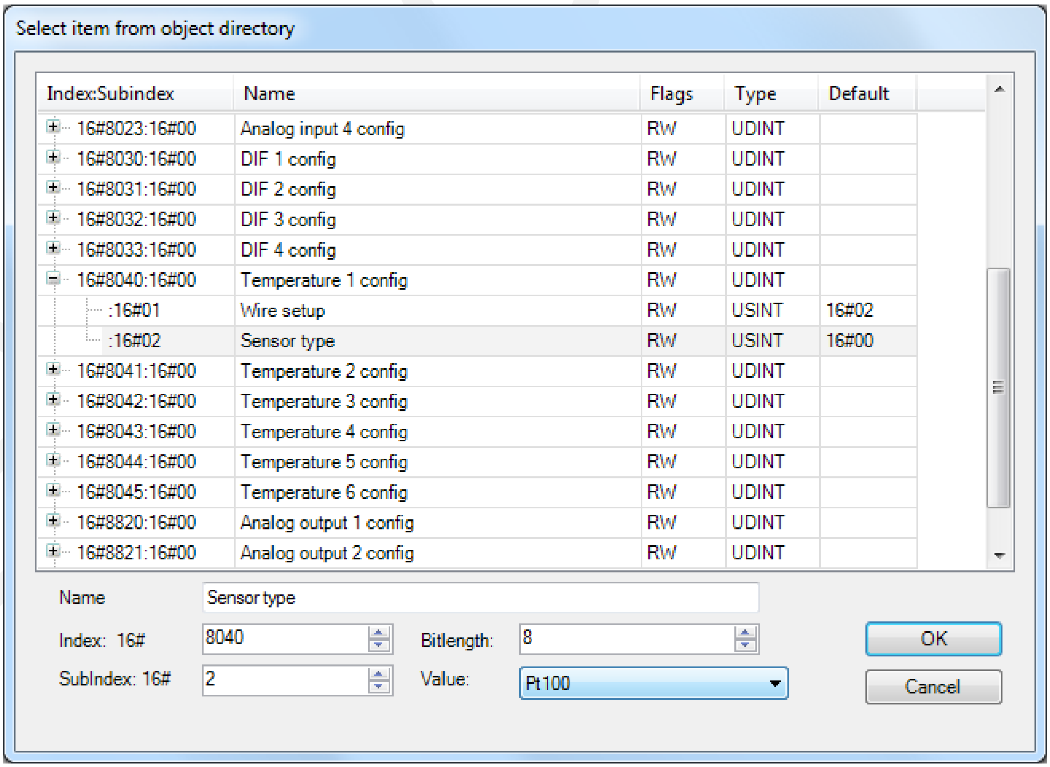
Index:Subindex	Name	Flags	Type	Default
16#8023:16#00	Analog input 4 config	RW	UDINT	
16#8030:16#00	DIF 1 config	RW	UDINT	
16#8031:16#00	DIF 2 config	RW	UDINT	
16#8032:16#00	DIF 3 config	RW	UDINT	
16#8033:16#00	DIF 4 config	RW	UDINT	
16#8040:16#00	Temperature 1 config	RW	UDINT	
:16#01	Wire setup	RW	USINT	16#02
:16#02	Sensor type	RW	USINT	16#00
16#8041:16#00	Temperature 2 config	RW	UDINT	
16#8042:16#00	Temperature 3 config	RW	UDINT	
16#8043:16#00	Temperature 4 config	RW	UDINT	
16#8044:16#00	Temperature 5 config	RW	UDINT	
16#8045:16#00	Temperature 6 config	RW	UDINT	
16#8820:16#00	Analog output 1 config	RW	UDINT	
16#8821:16#00	Analog output 2 config	RW	UDINT	

Name: Wire setup

Index: 16# 8040 Bitlength: 8

SubIndex: 16# 1 Value: 2_Wire

Figure 2.11: Wire setup



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#8023:16#00	Analog input 4 config	RW	UDINT	
16#8030:16#00	DIF 1 config	RW	UDINT	
16#8031:16#00	DIF 2 config	RW	UDINT	
16#8032:16#00	DIF 3 config	RW	UDINT	
16#8033:16#00	DIF 4 config	RW	UDINT	
16#8040:16#00	Temperature 1 config	RW	UDINT	
:16#01	Wire setup	RW	USINT	16#02
:16#02	Sensor type	RW	USINT	16#00
16#8041:16#00	Temperature 2 config	RW	UDINT	
16#8042:16#00	Temperature 3 config	RW	UDINT	
16#8043:16#00	Temperature 4 config	RW	UDINT	
16#8044:16#00	Temperature 5 config	RW	UDINT	
16#8045:16#00	Temperature 6 config	RW	UDINT	
16#8820:16#00	Analog output 1 config	RW	UDINT	
16#8821:16#00	Analog output 2 config	RW	UDINT	

Name: Sensor type

Index: 16# 8040 Bitlength: 8

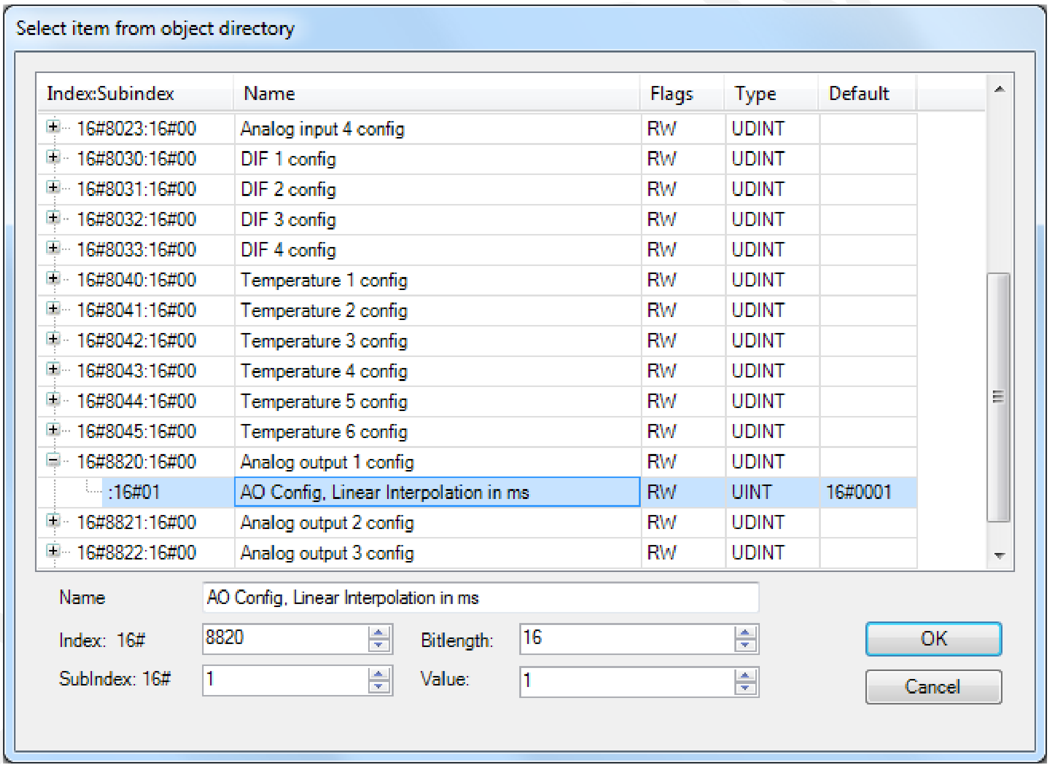
SubIndex: 16# 2 Value: Pt100

Figure 2.12: Sensor type

Channel	Type	Value range	Description
Temperature 1. . . 6 config			
Wire setup	USINT	0. . . 3	2_Wire: PT100/1000 2 wire 3_Wire: PT100/1000 3 wire 4_Wire: PT100/1000 4 wire
Sensor type	USINT	0. . . 2	Pt100: PT100 sensor Pt1000: PT1000 sensor NiCr: NiCr sensor

Analog output 1. . . 4 config

Channel	Type	Value range	Description
Analog output 1. . . 4 config			
AO Config.	UINT	0. . . 1000	Linear Interpolation in ms for DA Output channel


Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#8023:16#00	Analog input 4 config	RW	UDINT	
16#8030:16#00	DIF 1 config	RW	UDINT	
16#8031:16#00	DIF 2 config	RW	UDINT	
16#8032:16#00	DIF 3 config	RW	UDINT	
16#8033:16#00	DIF 4 config	RW	UDINT	
16#8040:16#00	Temperature 1 config	RW	UDINT	
16#8041:16#00	Temperature 2 config	RW	UDINT	
16#8042:16#00	Temperature 3 config	RW	UDINT	
16#8043:16#00	Temperature 4 config	RW	UDINT	
16#8044:16#00	Temperature 5 config	RW	UDINT	
16#8045:16#00	Temperature 6 config	RW	UDINT	
16#8820:16#00	Analog output 1 config	RW	UDINT	
:16#01	AO Config, Linear Interpolation in ms	RW	UINT	16#0001
16#8821:16#00	Analog output 2 config	RW	UDINT	
16#8822:16#00	Analog output 3 config	RW	UDINT	

Name: AO Config, Linear Interpolation in ms

Index: 16# 8820

Bitlength: 16

SubIndex: 16# 1

Value: 1

Figure 2.13: Analog output config

The linear interpolation can be set individually for the outputs. This means that the outputs can be changed linear over time instead of a step.
See example below, where output 1 is changed from 10 to 20 mA in a step and output 2 is changed from 10 to 20 mA linear over time.

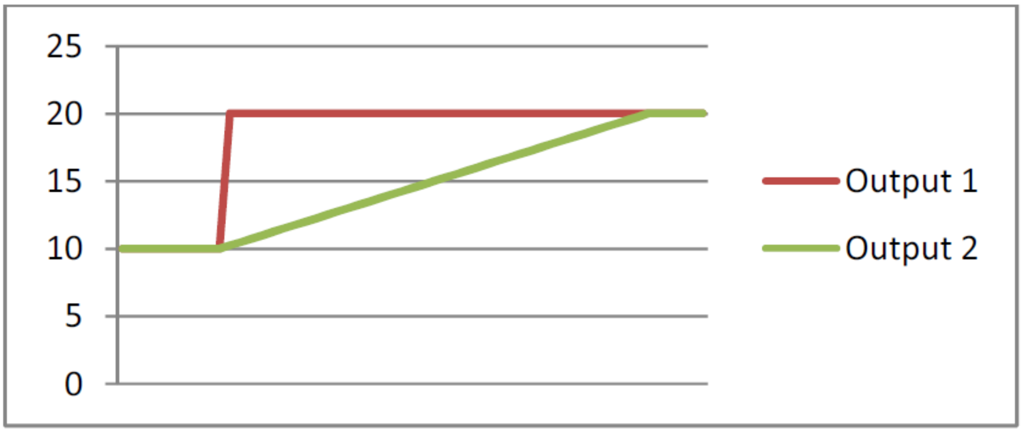


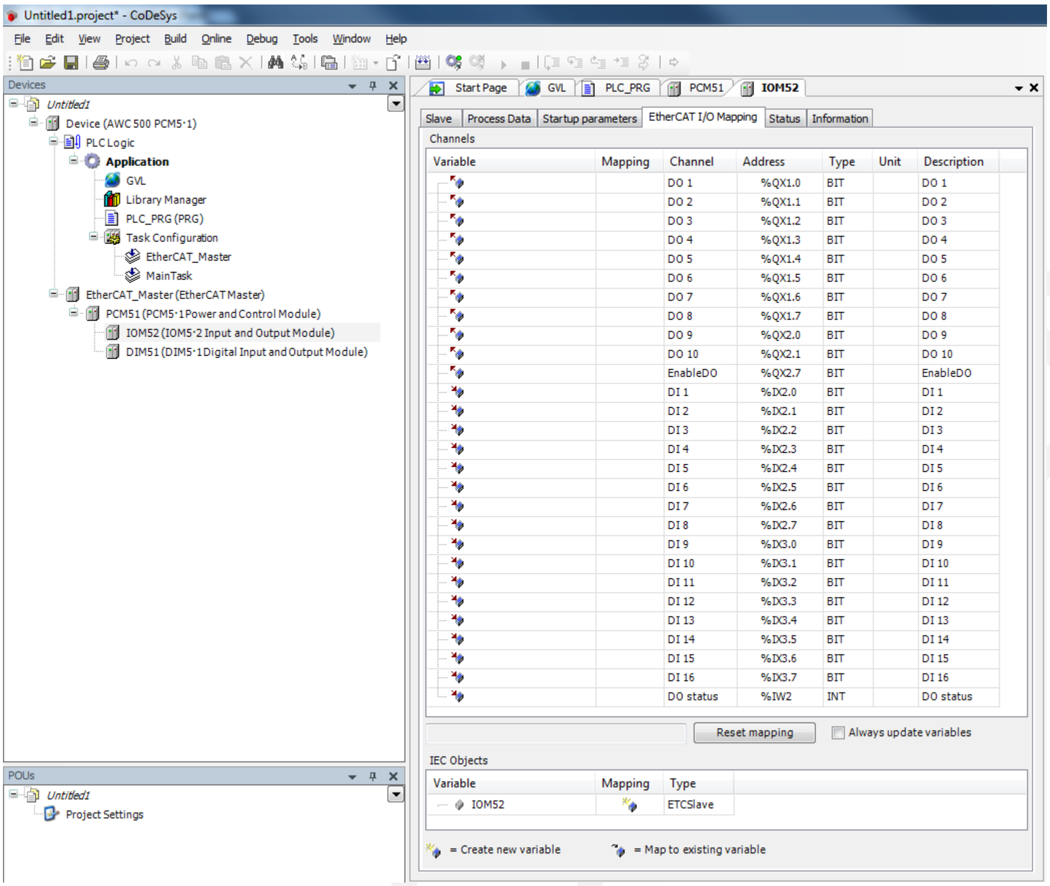
Figure 2.14: Linear interpolation

2.3 IOM5·2 – Input and Output Module

2.3.1 IOM5·2 – Terminal Description

Terminals		CODESYS Process Variable	Desciption
23	DI1	DI 1	Digital input 1
24	DI2	DI 2	Digital input 2
25	DI3	DI 3	Digital input 3
26	DI4	DI 4	Digital input 4
27	DI5	DI 5	Digital input 5
28	DI6	DI 6	Digital input 6
29	DI COM		Digital common input reference supply (DI1–DI12). 24 V for NPN input signal, GND for PNP input signal. Note: Terminal 29 and 69 are internally connected
63	DI7	DI 7	Digital input 7
64	DI8	DI 8	Digital input 8
65	DI9	DI 9	Digital input 9
66	DI10	DI 10	Digital input 10
67	DI11	DI 11	Digital input 11
68	DI12	DI 12	Digital input 12
69	DI COM		Digital common input reference supply (DI1–DI12). 24 V for NPN input signal, GND for PNP input signal. Note: Terminal 29 and 69 are internally connected
30	DI13+	DI 13	Digital input 13+
31	DI13-	DI 13	Digital input 13-
32	DI14+	DI 14	Digital input 14+
33	DI14-	DI 14	Digital input 14-
70	DI15+	DI 15	Digital input 15+
71	DI15-	DI 15	Digital input 15-
72	DI16+	DI 16	Digital input 16+
73	DI16-	DI 16	Digital input 16-
34	DO SUP+		24 V digital output supply. Note: Terminals 34 and 74 are internally connected
35	DO1	DO1	Digital Output 1
36	DO2	DO2	Digital Output 2
37	DO3	DO3	Digital Output 3
38	DO4	DO4	Digital Output 4
39	DO5	DO5	Digital Output 5
40	DO SUP-		GND digital output supply. Note: Terminals 40 and 80 are internally connected
74	DO SUP+		24 V digital output supply. Note: Terminals 34 and 74 are internally connected
75	DO6	DO6	Digital Output 6
76	DO7	DO7	Digital Output 7
77	DO8	DO8	Digital Output 8
78	DO9	DO9	Digital Output 9
79	DO10	DO10	Digital Output 10
80	DO SUP-		GND digital output supply. Note: Terminals 40 and 80 are internally connected

2.3.2 IOM5·2 –EtherCAT I/O Mapping



- Untitled1
  - Device (AWC 500 PCM5·1)
    - PLC Logic
      - Application
        - GVL
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - EtherCAT_Master
          - MainTask
    - EtherCAT_Master (EtherCAT Master)
      - PCM51 (PCM5·1 Power and Control Module)
        - IOM52 (IOM5·2 Input and Output Module)
        - DIM51 (DIM5·1 Digital Input and Output Module)

- Untitled1
  - Project Settings

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		DO 1	%QX1.0	BIT		DO 1
		DO 2	%QX1.1	BIT		DO 2
		DO 3	%QX1.2	BIT		DO 3
		DO 4	%QX1.3	BIT		DO 4
		DO 5	%QX1.4	BIT		DO 5
		DO 6	%QX1.5	BIT		DO 6
		DO 7	%QX1.6	BIT		DO 7
		DO 8	%QX1.7	BIT		DO 8
		DO 9	%QX2.0	BIT		DO 9
		DO 10	%QX2.1	BIT		DO 10
		EnableDO	%QX2.7	BIT		EnableDO
		DI 1	%IX2.0	BIT		DI 1
		DI 2	%IX2.1	BIT		DI 2
		DI 3	%IX2.2	BIT		DI 3
		DI 4	%IX2.3	BIT		DI 4
		DI 5	%IX2.4	BIT		DI 5
		DI 6	%IX2.5	BIT		DI 6
		DI 7	%IX2.6	BIT		DI 7
		DI 8	%IX2.7	BIT		DI 8
		DI 9	%IX3.0	BIT		DI 9
		DI 10	%IX3.1	BIT		DI 10
		DI 11	%IX3.2	BIT		DI 11
		DI 12	%IX3.3	BIT		DI 12
		DI 13	%IX3.4	BIT		DI 13
		DI 14	%IX3.5	BIT		DI 14
		DI 15	%IX3.6	BIT		DI 15
		DI 16	%IX3.7	BIT		DI 16
		DO status	%IW2	INT		DO status

□ Always update variables

IEC Objects

Variable	Mapping	Type
IOM52		ETCSlave

= Create new variable = Map to existing variable

Figure 2.15: IOM5·2 – EtherCAT I/O Mapping

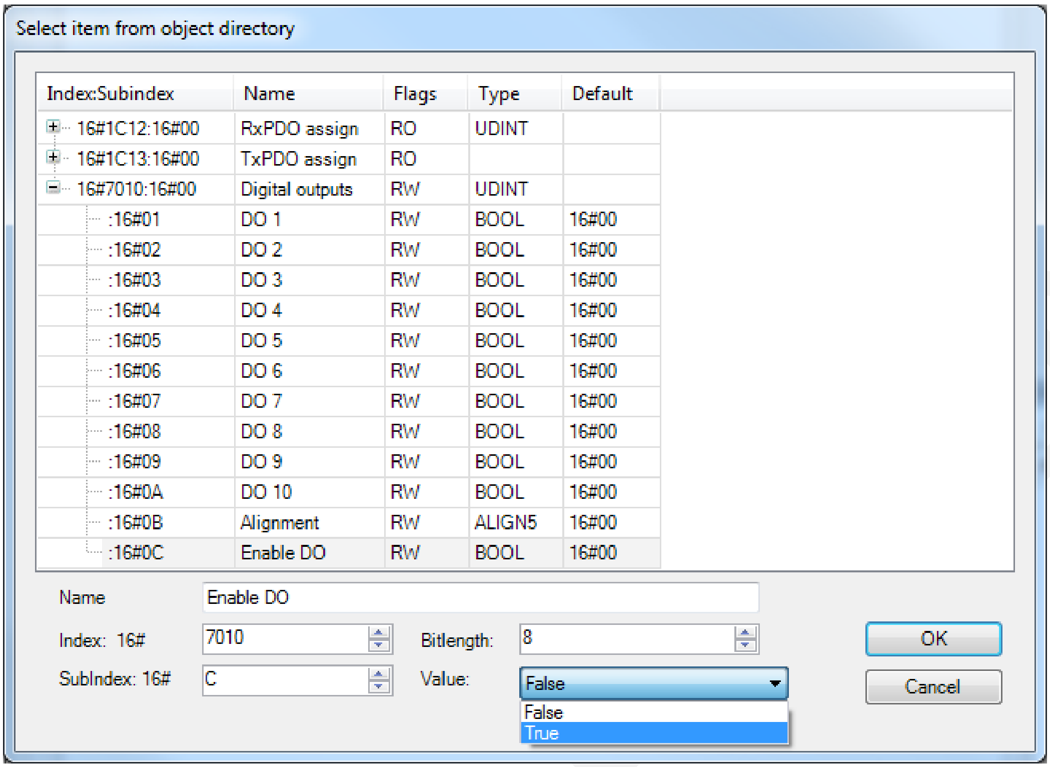
Channel	IEC Address	Type	Value range	Description
Digital Inputs				
DI 1...16	`%IX*`	BIT	0/1	Filtered state of digital inputs 1...16
DO Status	`%IW*`	INT	0...3	**Digital output status diagnostics** **Bit 0** indicate the status of the digital outputs. 0: An error is indicated if one of the following conditions is filled: • Supply voltage is below 8 V • Supply voltage is above 37 V • Error is reported on one of the digital output drivers. (Over-current or over-temperature) After power-up the digital output (**EnableDO**) has to be enabled before error-bit is reset. If an over-current or over-temperature status has occurred, the digital outputs (**EnableDO**) have to be disabled and then enabled again to clear the error state. 1 : Digital Outputs are OK **Bit 1** indicates state of IOM calibration. 0 : NOT calibrated, 1 : Calibrated

The digital inputs do not require any further configuration.

Channel	IEC Address	Type	Value range	Description
Digital outputs				
DO 1...10	`%QX*`	Bit	0/1	0 = sink(default), 1= source
Enable DO	`%QX*`	Bit		Enables digital outputs(default disabled)

Please note each cycle EnableDO must be set TRUE via a variable assiged TRUE.
eg. IOM52_EnableDO := TRUE; to enable the digital output.

2.3.3 IOM5·2 – Startup parameters



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#1C12:16#00	RxPDO assign	RO	UDINT	
16#1C13:16#00	TxPDO assign	RO		
16#7010:16#00	Digital outputs	RW	UDINT	
:16#01	DO 1	RW	BOOL	16#00
:16#02	DO 2	RW	BOOL	16#00
:16#03	DO 3	RW	BOOL	16#00
:16#04	DO 4	RW	BOOL	16#00
:16#05	DO 5	RW	BOOL	16#00
:16#06	DO 6	RW	BOOL	16#00
:16#07	DO 7	RW	BOOL	16#00
:16#08	DO 8	RW	BOOL	16#00
:16#09	DO 9	RW	BOOL	16#00
:16#0A	DO 10	RW	BOOL	16#00
:16#0B	Alignment	RW	ALIGN5	16#00
:16#0C	Enable DO	RW	BOOL	16#00

Name: Enable DO

Index: 16# 7010

Bitlength: 8

SubIndex: 16# C

Value: False

Figure 2.16: IOM5·2 enable digital outputs from startup parameters

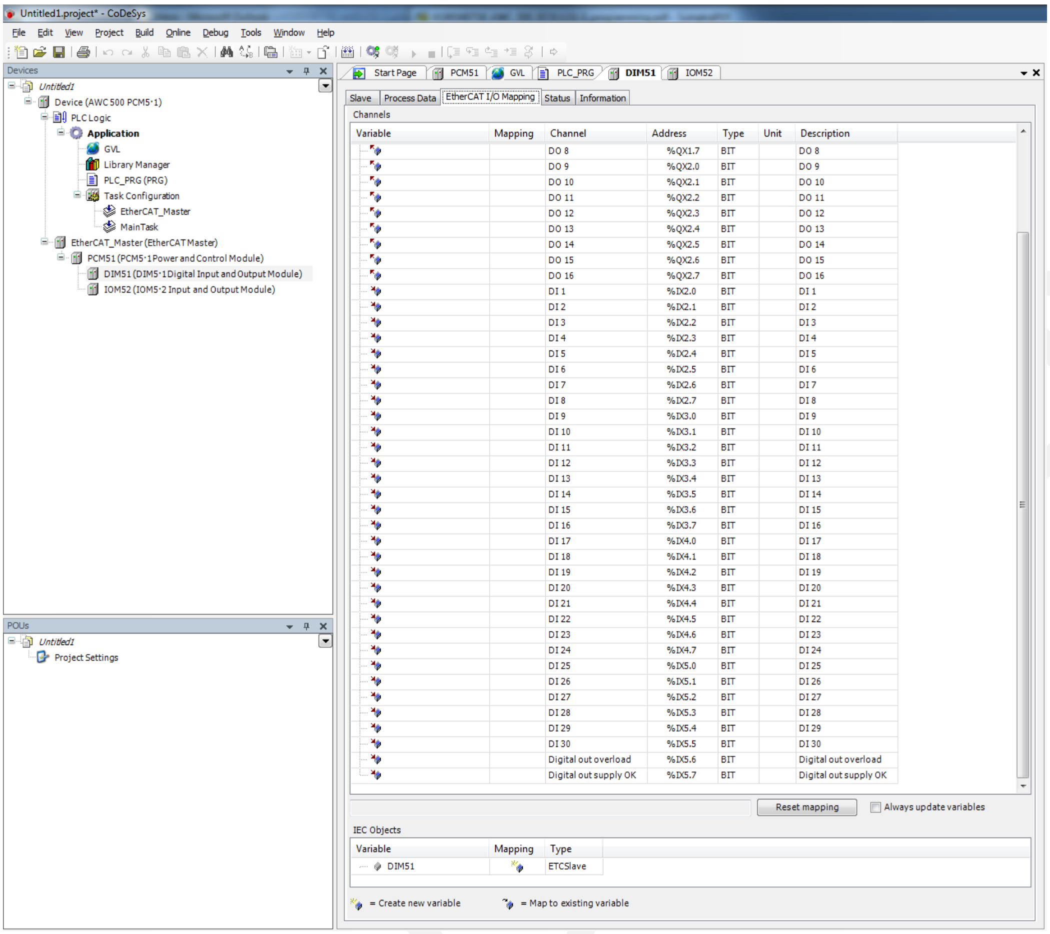
The digital outputs can be enabled from the startup parameters by setting `Enable DO` to `TRUE`, see Figure 2.16.

2.4 DIM5·1 – Input and Output Module

2.4.1 DIM5·1 – Terminal Description

Terminals		CODESYS Process Variable	Desciption
1	DO A SUP+		24 V digital output supply for DO1-DO8; Group A
2	DO1	DO1	Digital Output 1
3	DO2	DO2	Digital Output 2
4	DO3	DO3	Digital Output 3
5	DO4	DO4	Digital Output 4
6	DO5	DO5	Digital Output 5
7	DO6	DO6	Digital Output 6
8	DO7	DO7	Digital Output 7
9	DO8	DO8	Digital Output 8
10	DO A SUP-		GND digital output supply for DO1-DO8; Group A
38	DO A SUP+		24 V digital output supply for DO9-D16; Group B
39	DO9	DO9	Digital Output 9
40	DO10	DO10	Digital Output 10
41	DO11	DO11	Digital Output 11
42	DO12	DO12	Digital Output 12
43	DO13	DO13	Digital Output 13
44	DO14	DO14	Digital Output 14
45	DO15	DO15	Digital Output 15
46	DO16	DO16	Digital Output 16
47	DO A SUP-		GND digital output supply for DO9-D16; Group B
21	DI1	DI 1	Digital input 1
22	DI2	DI 2	Digital input 2
23	DI3	DI 3	Digital input 3
24	DI4	DI 4	Digital input 4
25	DI5	DI 5	Digital input 5
26	DI6	DI 6	Digital input 6
27	DI7	DI 7	Digital input 7
28	DI8	DI 8	Digital input 8
29	DI A COM		Digital common input reference supply (DI1-DI8). 24 V for NPN input signal, GND for PNP input signal.
58	DI9	DI 9	Digital input 9
59	DI10	DI 10	Digital input 10
60	DI11	DI 11	Digital input 11
61	DI12	DI 12	Digital input 12
62	DI13	DI 13	Digital input 13
63	DI14	DI 14	Digital input 14
64	DI15	DI 15	Digital input 15
65	DI16	DI 16	Digital input 16
66	DI B COM		Digital common input reference supply (DI9-DI16). 24 V for NPN input signal, GND for PNP input signal.
30	DI17	DI 17	Digital input 17
31	DI18	DI 18	Digital input 18
32	DI19	DI 19	Digital input 19
33	DI20	DI 20	Digital input 20
34	DI21	DI 21	Digital input 21
35	DI22	DI 22	Digital input 22
36	DI23	DI 23	Digital input 23
37	DI C COM		Digital common input reference supply (DI17-DI23). 24 V for NPN input signal, GND for PNP input signal.
67	DI24	DI 24	Digital input 24
68	DI25	DI 25	Digital input 25
69	DI26	DI 26	Digital input 26
70	DI27	DI 27	Digital input 27
71	DI28	DI 28	Digital input 28
72	DI29	DI 29	Digital input 29
73	DI30	DI 30	Digital input 30
74	DI D COM		Digital common input reference supply (DI24-D30). 24 V for NPN input signal, GND for PNP input signal.

2.4.2 DIM5·1 –EtherCAT I/O Mapping



Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		DO 8	%QX1.7	BIT		DO 8
		DO 9	%QX2.0	BIT		DO 9
		DO 10	%QX2.1	BIT		DO 10
		DO 11	%QX2.2	BIT		DO 11
		DO 12	%QX2.3	BIT		DO 12
		DO 13	%QX2.4	BIT		DO 13
		DO 14	%QX2.5	BIT		DO 14
		DO 15	%QX2.6	BIT		DO 15
		DO 16	%QX2.7	BIT		DO 16
		DI 1	%IX2.0	BIT		DI 1
		DI 2	%IX2.1	BIT		DI 2
		DI 3	%IX2.2	BIT		DI 3
		DI 4	%IX2.3	BIT		DI 4
		DI 5	%IX2.4	BIT		DI 5
		DI 6	%IX2.5	BIT		DI 6
		DI 7	%IX2.6	BIT		DI 7
		DI 8	%IX2.7	BIT		DI 8
		DI 9	%IX3.0	BIT		DI 9
		DI 10	%IX3.1	BIT		DI 10
		DI 11	%IX3.2	BIT		DI 11
		DI 12	%IX3.3	BIT		DI 12
		DI 13	%IX3.4	BIT		DI 13
		DI 14	%IX3.5	BIT		DI 14
		DI 15	%IX3.6	BIT		DI 15
		DI 16	%IX3.7	BIT		DI 16
		DI 17	%IX4.0	BIT		DI 17
		DI 18	%IX4.1	BIT		DI 18
		DI 19	%IX4.2	BIT		DI 19
		DI 20	%IX4.3	BIT		DI 20
		DI 21	%IX4.4	BIT		DI 21
		DI 22	%IX4.5	BIT		DI 22
		DI 23	%IX4.6	BIT		DI 23
		DI 24	%IX4.7	BIT		DI 24
		DI 25	%IX5.0	BIT		DI 25
		DI 26	%IX5.1	BIT		DI 26
		DI 27	%IX5.2	BIT		DI 27
		DI 28	%IX5.3	BIT		DI 28
		DI 29	%IX5.4	BIT		DI 29
		DI 30	%IX5.5	BIT		DI 30
		Digital out overload	%IX5.6	BIT		Digital out overload
		Digital out supply OK	%IX5.7	BIT		Digital out supply OK

IEC Objects

Variable	Mapping	Type
DIM51		ETCSlave

= Create new variable   = Map to existing variable

Figure 2.17: DIM5·1 – EtherCAT I/O Mapping

Channel	IEC Address	Type	Value range	Description
Digital inputs				
DI 1. . . 30	%IX*	BIT	0/1	Filtered state of digital inputs 1. . . 30
Digital out overload	%IX*	BIT	0/1	Overload state of the digital outputs
Digital out supply OK	%IX*	BIT	0/1	State of the digital output supply

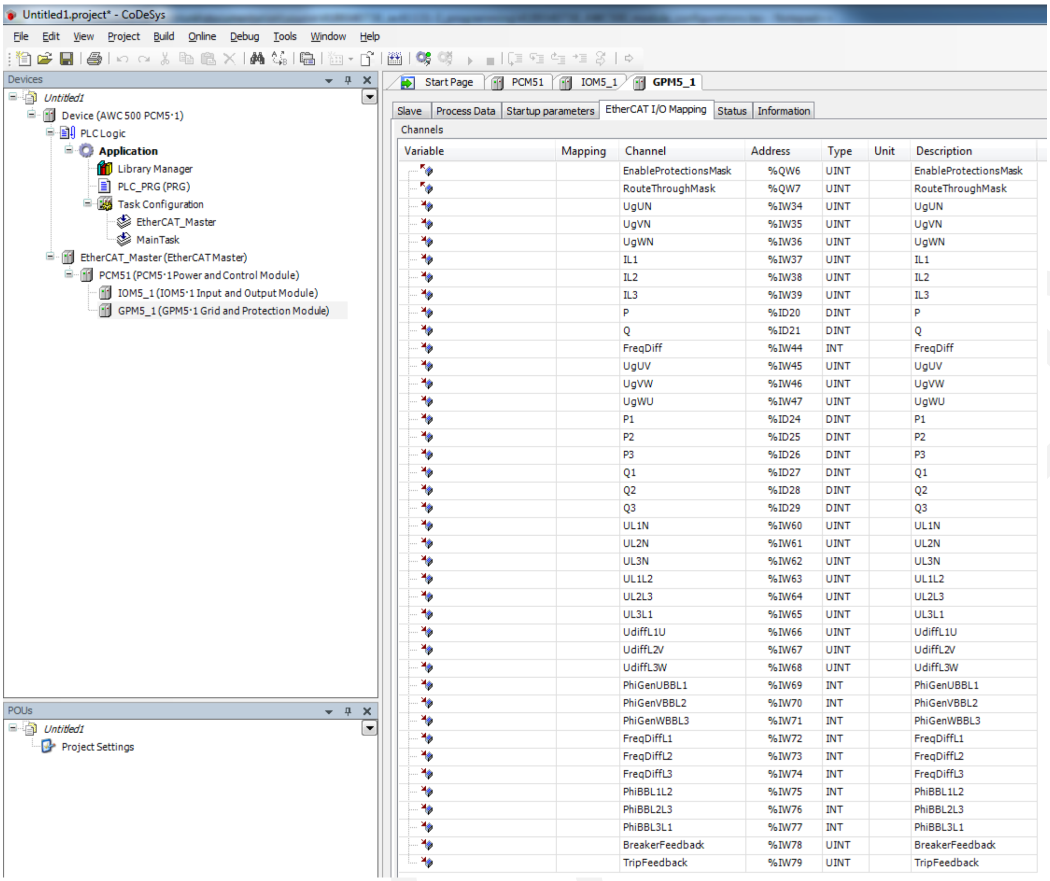
Channel	IEC Address	Type	Value range	Description
Digital outputs				
DO 1. . . 16	%QX*	BIT	0/1	0 = sink(default), 1= source

2.5 GPM5·1 – Grid Protection Module

2.5.1 GPM5·1 – Terminal Description

Terminal		Description	CoDeSys Process Variable
1	S1	Current L1	IL1
2	S2		
3	S1	Current L2	IL2
4	S2		
5	S1	Current L3	IL3
6	S2		
7	L1	Voltage L1	UgUN
8	L2	Voltage L2	UgVN
9	L3	Voltage L3	UgWN
10	N	Neutral	
11	L1	Voltage L1	UL1N
12	L2	Voltage L2	UL2N
13	L3	Voltage L3	UL3N
14	N	Neutral	
		Breaker ON/OFF	
15		Breaker On	SyncOK
16			
17		Breaker Off	
18			
		Feedback	
19	ON	Breaker FB On	
20	COM.	Common	
21	OFF	Breaker FB Off	

2.5.2 GPM5·1 - EtherCAT I/O Mapping



Variable	Mapping	Channel	Address	Type	Unit	Description
		EnableProtectionsMask	%QW6	UINT		EnableProtectionsMask
		RouteThroughMask	%QW7	UINT		RouteThroughMask
		UgUN	%IW34	UINT		UgUN
		UgVN	%IW35	UINT		UgVN
		UgWN	%IW36	UINT		UgWN
		IL1	%IW37	UINT		IL1
		IL2	%IW38	UINT		IL2
		IL3	%IW39	UINT		IL3
		P	%ID20	DINT		P
		Q	%ID21	DINT		Q
		FreqDiff	%IW44	INT		FreqDiff
		UgUV	%IW45	UINT		UgUV
		UgVW	%IW46	UINT		UgVW
		UgWU	%IW47	UINT		UgWU
		P1	%ID24	DINT		P1
		P2	%ID25	DINT		P2
		P3	%ID26	DINT		P3
		Q1	%ID27	DINT		Q1
		Q2	%ID28	DINT		Q2
		Q3	%ID29	DINT		Q3
		UL1N	%IW60	UINT		UL1N
		UL2N	%IW61	UINT		UL2N
		UL3N	%IW62	UINT		UL3N
		UL1L2	%IW63	UINT		UL1L2
		UL2L3	%IW64	UINT		UL2L3
		UL3L1	%IW65	UINT		UL3L1
		UdiffL1U	%IW66	UINT		UdiffL1U
		UdiffL2V	%IW67	UINT		UdiffL2V
		UdiffL3W	%IW68	UINT		UdiffL3W
		PhiGenUBBL1	%IW69	INT		PhiGenUBBL1
		PhiGenVBBL2	%IW70	INT		PhiGenVBBL2
		PhiGenWBBL3	%IW71	INT		PhiGenWBBL3
		FreqDiffL1	%IW72	INT		FreqDiffL1
		FreqDiffL2	%IW73	INT		FreqDiffL2
		FreqDiffL3	%IW74	INT		FreqDiffL3
		PhiBBL1L2	%IW75	INT		PhiBBL1L2
		PhiBBL2L3	%IW76	INT		PhiBBL2L3
		PhiBBL3L1	%IW77	INT		PhiBBL3L1
		BreakerFeedback	%IW78	UINT		BreakerFeedback
		TripFeedback	%IW79	UINT		TripFeedback

Figure 2.18: GPM5·1 - EtherCAT I/O Mapping

Channel	IEC Address	Type	Value range	Description
EnableProtectionMask	%QW*	UINT		EnableProtectionMask
RouteThroughMask	%IW*	UINT		RouteThroughMask

Channel	IEC Address	Type	Value range	Description
Generator voltages (Star)				Voltages UgU,V,WN : a value of 27027 means Un (either 240/sqr(3)Vrms or 690/sqr(3)Vrms is present)
UgUN	%IW*	UINT	0. . . 65535	Normalized Voltage between Generator U-N
UgVN	%IW*	UINT	0. . . 65535	Normalized Voltage between Generator V-N
UgWN	%IW*	UINT	0. . . 65535	Normalized Voltage between Generator W-N

Channel	IEC Address	Type	Value range	Description
Generator currents				Currents IL1,2,3 : a value of 15604 means In (either 1 Arms or 5 Arms) is present
IL1	%IW*	UINT		Normalized current in UN/L1
IL2	%IW*	UINT		Normalized current in VN/L2
IL3	%IW*	UINT		Normalized current in WN/L3

Channel	IEC Address	Type	Value range	Description
Active and reactive power				Real/Blindpower Psum, Qsum : a value of 365218308 means Un * In * sqr(3) is present in all 3 phases togeather
P	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Realpower in all 3 Phases (UgUN * IUL1 * cos(Phi UI) + UgVN * IVL2 *cos()2 + UgWN * IWL3 * cos()3) / (sqr(3) * Un * In) * 365218308
Q	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Blindpower in all 3 Phases (UgUN * IUL1 * sin(Phi UI)1 + UgVN * IVL2 * sin()2 + UgWN * IWL3 * sin()3) / (sqr(3) * Un * In) * 365218308
FreqDiff	`%IW*`	INT	-32768...32767	Normalized frequency difference between Bus Bar measurment and nominal frequency (50 Hz or 60 Hz) -32768 : frequency too low for measurement 0 : frequency is equal nominal frequency (either 50 Hz or 60 Hz) +32767 : indicates double frequency (either 100 Hz of 120 Hz), frequency is too high to be measured correct

Channel	IEC Address	Type	Value range	Description
Active power				P1,2,3, Q1,2,3 : 121739436 : Un / sqr(3) * In is present
P1	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Realpower in Phase 1 (UgUN * IUL1 * cos(Phi UI)1 / (Un * In) * 121739436)
P2	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Realpower in Phase 2
P3	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Realpower in Phase 3
Reactive power				
Q1	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Blindpower in Phase 1 (UgUN * IUL1 * sin(Phi UI)1 / (Un * In) * 121739436)
Q2	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Blindpower in Phase 2
Q3	`%ID*`	DINT	$-2^{31}\ldots2^{31}-1$	Normalized Blindpower in Phase 3

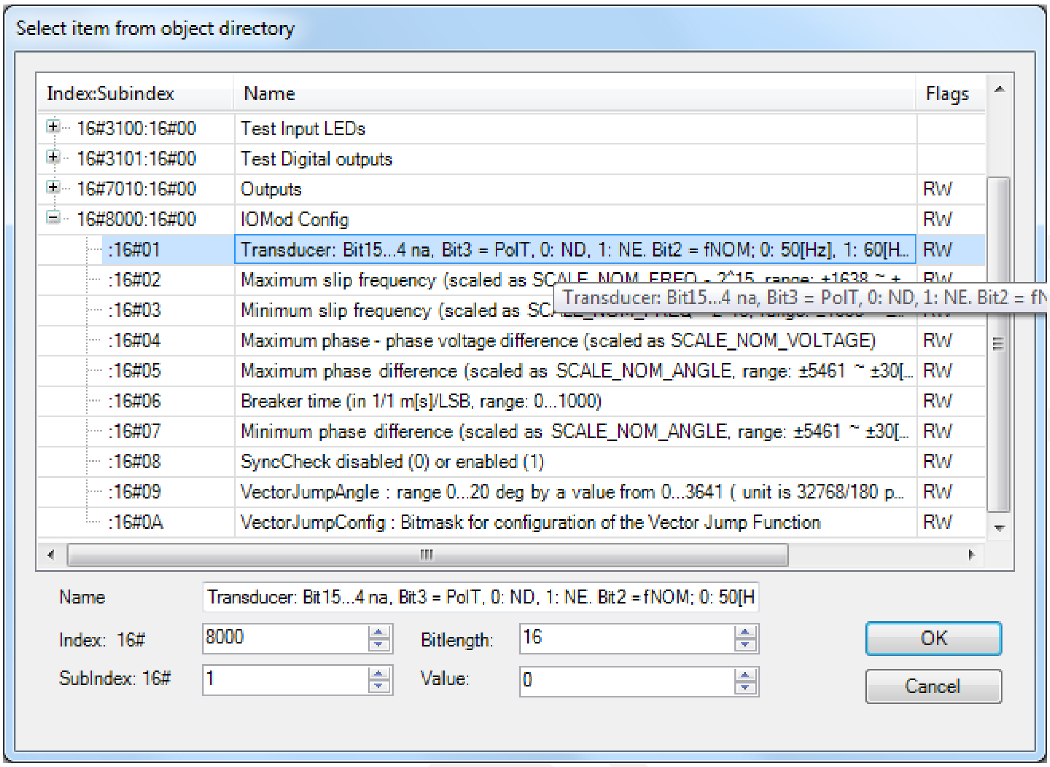
Channel	IEC Address	Type	Value range	Description
Generator voltages Phase-Phase				46811 : Un (either 240 Vrms or 690 Vrms is present)
UgUV	`%IW*`	UINT	0...65535	Normalized Voltage between Generator U-V
UgVW	`%IW*`	UINT	0...65535	Normalized Voltage between Generator V-W
UgVW	`%IW*`	UINT	0...65535	Normalized Voltage between Generator W-U

Channel	IEC Address	Type	Value range	Description
Busbar Voltages (Star)				UL1,L2,L3 N : a value of 27027 means Un (either 240/sqr(3)Vrms or 690/sqr(3)Vrms is present)
UL1N	%IW*	UINT	0...65535	Normalized Voltage between BusBar L1-N
UL2N	%IW*	UINT	0...65535	Normalized Voltage between BusBar L2-N
UL3N	%IW*	UINT	0...65535	Normalized Voltage between BusBar L3-N
Busbar Voltages (Delta)				A value of 46811 means Un (either 240 Vrms or 690 Vrms is present
UL1L2	%IW*	UINT	0...65535	Normalized Voltage between BusBar L1 and L2
UL2L3	%IW*	UINT	0...65535	Normalized Voltage between BusBar L2 and L3
UL3L1	%IW*	UINT	0...65535	Normalized Voltage between BusBar L3 and L1

Channel	IEC Address	Type	Value range	Description
Synchronization				UdiffL1U,L2V,L3W : a value of 46811 means Un (either 240 Vrms or 690 Vrms is present) Value -32768 means, frequency too low for measurement 0 means frequency is equal nominal frequency (either 50 Hz or 60 Hz) +32767 indicates double frequency (either 100 Hz of 120 Hz) and means, frequency is too high to be measured correct
UdiffL1U	%IW*	UINT	0...65535	Normalized Voltage between BusBar L1 and Generator U
UdiffL2V	%IW*	UINT	0...65535	Normalized Voltage between BusBar L2 and Generator V
UdiffL3W	%IW*	UINT	0...65535	Normalized Voltage between BusBar L3 and Generator W
PhiGenUBBL1	%IW*	INT	-32768...32767	Normalized Phaseangle between BusBar L1 and Generator U based upon zerocross
PhiGenVBBL2	%IW*	INT	-32768...32767	Normalized Phaseangle between BusBar L2 and Generator V based upon zerocross
PhiGenWBBL3	%IW*	INT	-32768...32767	Normalized Phaseangle between BusBar L3 and Generator W based upon zerocross
FreqDiffL1	%IW*	INT		FreqDiffL1
FreqDiffL2	%IW*	INT		FreqDiffL2
FreqDiffL3	%IW*	INT		FreqDiffL3
PhiBBL1L2	%IW*	INT		PhiBBL1L2
PhiBBL2L3	%IW*	INT		PhiBBL2L3
PhiBBL3L1	%IW*	INT		PhiBBL3L1
BreakerFeedback	%IW*	UINT		BreakerFeedback
TripFeedback	%IW*	UINT		TripFeedback

2.5.3 GPM5·1 – Startup parameters

Transducers



Select item from object directory

Index:Subindex	Name	Flags
16#3100:16#00	Test Input LEDs	
16#3101:16#00	Test Digital outputs	
16#7010:16#00	Outputs	RW
16#8000:16#00	IOMod Config	RW
:16#01	Transducer: Bit15...4 na, Bit3 = PolT, 0: ND, 1: NE. Bit2 = fNOM; 0: 50[Hz], 1: 60[H...	RW
:16#02	Maximum slip frequency (scaled as SCALE_NOM_FREQ - 2^15, range: ±1638 ~ ±	RW
:16#03	Minimum slip frequency (scaled as SC	
:16#04	Maximum phase - phase voltage difference (scaled as SCALE_NOM_VOLTAGE)	RW
:16#05	Maximum phase difference (scaled as SCALE_NOM_ANGLE, range: ±5461 ~ ±30[...	RW
:16#06	Breaker time (in 1/1 m[s]/LSB, range: 0...1000)	RW
:16#07	Minimum phase difference (scaled as SCALE_NOM_ANGLE, range: ±5461 ~ ±30[...	RW
:16#08	SyncCheck disabled (0) or enabled (1)	RW
:16#09	VectorJumpAngle : range 0...20 deg by a value from 0...3641 (unit is 32768/180 p...	RW
:16#0A	VectorJumpConfig : Bitmask for configuration of the Vector Jump Function	RW

Transducer: Bit15...4 na, Bit3 = PolT, 0: ND, 1: NE. Bit2 = fN

Name: Transducer: Bit15...4 na, Bit3 = PolT, 0: ND, 1: NE. Bit2 = fNOM; 0: 50[H

Index: 16# 8000

Bitlength: 16

SubIndex: 16# 1

Value: 0

Figure 2.19: Transducers

Prameter	Type	Value range	Description
Transducer	UINT	0. . . 64	Bit 0: AC nominal Input Voltage: 0= Un<=690 V; 1= Un<=240 V Bit 1 : nominal Input Current: 0= In=5 Arms; 1=In = 1 Arms Bit 2 : nominal frequency: 0=50 Hz; 1=60 Hz N.A Bit 3: polarity of trip output: 0=ND; 1 = NE (Not implemented) N.A Bit 4: DC nominal Input Voltage: 0= Un=400 V; 1= Un=140 V N.A Bit 5: Enable Direct Relay Handling - Relay 1: 0= Not enable; 1= Enable N.A Bit 6: Enable Direct Relay Handling - Relay 2: 0= Not enable; 1= Enable Bit 7. . . Bit 15 : not used

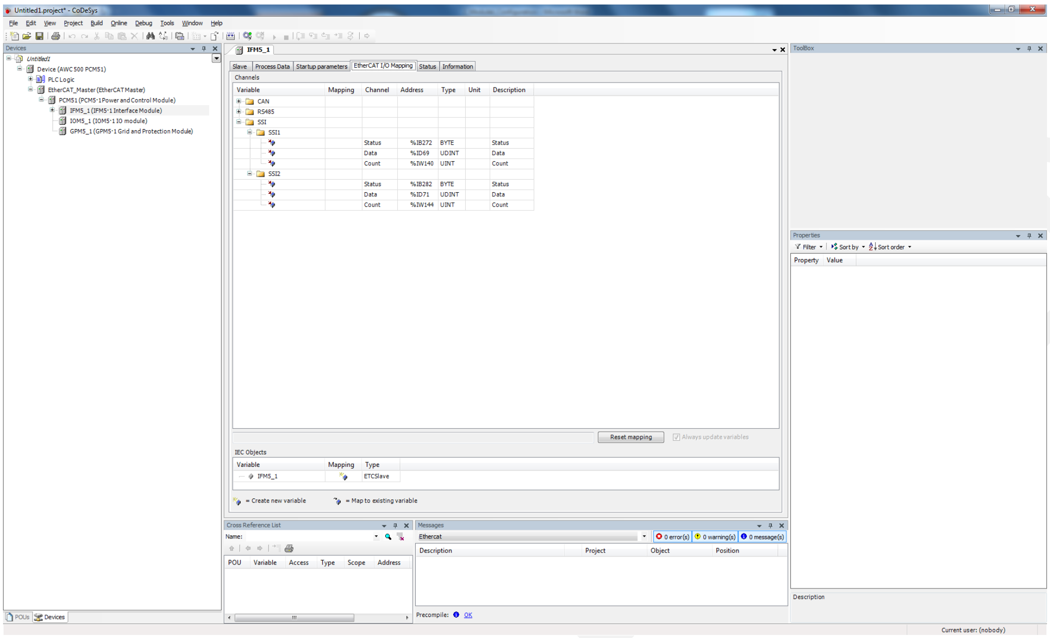
2.5.4 GPM5·1 – Conversion source code

```
PROGRAM PRG_CalculateGridValues
VAR
        rFactorVoltage :          LREAL;
        rFactorCurrent: LREAL;
        rFactorPower     :        LREAL;
        rTempValue       :        LREAL;

        rUL1N    : LREAL;         (* star voltage L1 N in [V] *)
        rUL2N    : LREAL;         (* star voltage L2 N in [V] *)
        rUL3N    : LREAL;         (* star voltage L3 N in [V] *)

        rIL1     : LREAL;         (* measurement grid Current L1 in [A] *)
        rIL2     : LREAL;         (* measurement grid Current L2 in [A] *)
        rIL3     : LREAL;        (* measurement grid Current L3 in [A] *)

        rPsum    : LREAL;        (* active Power in all 3 Phases in [W] *)
        rPL1     : LREAL;        (* active Power in L1 in [W] *)
        rPL2     : LREAL;        (* active Power in L2 in [W] *)
        rPL3     : LREAL;        (* active Power in L3 in [W] *)

        rQsum    : LREAL;        (* reactive Power in all 3 Phases in [VAR] *)
        rQL1     : LREAL;        (* reactive Power in L1 in [VAR] *)
        rQL2     : LREAL;        (* reactive Power in L2 in [VAR] *)
        rQL3     : LREAL;        (* reactive Power in L3 in [VAR] *)

        rSsum    :  LREAL;       (* apparent Power in all 3 Phases in [VAR] *)

        rFreq    : LREAL;        (* average Frequency of voltage on the 3 phases , range
                                                  0...100 Hz *)
        rCosPhi  : LREAL;

END_VAR
VAR CONSTANT
        rNOMINAL_FREQENCY :      LREAL := 50.000; (* nominal frequency = 50.000 mHz = 50 Hz*)
END_VAR

        (* volt per bit = 690 V per 46811bit bits = 14.740125184251564F *)
        rFactorVoltage := 690.0/46811.0;  (* Volt per bit raw reading from GPM *)

        (*current per bit = 2000 A per 15604 bits = 128.17226352217380158933F *)
        rFactorCurrent := 2000.0/15604.0;  (* A per bit raw reading from GPM *)

        (*power per bit = 2000 A * 400 V * 3 per 365218308 bits  = 0.0065714120771842576 96F*)
        rFactorPower := (SQRT(3.0)* 2000.0 * 400.0)/365218308.0;    (* W per bit raw reading
                                                                                                  realpo

        rUL1N:= rFactorVoltage * UINT_TO_LREAL(guiGPM_UgUN); (* star voltage L1 N in V *)
        rUL2N:= rFactorVoltage * UINT_TO_LREAL(guiGPM_UgVN);
        rUL3N:= rFactorVoltage * UINT_TO_LREAL(guiGPM_UgWN);

        (* Delta voltages not used
        := rFactorVoltage * UINT_TO_LREAL(guiGPM_UgUV); (* delta Voltage L1 L2 in V *)
        := rFactorVoltage * UINT_TO_LREAL(guiGPM_UgVW);
        := rFactorVoltage * UINT_TO_LREAL(guiGPM_UgWU);
        *)

        (* CURRENTS : all currents are calculated in unit mArms ( miliampere rms )
        CT ratio is 2000 / 1 and GPM delivers 15604 as DU16, when 2000 A are apparent, so the
        factor is 2000.000 A / 15604 bit *)
        rIL1 := rFactorCurrent * UINT_TO_LREAL(guiGPM_IUL1); (* measurement grid Current *)
        rIL2 := rFactorCurrent * UINT_TO_LREAL(guiGPM_IVL2);
        rIL3 := rFactorCurrent * UINT_TO_LREAL(guiGPM_IWL3);

        (* Power measurement : unit for all values is 1 W FOR realpower and 1 VAR for
       Reactivepower GPM delivers 365218308 when 690 V * 2000 A * SQRT(3) is apparent, so the
           factor is 2390230.11444 W / 365218308 bit *)
        (* active Power in all 3 Phases in W *)
        rPsum:= rFactorPower * DINT_TO_LREAL(gdiGPM_Psum) ;
        rPL1 := rFactorPower * DINT_TO_LREAL(gdiGPM_P1) ;       (* active Power in L1 in W *)
        rPL2 := rFactorPower * DINT_TO_LREAL(gdiGPM_P2) ;       (* active Power in L2 in W *)
        rPL3 := rFactorPower * DINT_TO_LREAL(gdiGPM_P3) ;       (* active Power in L3 in W *)

        (* reactive Power in all 3 Phases in VAR *)
        rQsum := rFactorPower * DINT_TO_LREAL(gdiGPM_Qsum);
        rQL1 := rFactorPower * DINT_TO_LREAL(gdiGPM_Q1) ; (* reactive Power in L1 in VAR *)
        rQL2 := rFactorPower * DINT_TO_LREAL(gdiGPM_Q2) ; (* reactive Power in L2 in VAR *)
        rQL3 := rFactorPower * DINT_TO_LREAL(gdiGPM_Q3) ; (* reactive Power in L3 in VAR *)

        (* Frequency measurement : Nominal Frequency 50,000 Hz *)
        rFreq := rNOMINAL_FREQENCY +
                ( rNOMINAL_FREQENCY * INT_TO_LREAL(giGPM_FreqDiff) *    1.0/32768.00 );


        (* Calculate CosPhi

                               P
             CosPhi = ──────────────────────
                    sprt( P^2 + Q^2 )

        *)
        rTempValue := rPsum*rPsum + rQsum*rQsum;

        IF ( rTempValue > 0.0 ) THEN
                rSsum := SQRT(rTempValue);
                        rCosPhi := rPsum/SQRT(rTempValue);
        ELSE
                rCosPhi := 0.0;
                rSsum := 0.0;
        END_IF
```

2.6 IFM5·1 – Interface and Fieldbus Module

2.6.1 IFM5·1 – EtherCAT I/O Mapping

SSI – EtherCAT I/O Mapping



IFM5_1

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
CAN						
RS485						
SSI						
SSI1						
		Status	%IB272	BYTE		Status
		Data	%ID69	UDINT		Data
		Count	%IW140	UINT		Count
SSI2						
		Status	%IB282	BYTE		Status
		Data	%ID71	UDINT		Data
		Count	%IW144	UINT		Count

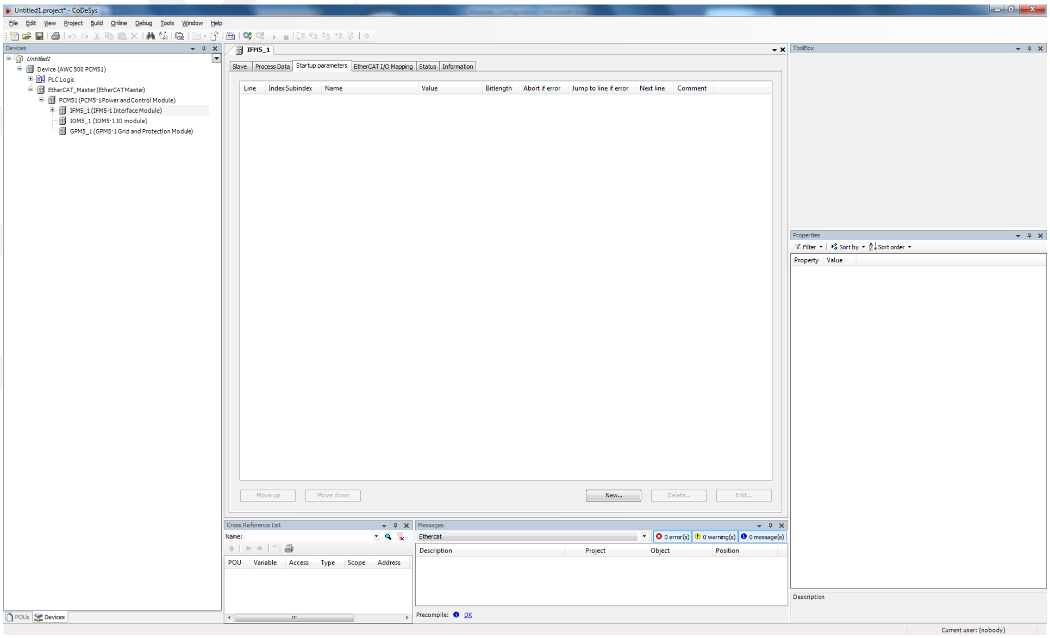
IEC Objects

Variable	Mapping	Type
IFM5_1		ETCSlave

Figure 2.20: SSI – EtherCAT I/O Mapping

Channel	IEC Address	Type	Description
Status	%IB*	BYTE	Status
Data	%ID*	UDINT	Data
Count	%IW*	UINT	Count

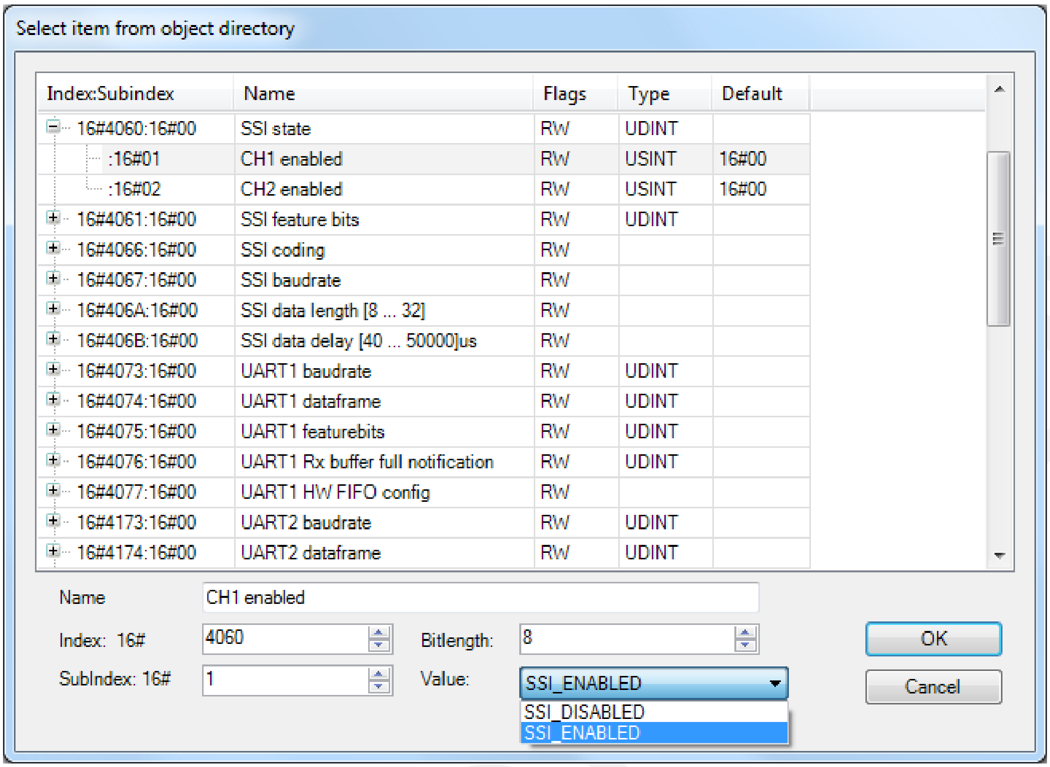
SSI – Startup parameters



Line	Index:Subindex	Name	Value	Bitlength	Abort if error	Jump to line if error	Next line	Comment

Figure 2.21: SSI – Startup parameters

SSI state



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#4060:16#00	SSI state	RW	UDINT	
:16#01	CH1 enabled	RW	USINT	16#00
:16#02	CH2 enabled	RW	USINT	16#00
16#4061:16#00	SSI feature bits	RW	UDINT	
16#4066:16#00	SSI coding	RW		
16#4067:16#00	SSI baudrate	RW		
16#406A:16#00	SSI data length [8 ... 32]	RW		
16#406B:16#00	SSI data delay [40 ... 50000]us	RW		
16#4073:16#00	UART1 baudrate	RW	UDINT	
16#4074:16#00	UART1 dataframe	RW	UDINT	
16#4075:16#00	UART1 featurebits	RW	UDINT	
16#4076:16#00	UART1 Rx buffer full notification	RW	UDINT	
16#4077:16#00	UART1 HW FIFO config	RW		
16#4173:16#00	UART2 baudrate	RW	UDINT	
16#4174:16#00	UART2 dataframe	RW	UDINT	

Name: CH1 enabled

Index: 16# 4060

SubIndex: 16# 1

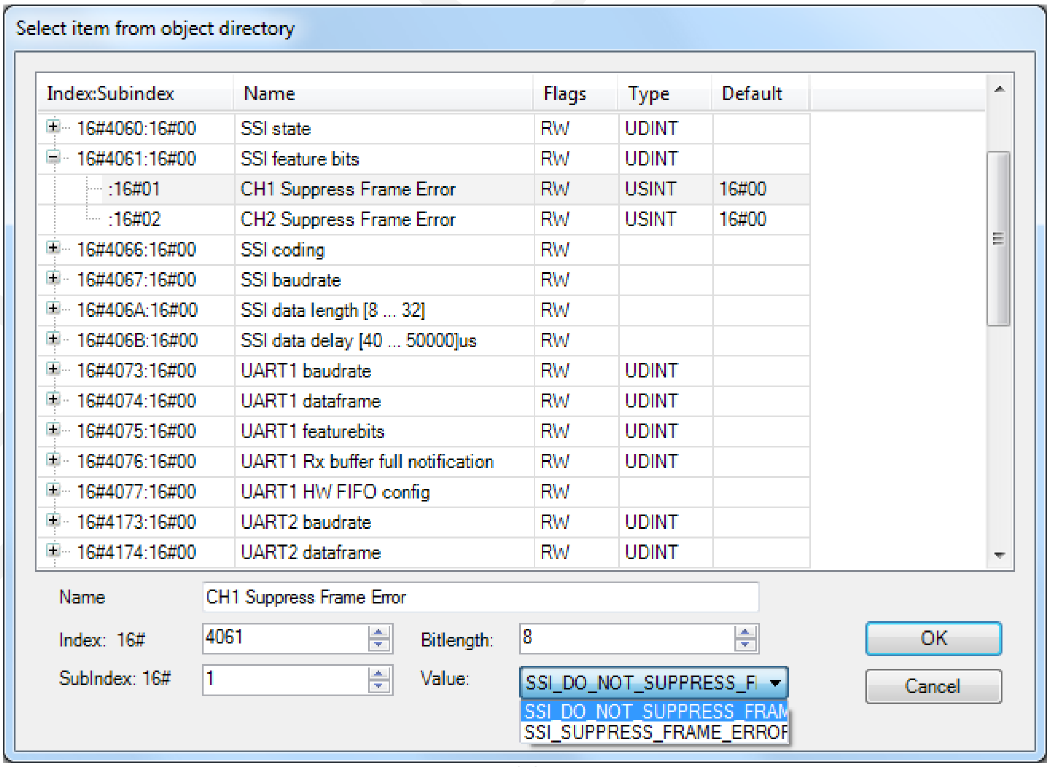
Bitlength: 8

Value: SSI_ENABLED

SSI_DISABLED
SSI_ENABLED

Figure 2.22: SSI state

SSI feature bits



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#4060:16#00	SSI state	RW	UDINT	
16#4061:16#00	SSI feature bits	RW	UDINT	
:16#01	CH1 Suppress Frame Error	RW	USINT	16#00
:16#02	CH2 Suppress Frame Error	RW	USINT	16#00
16#4066:16#00	SSI coding	RW		
16#4067:16#00	SSI baudrate	RW		
16#406A:16#00	SSI data length [8 ... 32]	RW		
16#406B:16#00	SSI data delay [40 ... 50000]us	RW		
16#4073:16#00	UART1 baudrate	RW	UDINT	
16#4074:16#00	UART1 dataframe	RW	UDINT	
16#4075:16#00	UART1 featurebits	RW	UDINT	
16#4076:16#00	UART1 Rx buffer full notification	RW	UDINT	
16#4077:16#00	UART1 HW FIFO config	RW		
16#4173:16#00	UART2 baudrate	RW	UDINT	
16#4174:16#00	UART2 dataframe	RW	UDINT	

Name: CH1 Suppress Frame Error

Index: 16# 4061

SubIndex: 16# 1

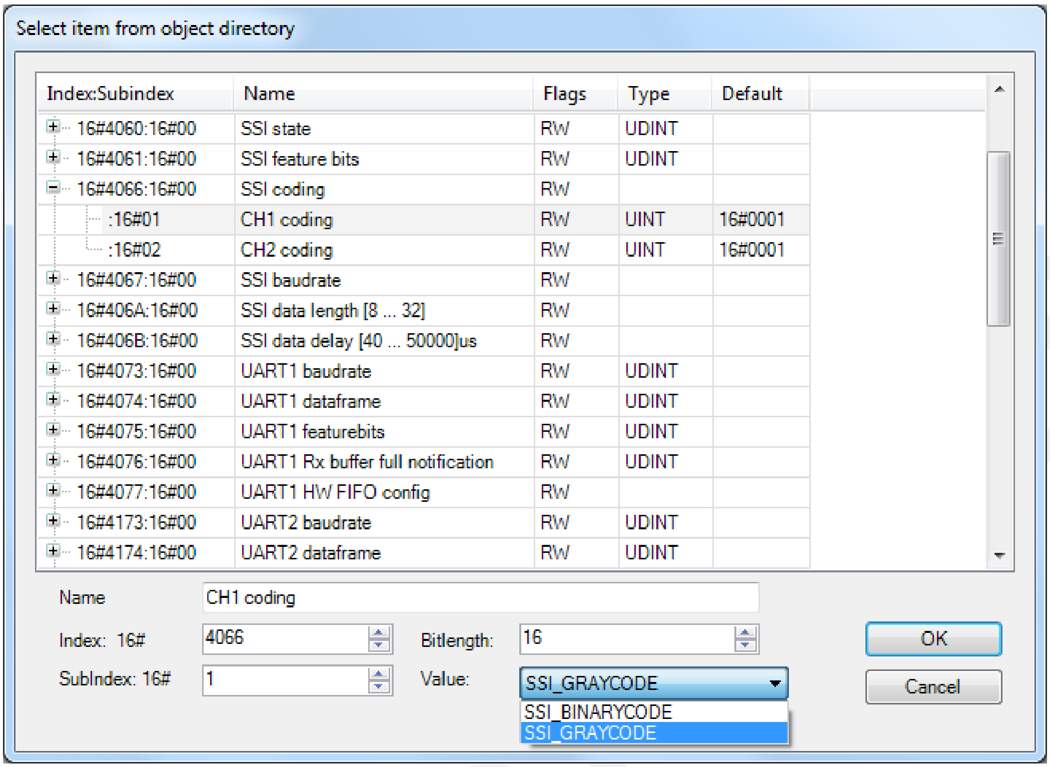
Bitlength: 8

Value: SSI_DO_NOT_SUPPRESS_F

SSI_DO_NOT_SUPPRESS_FRAM
SSI_SUPPRESS_FRAME_ERROF

Figure 2.23: SSI feature bits

SSI coding



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#4060:16#00	SSI state	RW	UDINT	
16#4061:16#00	SSI feature bits	RW	UDINT	
16#4066:16#00	SSI coding	RW		
:16#01	CH1 coding	RW	UINT	16#0001
:16#02	CH2 coding	RW	UINT	16#0001
16#4067:16#00	SSI baudrate	RW		
16#406A:16#00	SSI data length [8 ... 32]	RW		
16#406B:16#00	SSI data delay [40 ... 50000]us	RW		
16#4073:16#00	UART1 baudrate	RW	UDINT	
16#4074:16#00	UART1 dataframe	RW	UDINT	
16#4075:16#00	UART1 featurebits	RW	UDINT	
16#4076:16#00	UART1 Rx buffer full notification	RW	UDINT	
16#4077:16#00	UART1 HW FIFO config	RW		
16#4173:16#00	UART2 baudrate	RW	UDINT	
16#4174:16#00	UART2 dataframe	RW	UDINT	

Name: CH1 coding

Index: 16#: 4066

Bitlength: 16

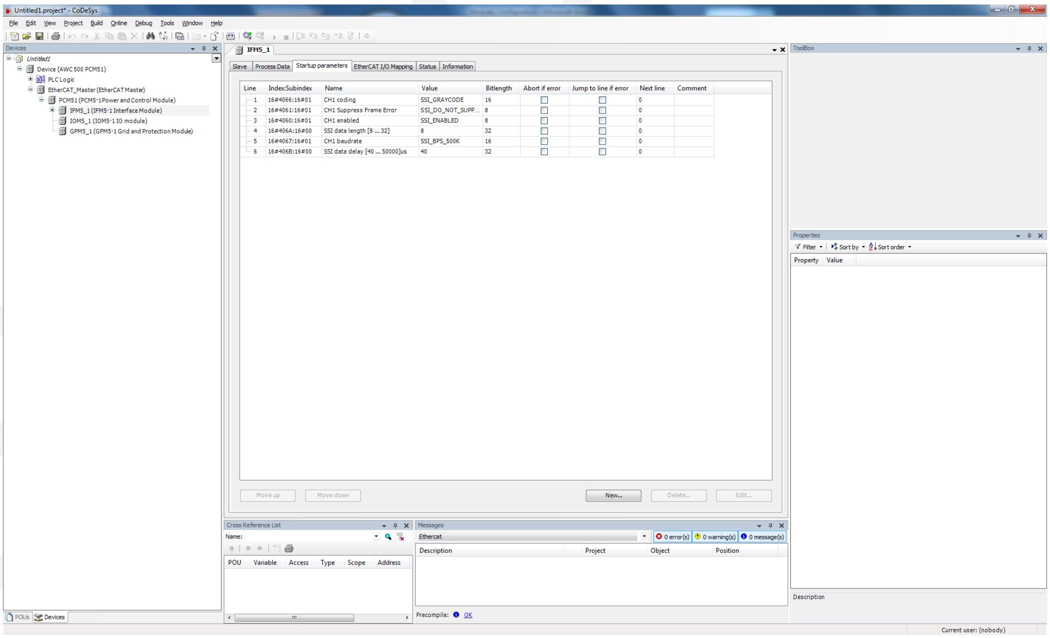
SubIndex: 16#: 1

Value: SSI_GRAYCODE

SSI_BINARYCODE
SSI_GRAYCODE

Figure 2.24: SSI coding

2.6.2 IFM5·1 – Startup parameters



Untitled1.project* - CoDeSys

Devices:
- Untitled1
  - Device (AWC 500 PCM51)
    - PLC Logic
    - EtherCAT_Master (EtherCAT Master)
      - PCM51 (PCM5·1 Power and Control Module)
        - IFM5_1 (IFM5·1 Interface Module)
        - IOM5_1 (IOM5·1 IO module)
        - GPM5_1 (GPM5·1 Grid and Protection Module)

IFM5_1

Line	Index:Subindex	Name	Value	Bitlength	Abort if error	Jump to line if error	Next line	Comment
1	16#4066:16#01	CH1 coding	SSI_GRAYCODE	16	□	□	0	
2	16#4061:16#01	CH1 Suppress Frame Error	SSI_DO_NOT_SUPP...	8	□	□	0	
3	16#4060:16#01	CH1 enabled	SSI_ENABLED	8	□	□	0	
4	16#406A:16#00	SSI data length [8 ... 32]	8	32	□	□	0	
5	16#4067:16#01	CH1 baudrate	SSI_BPS_500K	16	□	□	0	
6	16#406B:16#00	SSI data delay [40 ... 50000]us	40	32	□	□	0	

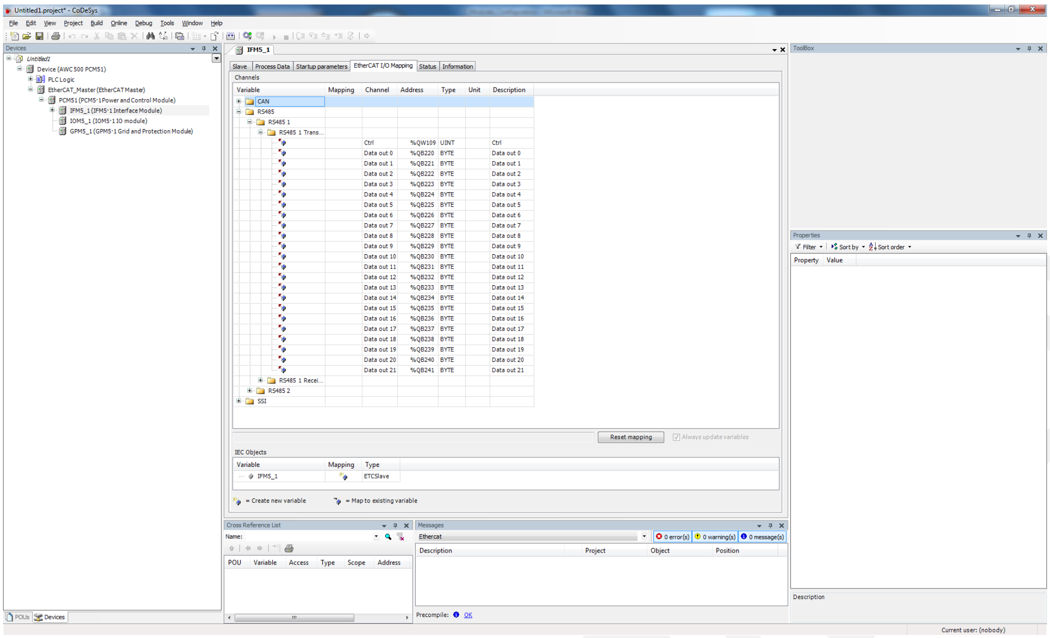
Precompile: OK

0 error(s) 0 warning(s) 0 message(s)

Current user: (nobody)

Figure 2.25: IFM5·1 – Startup parameters

RS485 UART – EtherCAT I/O Mapping



Variable	Mapping	Channel	Address	Type	Unit	Description
CAN						
RS485						
RS485 1						
RS485 1 Trans...						
		Ctrl	%QW109	UINT		Ctrl
		Data out 0	%QB220	BYTE		Data out 0
		Data out 1	%QB221	BYTE		Data out 1
		Data out 2	%QB222	BYTE		Data out 2
		Data out 3	%QB223	BYTE		Data out 3
		Data out 4	%QB224	BYTE		Data out 4
		Data out 5	%QB225	BYTE		Data out 5
		Data out 6	%QB226	BYTE		Data out 6
		Data out 7	%QB227	BYTE		Data out 7
		Data out 8	%QB228	BYTE		Data out 8
		Data out 9	%QB229	BYTE		Data out 9
		Data out 10	%QB230	BYTE		Data out 10
		Data out 11	%QB231	BYTE		Data out 11
		Data out 12	%QB232	BYTE		Data out 12
		Data out 13	%QB233	BYTE		Data out 13
		Data out 14	%QB234	BYTE		Data out 14
		Data out 15	%QB235	BYTE		Data out 15
		Data out 16	%QB236	BYTE		Data out 16
		Data out 17	%QB237	BYTE		Data out 17
		Data out 18	%QB238	BYTE		Data out 18
		Data out 19	%QB239	BYTE		Data out 19
		Data out 20	%QB240	BYTE		Data out 20
		Data out 21	%QB241	BYTE		Data out 21
RS485 1 Recei...						
RS485 2						
SSI						

Variable	Mapping	Type
IFM5_1		ETCSlave

POU	Variable	Access	Type	Scope	Address

Description	Project	Object	Position

Figure 2.26: RS485 UART – EtherCAT I/O Mapping

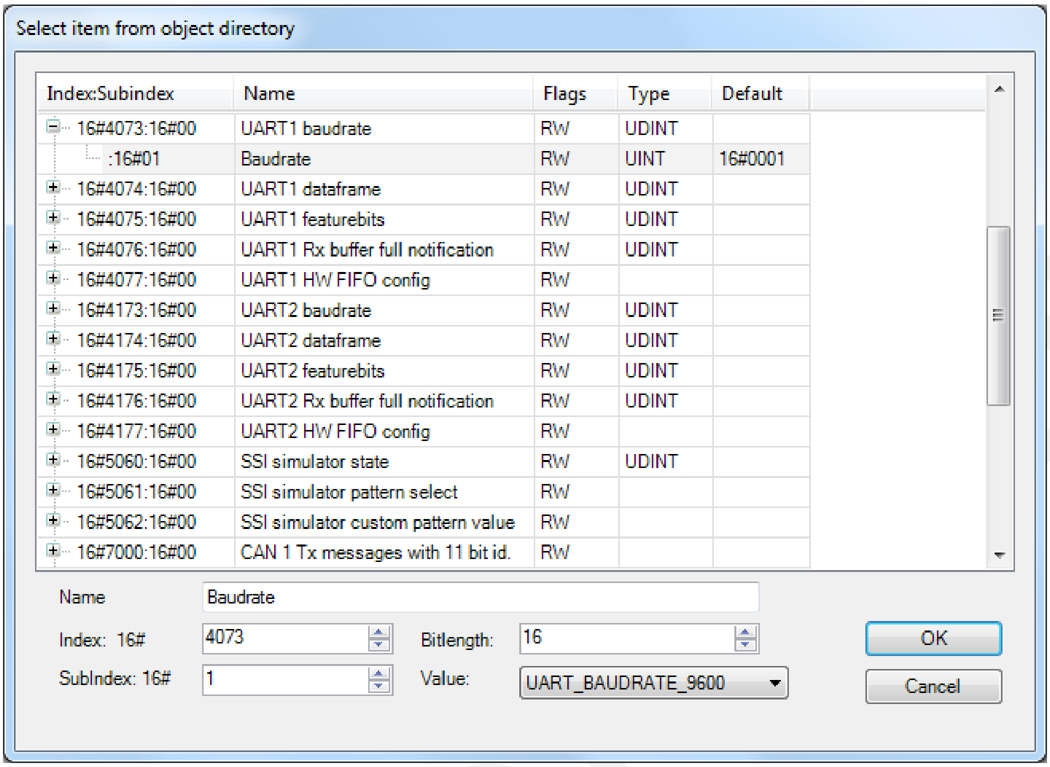
Channel	IEC Address	Type	Description
Ctrl	%QW*	UINT	Ctrl
Data out 0	%QB*	BYTE	Data out 0
...	...	...	...
Data out 21	%QB*	BYTE	Data out 21

Table 2.1: Transmit

Channel	IEC Address	Type	Description
Status	%IW*	UINT	Status
Data out 0	%IB*	BYTE	Data out 0
...	...	...	...
Data out 21	%IB*	BYTE	Data out 21

Table 2.2: Recieve

RS485 UART – Start-up parameters



Select item from object directory

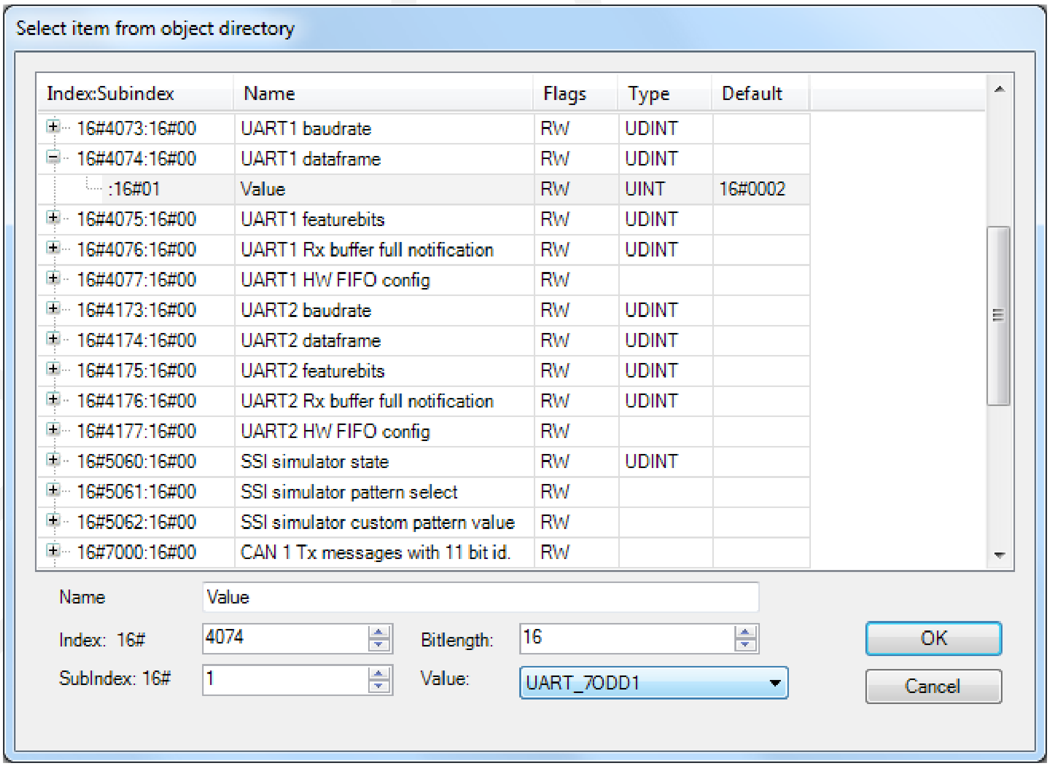
Index:Subindex	Name	Flags	Type	Default
16#4073:16#00	UART1 baudrate	RW	UDINT	
:16#01	Baudrate	RW	UINT	16#0001
16#4074:16#00	UART1 dataframe	RW	UDINT	
16#4075:16#00	UART1 featurebits	RW	UDINT	
16#4076:16#00	UART1 Rx buffer full notification	RW	UDINT	
16#4077:16#00	UART1 HW FIFO config	RW		
16#4173:16#00	UART2 baudrate	RW	UDINT	
16#4174:16#00	UART2 dataframe	RW	UDINT	
16#4175:16#00	UART2 featurebits	RW	UDINT	
16#4176:16#00	UART2 Rx buffer full notification	RW	UDINT	
16#4177:16#00	UART2 HW FIFO config	RW		
16#5060:16#00	SSI simulator state	RW	UDINT	
16#5061:16#00	SSI simulator pattern select	RW		
16#5062:16#00	SSI simulator custom pattern value	RW		
16#7000:16#00	CAN 1 Tx messages with 11 bit id.	RW		

Name: Baudrate

Index: 16# 4073 Bitlength: 16

SubIndex: 16# 1 Value: UART_BAUDRATE_9600

Figure 2.27: Baudrate



Select item from object directory

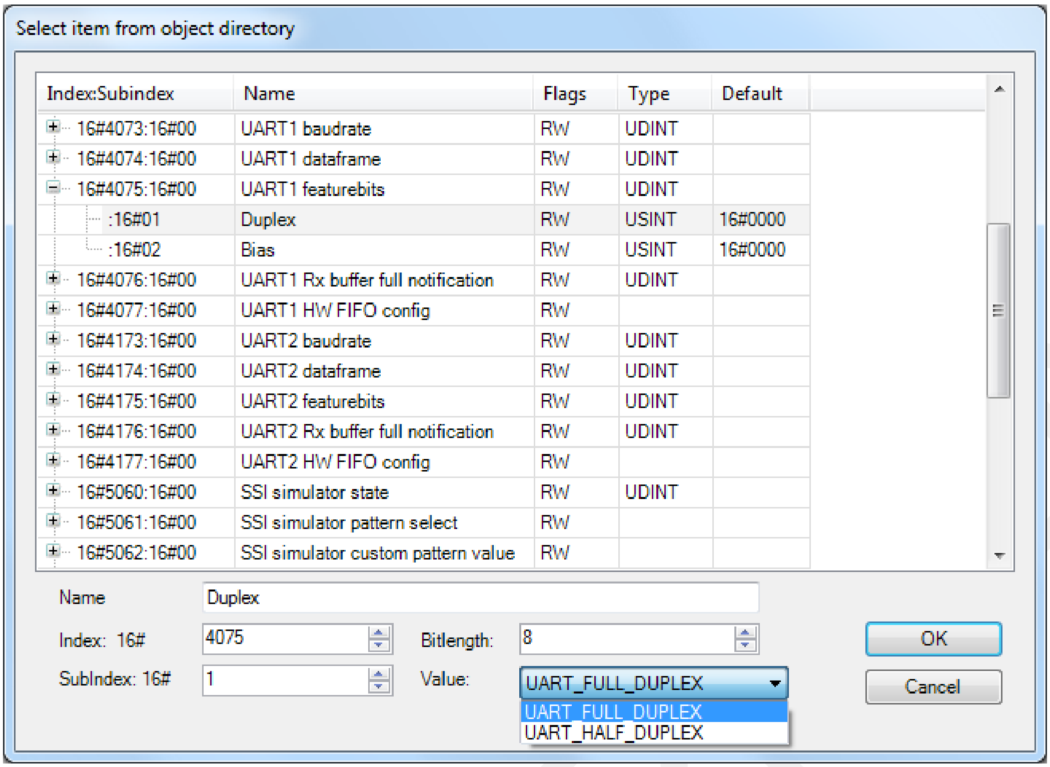
Index:Subindex	Name	Flags	Type	Default
16#4073:16#00	UART1 baudrate	RW	UDINT	
16#4074:16#00	UART1 dataframe	RW	UDINT	
:16#01	Value	RW	UINT	16#0002
16#4075:16#00	UART1 featurebits	RW	UDINT	
16#4076:16#00	UART1 Rx buffer full notification	RW	UDINT	
16#4077:16#00	UART1 HW FIFO config	RW		
16#4173:16#00	UART2 baudrate	RW	UDINT	
16#4174:16#00	UART2 dataframe	RW	UDINT	
16#4175:16#00	UART2 featurebits	RW	UDINT	
16#4176:16#00	UART2 Rx buffer full notification	RW	UDINT	
16#4177:16#00	UART2 HW FIFO config	RW		
16#5060:16#00	SSI simulator state	RW	UDINT	
16#5061:16#00	SSI simulator pattern select	RW		
16#5062:16#00	SSI simulator custom pattern value	RW		
16#7000:16#00	CAN 1 Tx messages with 11 bit id.	RW		

Name: Value

Index: 16# 4074 Bitlength: 16

SubIndex: 16# 1 Value: UART_7ODD1

Figure 2.28: Dataframe



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#4073:16#00	UART1 baudrate	RW	UDINT	
16#4074:16#00	UART1 dataframe	RW	UDINT	
16#4075:16#00	UART1 featurebits	RW	UDINT	
:16#01	Duplex	RW	USINT	16#0000
:16#02	Bias	RW	USINT	16#0000
16#4076:16#00	UART1 Rx buffer full notification	RW	UDINT	
16#4077:16#00	UART1 HW FIFO config	RW		
16#4173:16#00	UART2 baudrate	RW	UDINT	
16#4174:16#00	UART2 dataframe	RW	UDINT	
16#4175:16#00	UART2 featurebits	RW	UDINT	
16#4176:16#00	UART2 Rx buffer full notification	RW	UDINT	
16#4177:16#00	UART2 HW FIFO config	RW		
16#5060:16#00	SSI simulator state	RW	UDINT	
16#5061:16#00	SSI simulator pattern select	RW		
16#5062:16#00	SSI simulator custom pattern value	RW		

Name: Duplex

Index: 16# 4075

Bitlength: 8

SubIndex: 16# 1

Value: UART_FULL_DUPLEX

Figure 2.29: Duplex

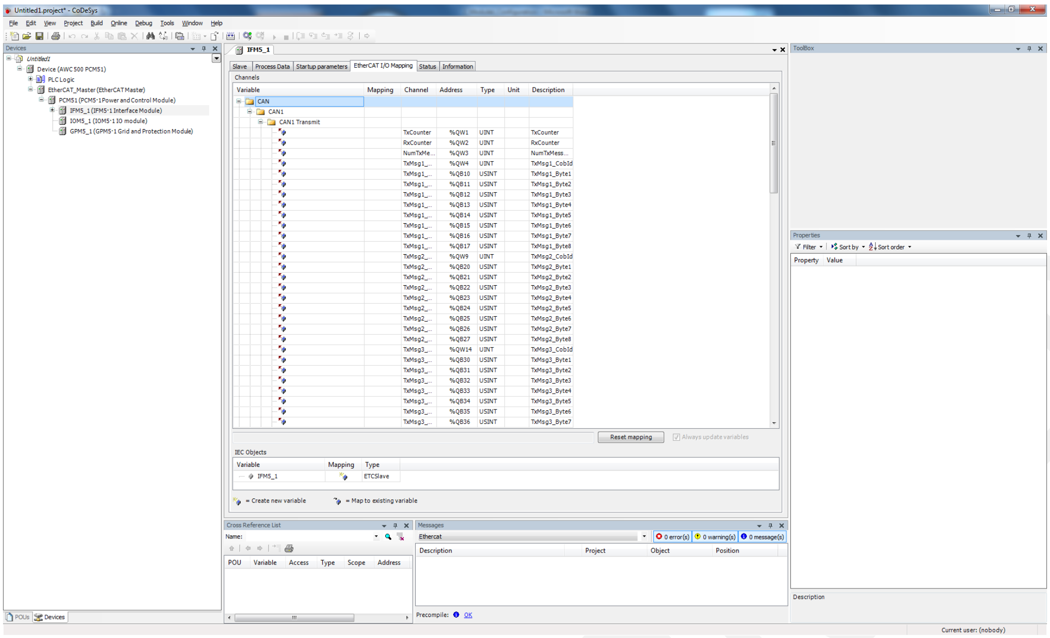
Rx Trigger level: For the RX SW circular buffer of the IFM.
A warning is given when the trigger level is reached on the RX in the UART. If the level is set to 0 the warning is disabled, if the trigger level is set to 2000 a warning is raised if there are more than 2000 bytes in the RX buffer on IFM5·1 (ie EtherCAT takes bytes of too slow.) There is 2048 bytes reserved for RX.

HW FIFO mode: Setup the number of bytes for the UART FIFO.
Valid options are none, 16 or 64 bytes. If none is selected an interrupt of the IFM CPU will occure each time a byte is received. The default value is 16 bytes.

HW FIFO Trigger: Trigger is used in combination with mode to tell when the FIFO must tell IFM CPU that it has to empty the FIFO. E.g. Mode is set to 16 bytes and trigger to 8 bytes, then the FIFO will be emptied every time there are 8 bytes in it.

In general it is suggested that the trigger level is set to 0, the fifo to 16 and the trigger to 8 (default)
Any other case is for surveillance purposes, or if it is required to run at baudrates larger than 115200.

CAN – EtherCAT I/O Mapping



IFM5_1

Slave | Process Data | Startup parameters | EtherCAT I/O Mapping | Status | Information

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
CAN						
CAN1						
CAN1 Transmit						
		TxCounter	%QW1	UINT		TxCounter
		RxCounter	%QW2	UINT		RxCounter
		NumTxMe...	%QW3	UINT		NumTxMess...
		TxMsg1_...	%QW4	UINT		TxMsg1_CobId
		TxMsg1_...	%QB10	USINT		TxMsg1_Byte1
		TxMsg1_...	%QB11	USINT		TxMsg1_Byte2
		TxMsg1_...	%QB12	USINT		TxMsg1_Byte3
		TxMsg1_...	%QB13	USINT		TxMsg1_Byte4
		TxMsg1_...	%QB14	USINT		TxMsg1_Byte5
		TxMsg1_...	%QB15	USINT		TxMsg1_Byte6
		TxMsg1_...	%QB16	USINT		TxMsg1_Byte7
		TxMsg1_...	%QB17	USINT		TxMsg1_Byte8
		TxMsg2_...	%QW9	UINT		TxMsg2_CobId
		TxMsg2_...	%QB20	USINT		TxMsg2_Byte1
		TxMsg2_...	%QB21	USINT		TxMsg2_Byte2
		TxMsg2_...	%QB22	USINT		TxMsg2_Byte3
		TxMsg2_...	%QB23	USINT		TxMsg2_Byte4
		TxMsg2_...	%QB24	USINT		TxMsg2_Byte5
		TxMsg2_...	%QB25	USINT		TxMsg2_Byte6
		TxMsg2_...	%QB26	USINT		TxMsg2_Byte7
		TxMsg2_...	%QB27	USINT		TxMsg2_Byte8
		TxMsg3_...	%QW14	UINT		TxMsg3_CobId
		TxMsg3_...	%QB30	USINT		TxMsg3_Byte1
		TxMsg3_...	%QB31	USINT		TxMsg3_Byte2
		TxMsg3_...	%QB32	USINT		TxMsg3_Byte3
		TxMsg3_...	%QB33	USINT		TxMsg3_Byte4
		TxMsg3_...	%QB34	USINT		TxMsg3_Byte5
		TxMsg3_...	%QB35	USINT		TxMsg3_Byte6
		TxMsg3_...	%QB36	USINT		TxMsg3_Byte7

IEC Objects

Variable	Mapping	Type
IFM5_1		ETCSlave

= Create new variable = Map to existing variable

Untitled1.project* - CoDeSys
File Edit View Project Build Online Debug Tools Window Help
Devices
Untitled1
Device (AWC 500 PCMS1)
PLC Logic
EtherCAT_Master (EtherCAT Master)
PCMS1 (PCM5-1 Power and Control Module)
IFM5_1 (IFM5-1 Interface Module)
IOM5_1 (IOM5-1 IO module)
GPM5_1 (GPM5-1 Grid and Protection Module)
Reset mapping
Always update variables
Cross Reference List
Name:
POU Variable Access Type Scope Address
Messages
Ethercat
0 error(s) 0 warning(s) 0 message(s)
Description Project Object Position
Precompile: OK
ToolBox
Properties
Filter Sort by Sort order
Property Value
Description
POUs Devices
Current user: (nobody)

Figure 2.30: CAN – EtherCAT I/O Mapping

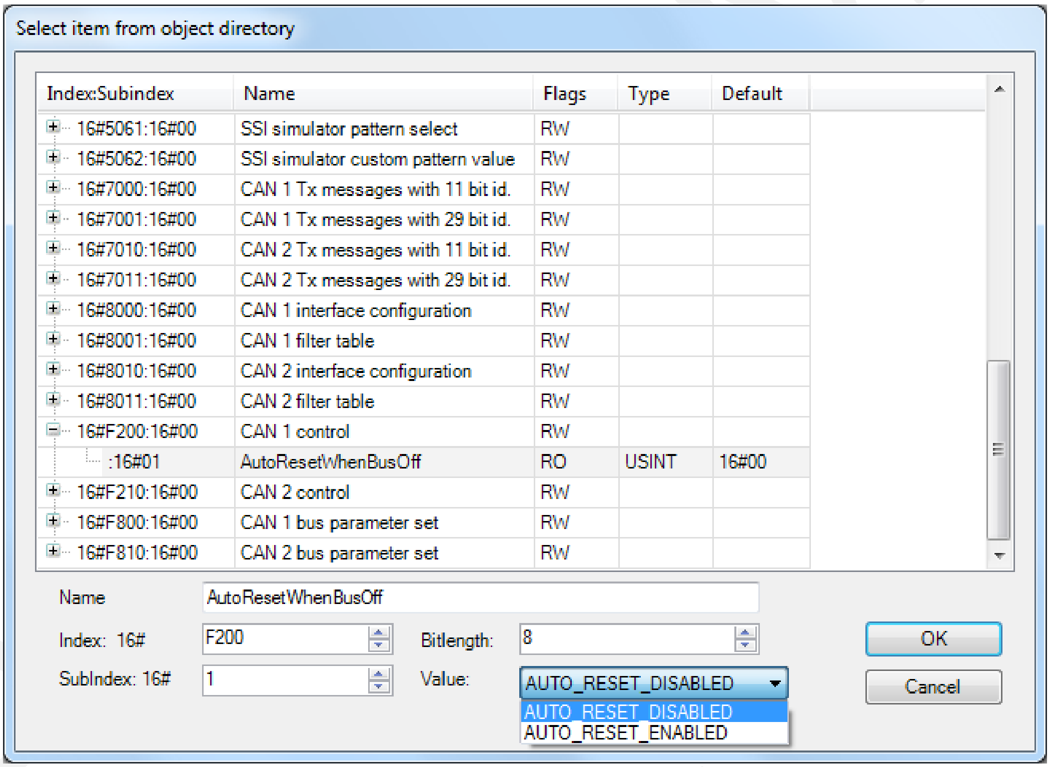
Channel	IEC Address	Type	Description
TxCounter	%QW*	UINT	TxCounter
RxCounter	%QB*	UINT	RxCounter
NumTxMessages	%QB*	UINT	NumTxMessages
TxMsg1_CobId	%QW*	UINT	TxMsg1_CobId
TxMsg1_Byte1	%QB*	BYTE	TxMsg1_Byte1
...	...	...	...
TxMsg1_Byte8	%QB*	BYTE	TxMsg1_Byte8
TxMsg2_CobId	%QW*	UINT	TxMsg2_CobId
...	...	...	...
...	...	...	...
TxMsg10_CobId	%QW*	UINT	TxMsg10_CobId
TxMsg10_Byte1	%QB*	BYTE	TxMsg10_Byte1
...	...	...	...
TxMsg10_Byte8	%QB*	BYTE	TxMsg10_Byte8
AutoResetWhenBusOff	%QX*	BUT	AutoResetWhenBusOff

Table 2.3: Transmit

Channel	IEC Address	Type	Description
TxCounter	%QW*	UINT	TxCounter
RxCounter	%QB*	UINT	RxCounter
NumRxMessages	%QB*	UINT	NumRxMessages
RxMsg1_CobId	%QW*	UINT	RxMsg1_CobId
RxMsg1_Byte1	%QB*	BYTE	RxMsg1_Byte1
...	...	...	...
RxMsg1_Byte8	%QB*	BYTE	RxMsg1_Byte8
RxMsg2_CobId	%QW*	UINT	RxMsg2_CobId
...	...	...	...
...	...	...	...
RxMsg10_CobId	%QW*	UINT	RxMsg10_CobId
RxMsg10_Byte1	%QB*	BYTE	RxMsg10_Byte1
...	...	...	...
RxMsg10_Byte8	%QB*	BYTE	RxMsg10_Byte8

Table 2.4: Recieve

CAN – Start-up parameters


Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#5061:16#00	SSI simulator pattern select	RW		
16#5062:16#00	SSI simulator custom pattern value	RW		
16#7000:16#00	CAN 1 Tx messages with 11 bit id.	RW		
16#7001:16#00	CAN 1 Tx messages with 29 bit id.	RW		
16#7010:16#00	CAN 2 Tx messages with 11 bit id.	RW		
16#7011:16#00	CAN 2 Tx messages with 29 bit id.	RW		
16#8000:16#00	CAN 1 interface configuration	RW		
16#8001:16#00	CAN 1 filter table	RW		
16#8010:16#00	CAN 2 interface configuration	RW		
16#8011:16#00	CAN 2 filter table	RW		
16#F200:16#00	CAN 1 control	RW		
:16#01	AutoResetWhenBusOff	RO	USINT	16#00
16#F210:16#00	CAN 2 control	RW		
16#F800:16#00	CAN 1 bus parameter set	RW		
16#F810:16#00	CAN 2 bus parameter set	RW		

Name: AutoResetWhenBusOff

Index: 16# F200

Bitlength: 8

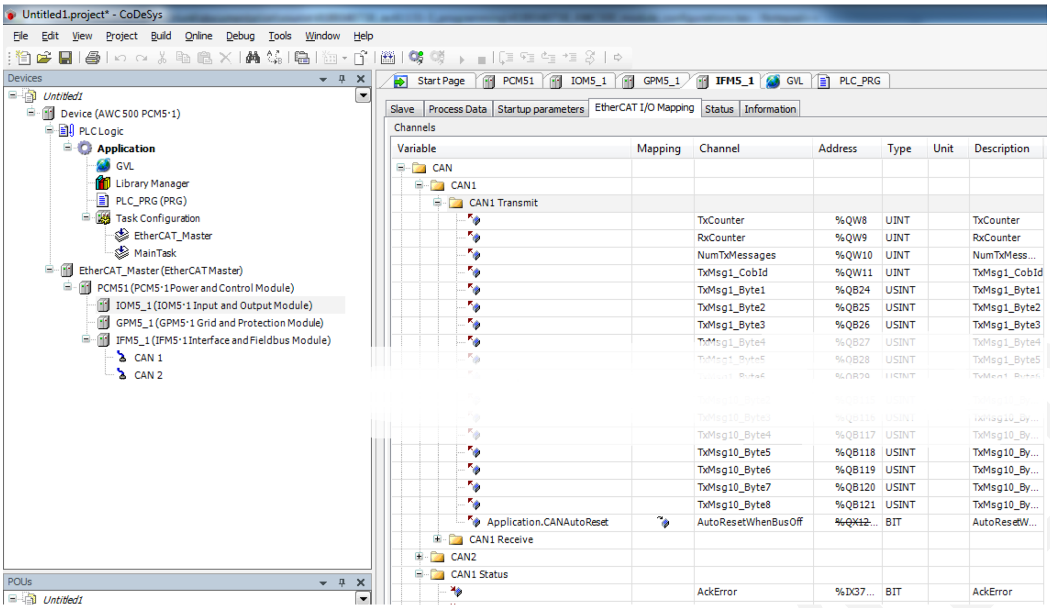
SubIndex: 16# 1

Value: AUTO_RESET_DISABLED

Figure 2.31: AutoResetWhenBusOff

When `AutoResetWhenBusOff` is set as an startup parameter the value is set only once during startup. As an alternative this value can be set each execusion cycle. This can be achieved by mapping `AutoResetWhenBusOff` to a global variable, assigned `TRUE`, in the EtherCAT I/O Mapping.

```
VAR_GLOBAL
        CANAutoReset  :  BOOL  :=  TRUE;
END_VAR
```

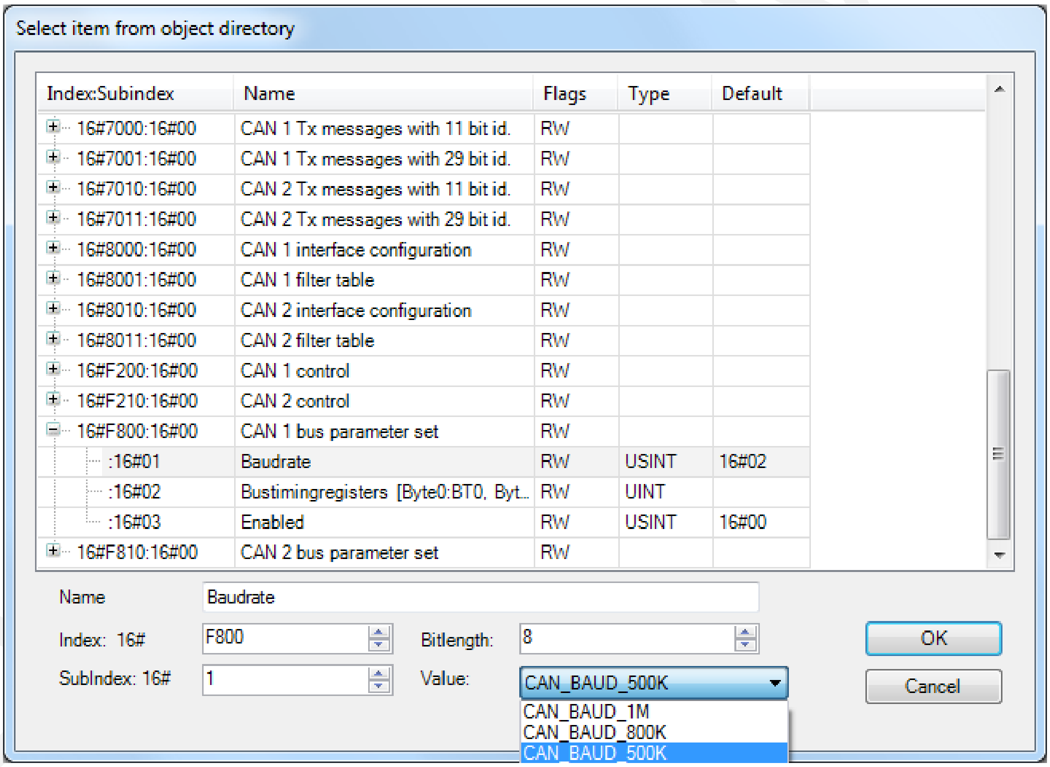


- Untitled1
  - Device (AWC 500 PCM5-1)
    - PLC Logic
      - Application
        - GVL
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - EtherCAT_Master
          - MainTask
    - EtherCAT_Master (EtherCAT Master)
      - PCM51 (PCM5-1 Power and Control Module)
        - IOM5_1 (IOM5-1 Input and Output Module)
        - GPM5_1 (GPM5-1 Grid and Protection Module)
        - IFM5_1 (IFM5-1 Interface and Fieldbus Module)
          - CAN 1
          - CAN 2

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
CAN						
CAN1						
CAN1 Transmit						
		TxCounter	%QW8	UINT		TxCounter
		RxCounter	%QW9	UINT		RxCounter
		NumTxMessages	%QW10	UINT		NumTxMess...
		TxMsg1_CobId	%QW11	UINT		TxMsg1_CobId
		TxMsg1_Byte1	%QB24	USINT		TxMsg1_Byte1
		TxMsg1_Byte2	%QB25	USINT		TxMsg1_Byte2
		TxMsg1_Byte3	%QB26	USINT		TxMsg1_Byte3
		TxMsg1_Byte4	%QB27	USINT		TxMsg1_Byte4
		TxMsg1_Byte5	%QB28	USINT		TxMsg1_Byte5
		TxMsg1_Byte6	%QB29	USINT		TxMsg1_Byte6
		TxMsg10_Byte2	%QB115	USINT		TxMsg10_By...
		TxMsg10_Byte3	%QB116	USINT		TxMsg10_By...
		TxMsg10_Byte4	%QB117	USINT		TxMsg10_By...
		TxMsg10_Byte5	%QB118	USINT		TxMsg10_By...
		TxMsg10_Byte6	%QB119	USINT		TxMsg10_By...
		TxMsg10_Byte7	%QB120	USINT		TxMsg10_By...
		TxMsg10_Byte8	%QB121	USINT		TxMsg10_By...
Application.CANAutoReset		AutoResetWhenBusOff	~~%QX12~~...	BIT		AutoResetW...
CAN1 Receive						
CAN2						
CAN1 Status						
		AckError	%IX37...	BIT		AckError

Figure 2.32: AutoResetWhenBusOff mapped to global variable



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#7000:16#00	CAN 1 Tx messages with 11 bit id.	RW		
16#7001:16#00	CAN 1 Tx messages with 29 bit id.	RW		
16#7010:16#00	CAN 2 Tx messages with 11 bit id.	RW		
16#7011:16#00	CAN 2 Tx messages with 29 bit id.	RW		
16#8000:16#00	CAN 1 interface configuration	RW		
16#8001:16#00	CAN 1 filter table	RW		
16#8010:16#00	CAN 2 interface configuration	RW		
16#8011:16#00	CAN 2 filter table	RW		
16#F200:16#00	CAN 1 control	RW		
16#F210:16#00	CAN 2 control	RW		
16#F800:16#00	CAN 1 bus parameter set	RW		
:16#01	Baudrate	RW	USINT	16#02
:16#02	Bustimingregisters [Byte0:BT0, Byt...	RW	UINT	
:16#03	Enabled	RW	USINT	16#00
16#F810:16#00	CAN 2 bus parameter set	RW		

Name: Baudrate

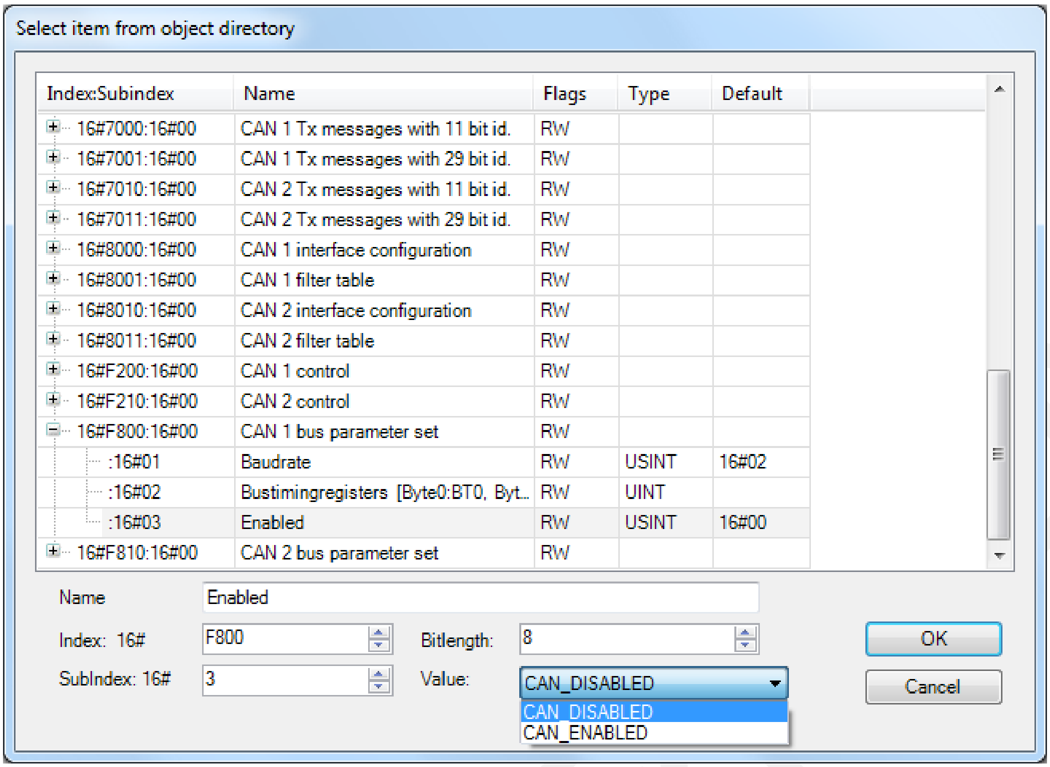
Index: 16# F800

Bitlength: 8

SubIndex: 16# 1

Value: CAN_BAUD_500K

Figure 2.33: Baudrate



Select item from object directory

Index:Subindex	Name	Flags	Type	Default
16#7000:16#00	CAN 1 Tx messages with 11 bit id.	RW		
16#7001:16#00	CAN 1 Tx messages with 29 bit id.	RW		
16#7010:16#00	CAN 2 Tx messages with 11 bit id.	RW		
16#7011:16#00	CAN 2 Tx messages with 29 bit id.	RW		
16#8000:16#00	CAN 1 interface configuration	RW		
16#8001:16#00	CAN 1 filter table	RW		
16#8010:16#00	CAN 2 interface configuration	RW		
16#8011:16#00	CAN 2 filter table	RW		
16#F200:16#00	CAN 1 control	RW		
16#F210:16#00	CAN 2 control	RW		
16#F800:16#00	CAN 1 bus parameter set	RW		
:16#01	Baudrate	RW	USINT	16#02
:16#02	Bustimingregisters [Byte0:BT0, Byt...	RW	UINT	
:16#03	Enabled	RW	USINT	16#00
16#F810:16#00	CAN 2 bus parameter set	RW		

Name: Enabled

Index: 16# F800

Bitlength: 8

SubIndex: 16# 3

Value: CAN_DISABLED

Figure 2.34: Enable/disable CAN bus

3 Communication

3.1 Creating a CANopen slave interface

If the AWC 500 is to exchange variables information with a CANopen Master e.g. on another AWC 500 system the CANopen slave interface can be used.

- Add CAN Local device to CAN channel of IFM5.1
- Specify variables for exchange
- Generate EDS file for CANopen Master
- Link I/O variables for exchange

3.1.1 Add CAN Local device to CAN channel of IFM5.1("Add→Device. . . ")

Start by adding a device on the IFM5.1:

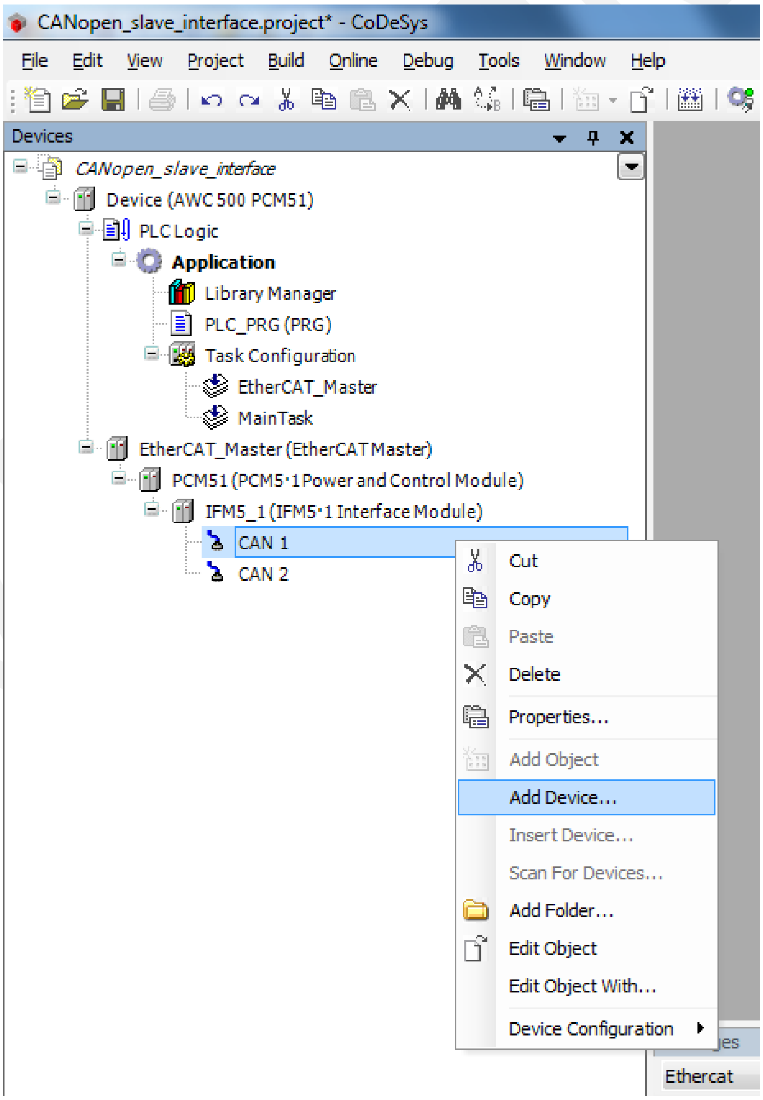
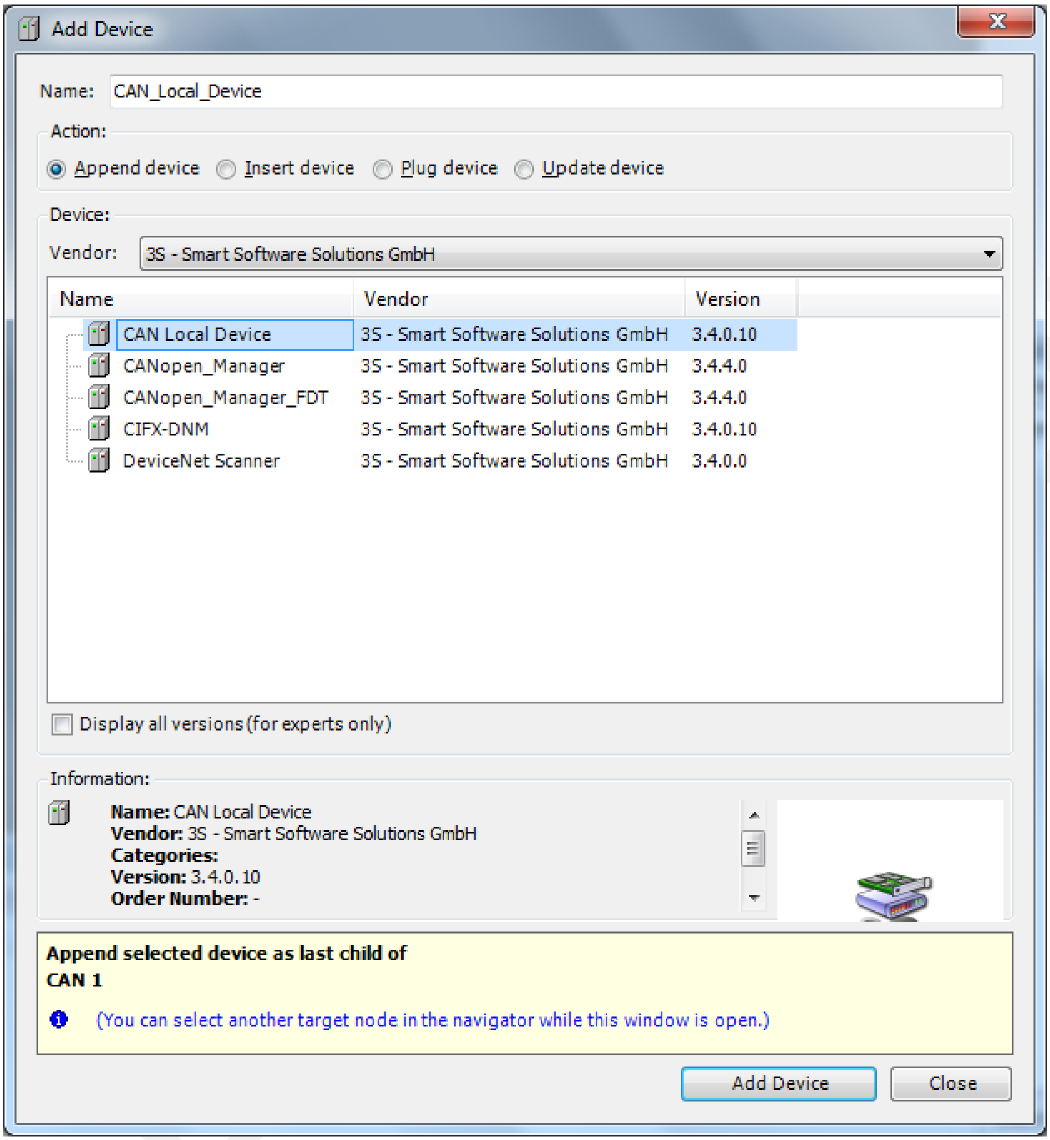


Figure 3.1: Add device

Change Vendor from DEIF to 3S - Smart Software solutions and select "CAN Local Device":



Name: CAN_Local_Device

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
CAN Local Device	3S - Smart Software Solutions GmbH	3.4.0.10
CANopen_Manager	3S - Smart Software Solutions GmbH	3.4.4.0
CANopen_Manager_FDT	3S - Smart Software Solutions GmbH	3.4.4.0
CIFX-DNM	3S - Smart Software Solutions GmbH	3.4.0.10
DeviceNet Scanner	3S - Smart Software Solutions GmbH	3.4.0.0

□ Display all versions (for experts only)

Information:

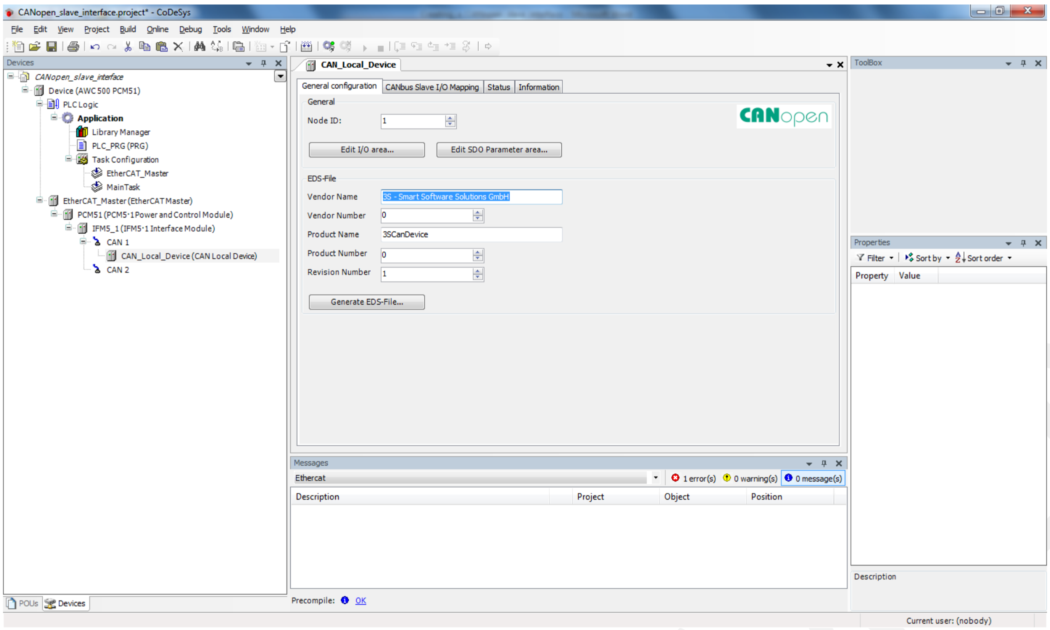
Name: CAN Local Device
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.0.10
Order Number: -

Append selected device as last child of
CAN 1

(You can select another target node in the navigator while this window is open.)

Figure 3.2: Select CAN Local Device

This creates a local CANopen device. This allows us to generate an EDS file for the CANopen master and specify IO variables for exchange:



Node ID: 1

Vendor Name: 3S - Smart Software Solutions GmbH

Vendor Number: 0

Product Name: 3SCanDevice

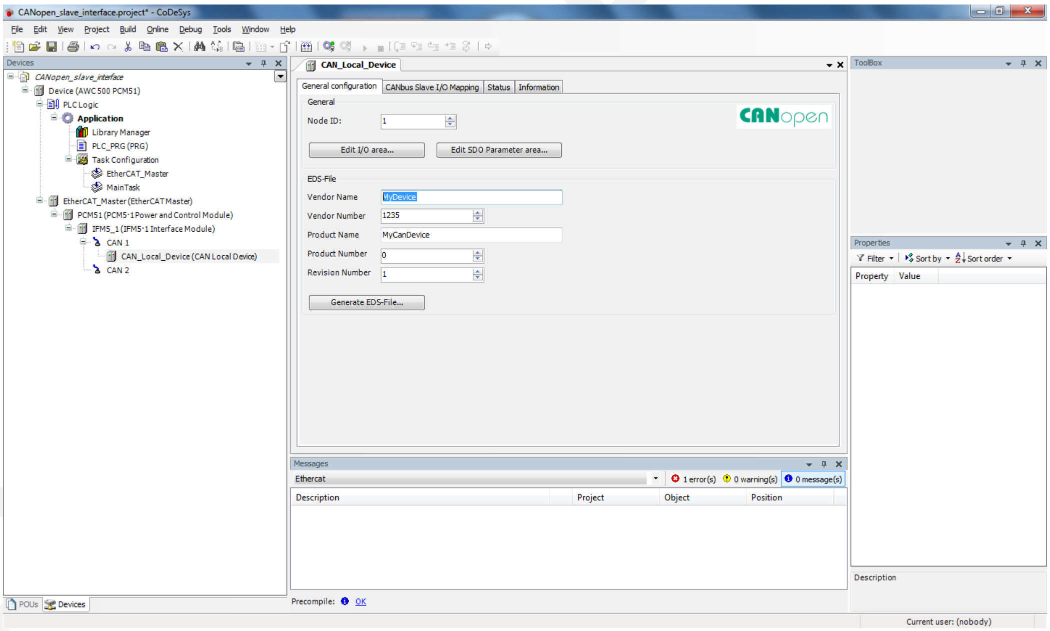
Product Number: 0

Revision Number: 1

Figure 3.3: CAN Local Device

3.1.2 Specify variables for exchange

Start by modifying the CAN EDS file information:



Node ID: 1

Vendor Name: MyDevice

Vendor Number: 1235

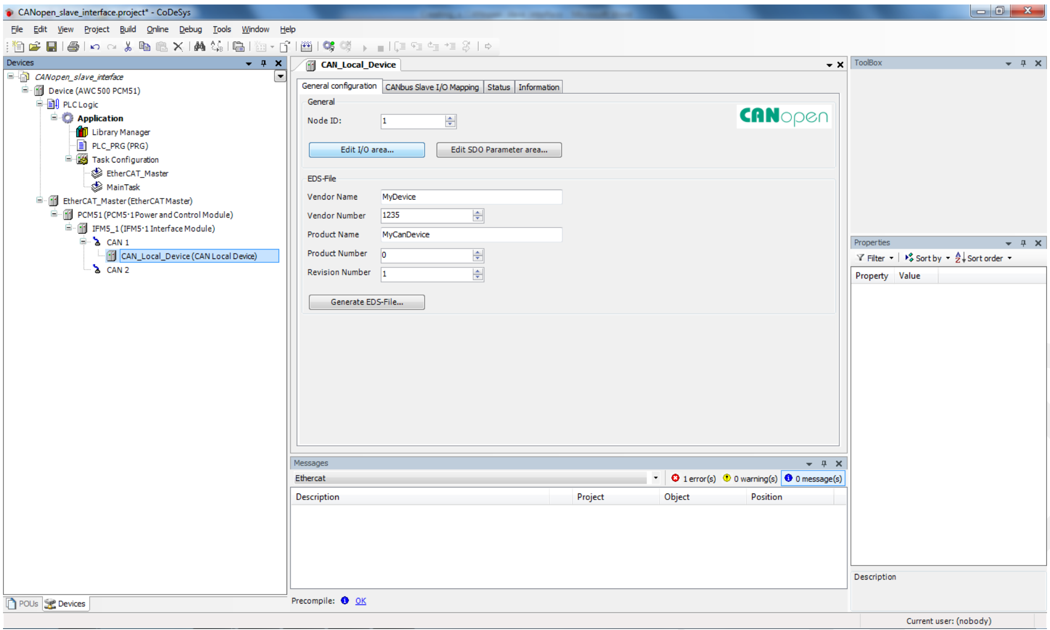
Product Name: MyCanDevice

Product Number: 0

Revision Number: 1

Figure 3.4: Modifying the CAN EDS file information

Then "Edit I/O area. . . " to add variables for exchange:



General

Node ID: 1

EDS-File

Vendor Name: MyDevice

Vendor Number: 1235

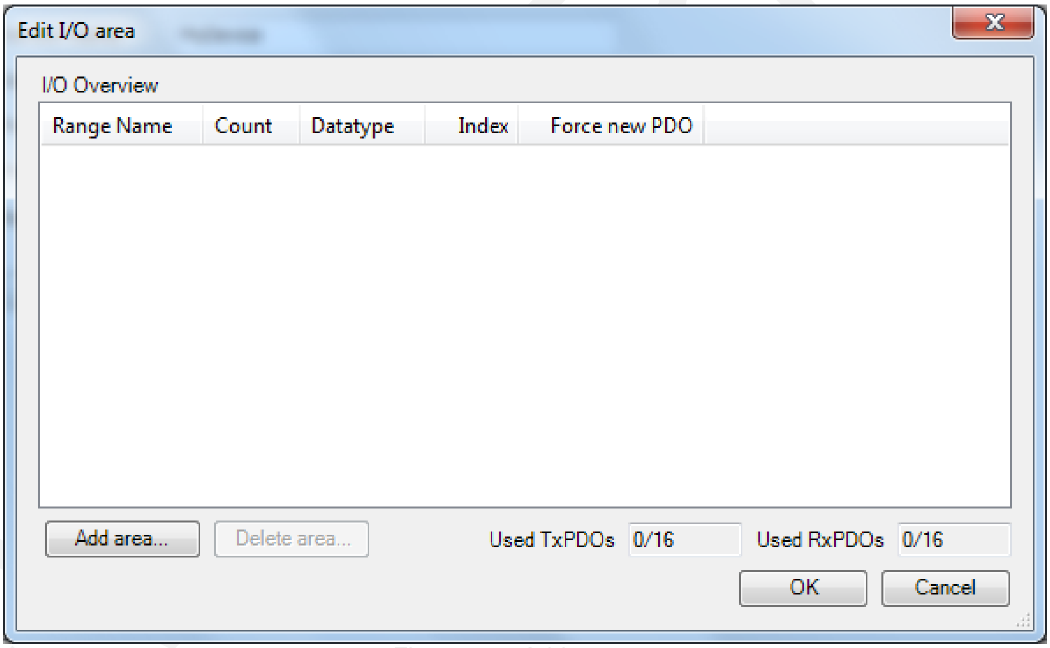
Product Name: MyCanDevice

Product Number: 0

Revision Number: 1

Figure 3.5: Edit I/O area

Here you can add variables with "Add area. . . ":

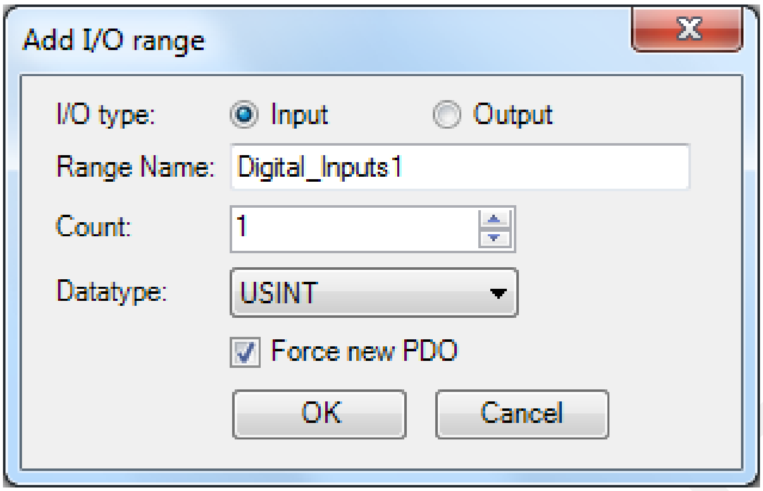


Range Name	Count	Datatype	Index	Force new PDO

Used TxPDOs 0/16

Used RxPDOs 0/16

Figure 3.6: Add area. . .



Add I/O range

I/O type: ● Input ○ Output

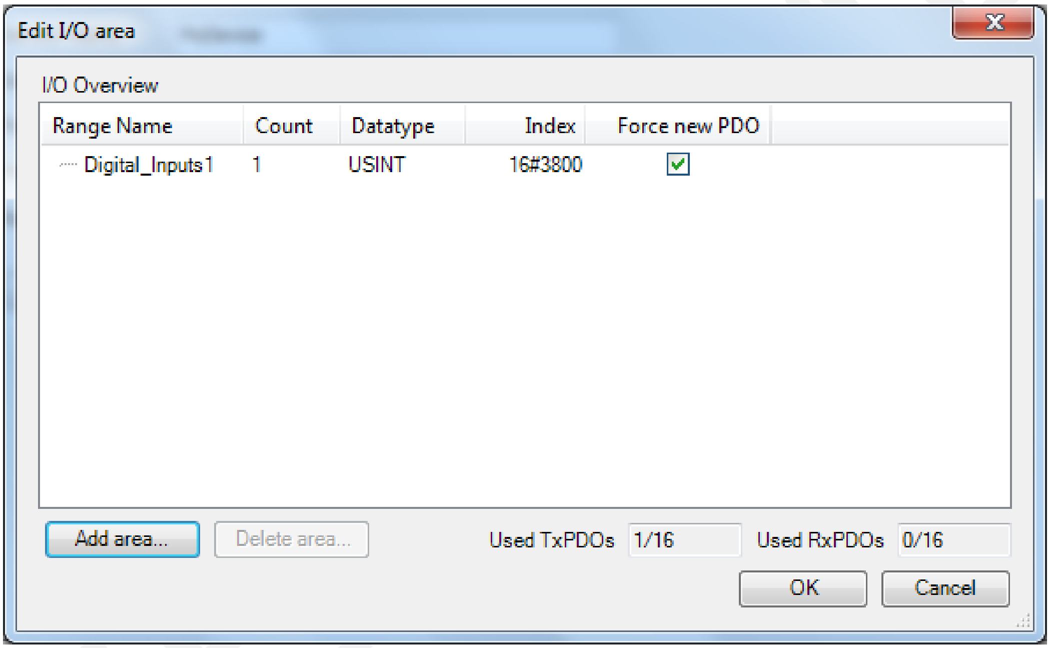
Range Name: Digital_Inputs1

Count: 1

Datatype: USINT

☑ Force new PDO

Figure 3.7: Add I/O range



Edit I/O area

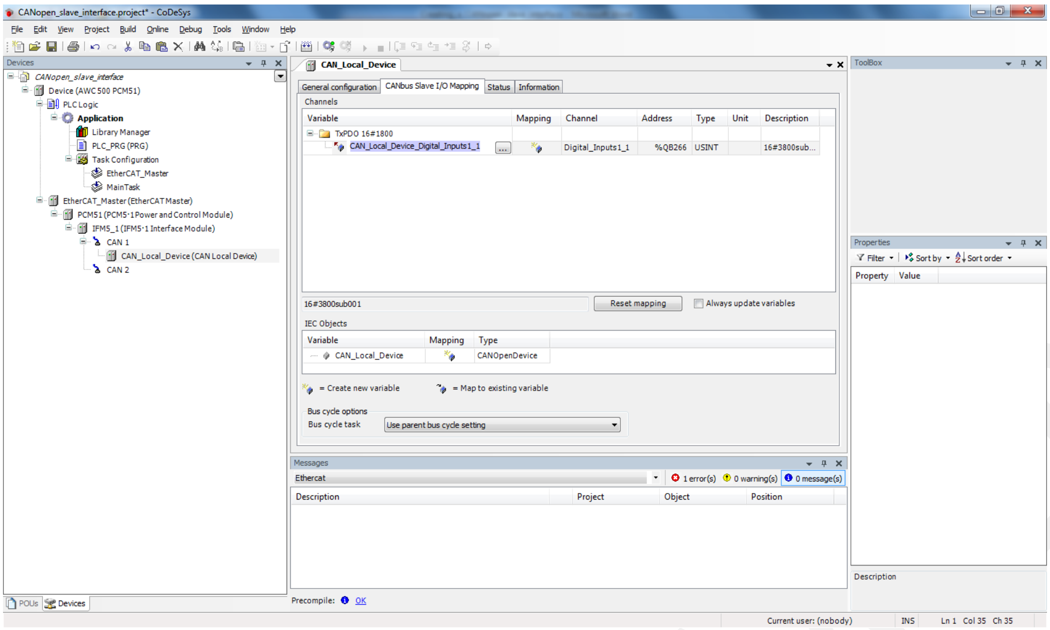
I/O Overview

Range Name	Count	Datatype	Index	Force new PDO
Digital_Inputs1	1	USINT	16#3800	☑

Used TxPDOs 1/16 Used RxPDOs 0/16

Figure 3.8: Edit I/O area

This creates the variables for exchange:



CANopen_slave_interface.project* - CoDeSys

Devices

- CANopen_slave_interface
  - Device (AWC 500 PCM51)
    - PLC Logic
      - Application
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - EtherCAT_Master
          - MainTask
    - EtherCAT_Master (EtherCAT Master)
      - PCM51 (PCM5·1 Power and Control Module)
        - IFM5_1 (IFM5·1 Interface Module)
          - CAN 1
            - CAN_Local_Device (CAN Local Device)
          - CAN 2

CAN_Local_Device — CANbus Slave I/O Mapping

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
TxPDO 16#1800						
CAN_Local_Device_Digital_Inputs1_1		Digital_Inputs1_1	%QB266	USINT		16#3800sub...

16#3800sub001

IEC Objects

Variable	Mapping	Type
CAN_Local_Device		CANOpenDevice

Bus cycle options
Bus cycle task: Use parent bus cycle setting

Messages: Ethercat — 1 error(s), 0 warning(s), 0 message(s)

Precompile: OK

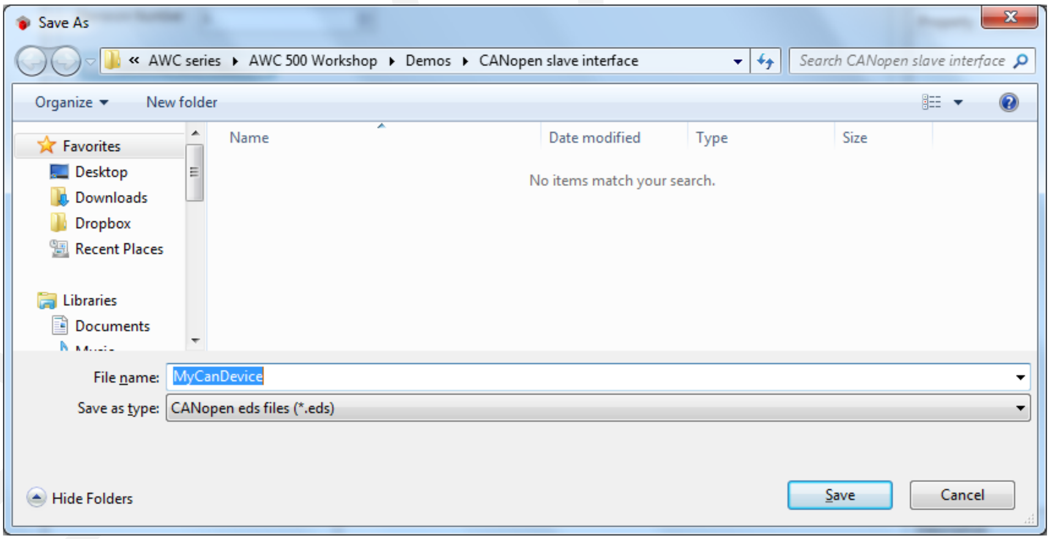
Current user: (nobody)

Figure 3.9: This creates the variables for exchange

The variable CAN_Local_Device_Digital_Inputs1_1 can be access from the PLC program or the mapping of the PDO can be changed to an existing variable in the PLC program.

3.1.3 Generate EDS file for CANopen Master

Finally generate an ESD file with the CANopen slave interface information using "General Information→Generate EDS-file. . . "



Save As

« AWC series ▸ AWC 500 Workshop ▸ Demos ▸ CANopen slave interface

No items match your search.

File name: MyCanDevice
Save as type: CANopen eds files (*.eds)

Figure 3.10: Generate EDS-file

3.1.4 Monitoring the state of the CANopen slave interface

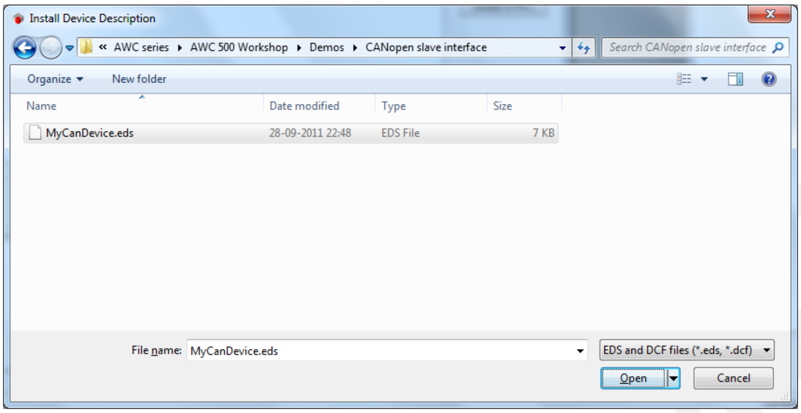
The IEC Object of the CANopen slave can be accessed to monitor the state:

```
VAR
    LocalState : CANOPENSTATE;
END_VAR

LocalState := CAN_Local_Device.GetState;
```

3.2 AWC 500 Inter system communication

Two AWC 500 systems can communicate together with the CANopen master/slave interface.

This same EDS file create above can be added to the CoDeSys Device repository (Tools→Device repository→Install. . . (EDS and DCF-files (*.eds, *.dcf))



Install Device Description
« AWC series ▸ AWC 500 Workshop ▸ Demos ▸ CANopen slave interface
Search CANopen slave interface
Organize ▾
New folder
Open
Cancel

Name	Date modified	Type	Size
MyCanDevice.eds	28-09-2011 22:48	EDS File	7 KB

File name: MyCanDevice.eds

EDS and DCF files (*.eds, *.dcf)

Figure 3.11: Install Device Description

Then in the CANopen master project add the device under the master:

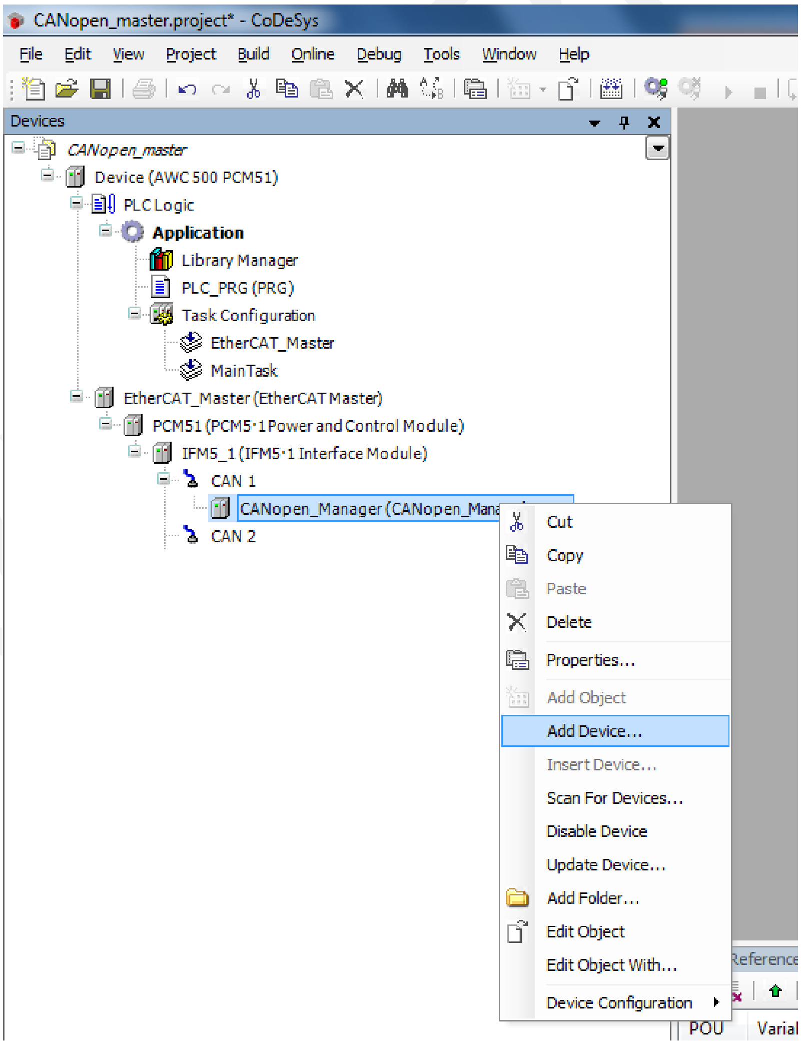
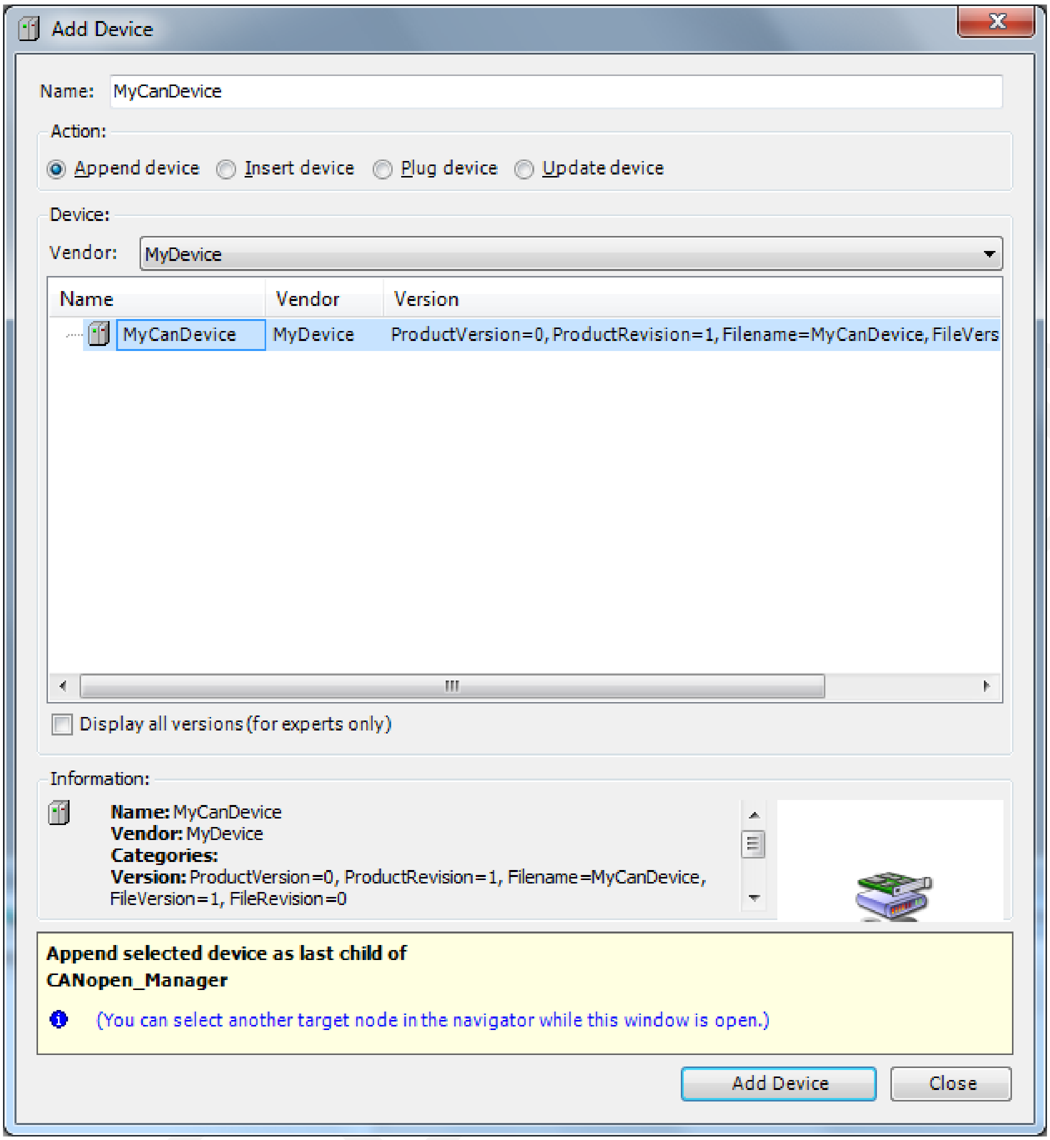


Figure 3.12: Adding device in CANopen master project

Locate the CANopen slave interface:



Add Device

Name: MyCanDevice

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: MyDevice

Name	Vendor	Version
MyCanDevice	MyDevice	ProductVersion=0, ProductRevision=1, Filename=MyCanDevice, FileVers

□ Display all versions (for experts only)

Information:

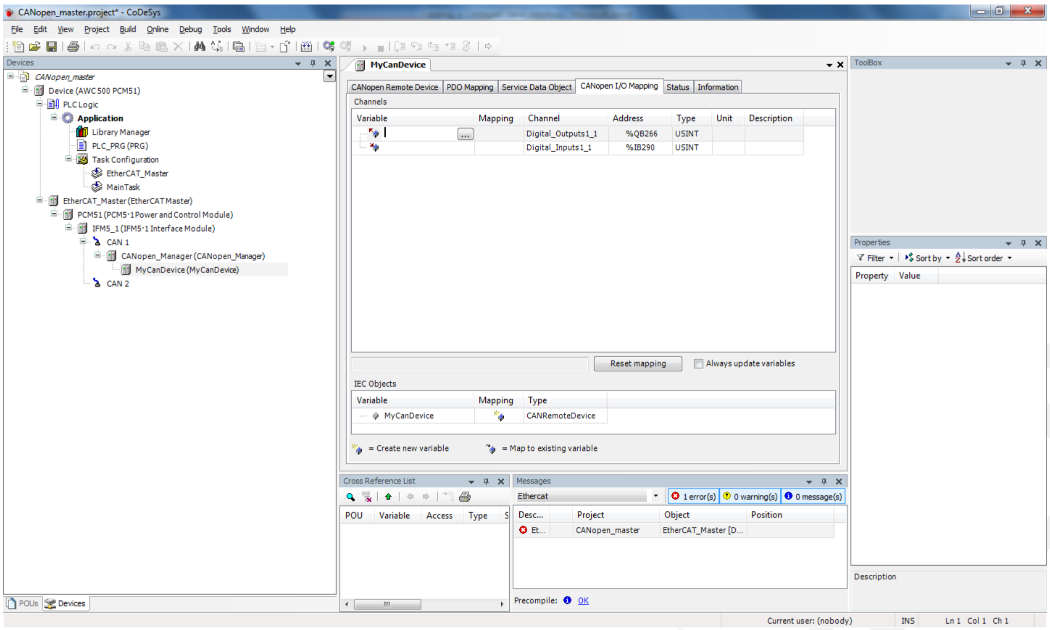
Name: MyCanDevice
Vendor: MyDevice
Categories:
Version: ProductVersion=0, ProductRevision=1, Filename=MyCanDevice, FileVersion=1, FileRevision=0

Append selected device as last child of CANopen_Manager

(You can select another target node in the navigator while this window is open.)

Figure 3.13: Locate the CANopen slave interface

Now the CANopen I/O variables are ready for linking:



Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
...		Digital_Outputs1_1	%QB266	USINT		
		Digital_Inputs1_1	%IB290	USINT		

☐ Always update variables

IEC Objects

Variable	Mapping	Type
MyCanDevice		CANRemoteDevice

= Create new variable = Map to existing variable

POU	Variable	Access	Type	S

Desc...	Project	Object	Position
Et...	CANopen_master	EtherCAT_Master [D...	

Figure 3.14: CANopen I/O variables are ready for linking

3.3 Demo

Optionally see the demo project `\CANopen Slave interface\CANopen_slave_interface.project`

3.4 Setting up a CANopen master

This section describes how to set up a CAN open master on the AWC 500 .
The steps are :

- Add CANopen manager to CAN channel of IFM5.1
- Scan for devices or add the devices manually via EDS files
- Link I/O variables for exchange

3.4.1 Add CANopen manager to CAN channel of IFM5.1("Add→Device. . . ")

The IFM5.1 has two separate CAN channels CAN 1 and CAN 2:

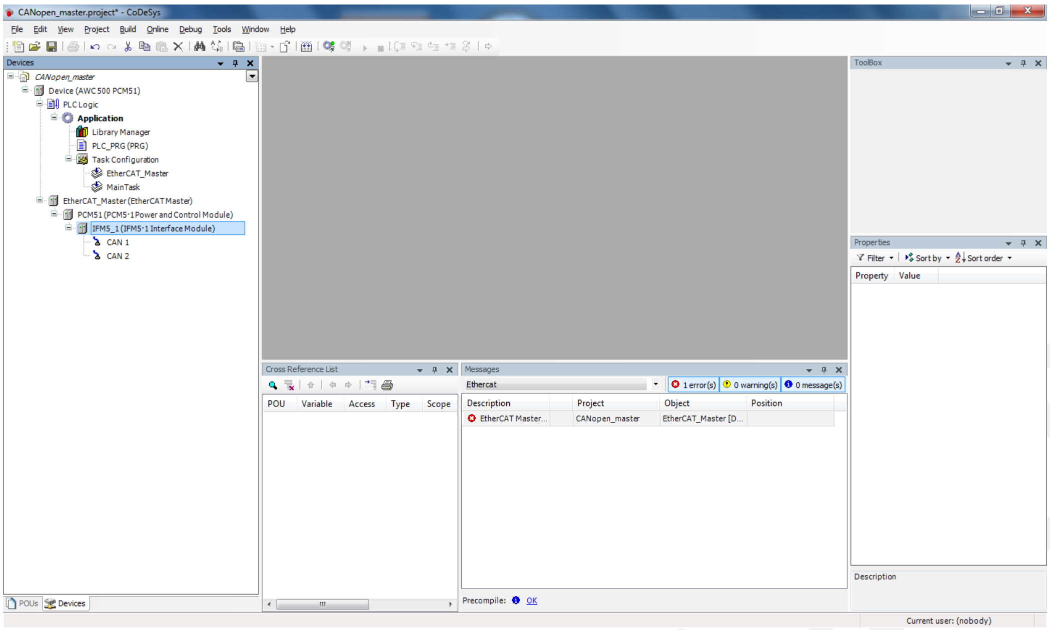


Figure 3.15: The two CAN channels of the IFM5.1

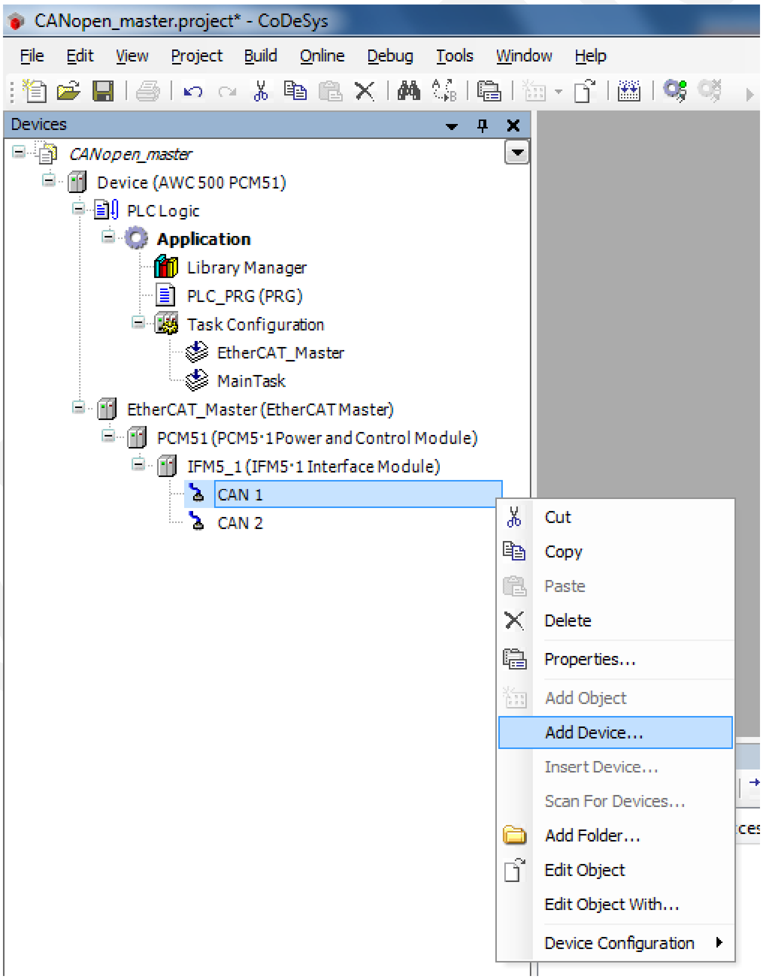
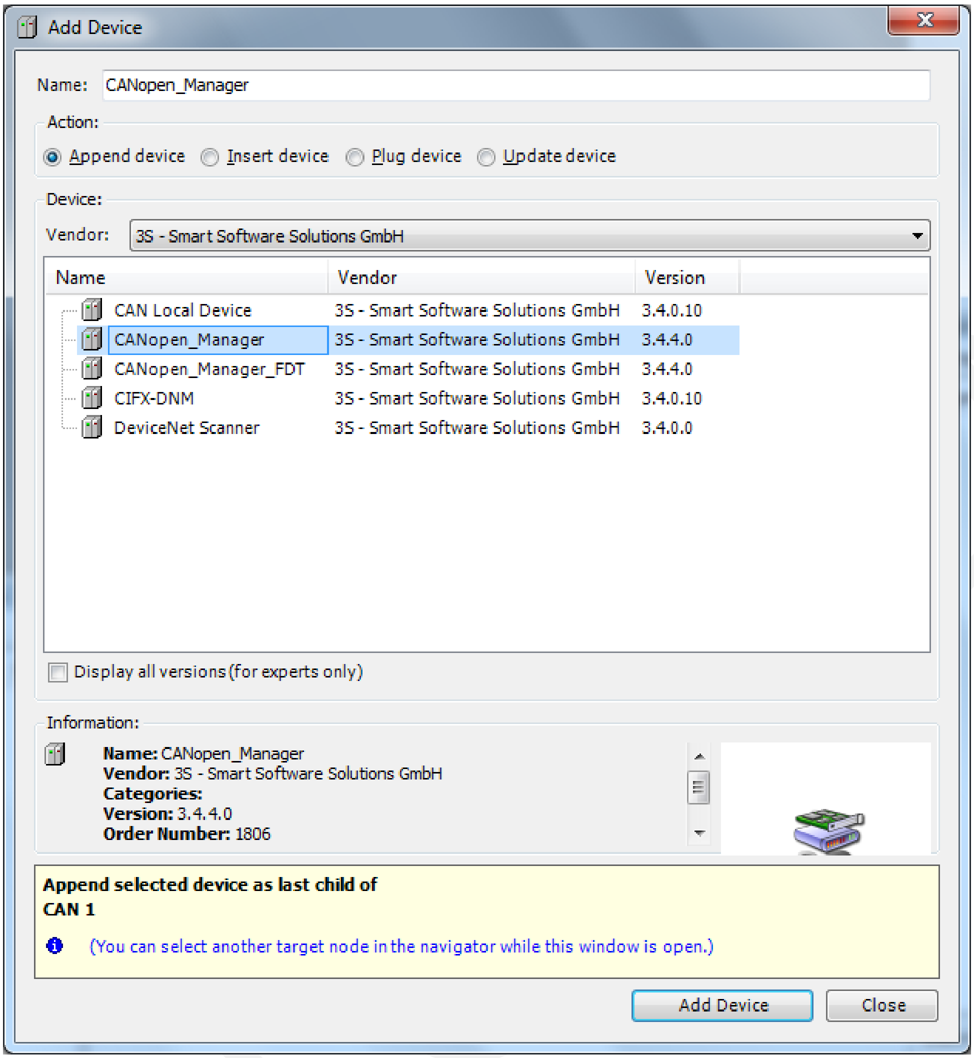


Figure 3.16: Add device

Change Vendor from DEIF to 3S - Smart Software solutions and select CANopen_manager



Add Device

Name: CANopen_Manager

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
CAN Local Device	3S - Smart Software Solutions GmbH	3.4.0.10
CANopen_Manager	3S - Smart Software Solutions GmbH	3.4.4.0
CANopen_Manager_FDT	3S - Smart Software Solutions GmbH	3.4.4.0
CIFX-DNM	3S - Smart Software Solutions GmbH	3.4.0.10
DeviceNet Scanner	3S - Smart Software Solutions GmbH	3.4.0.0

□ Display all versions (for experts only)

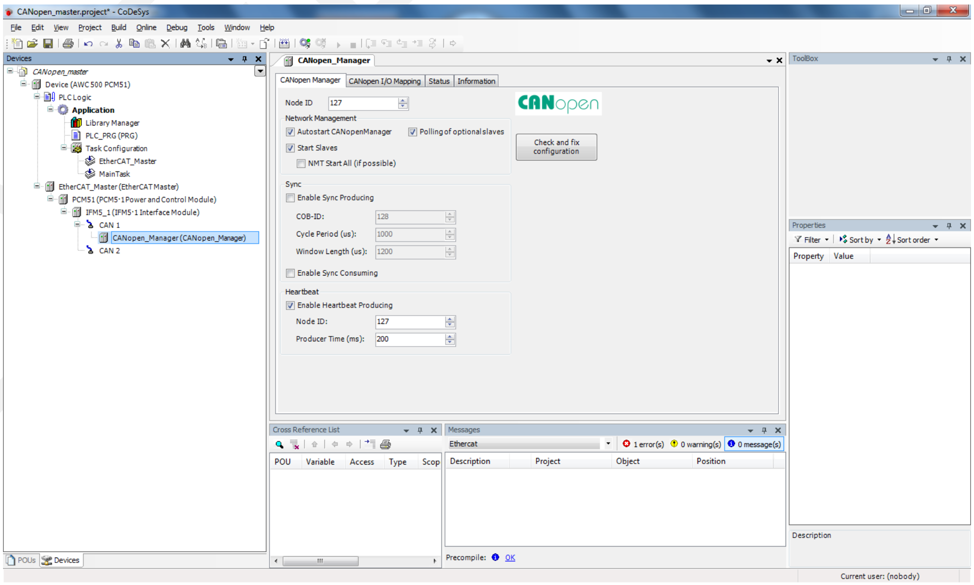
Information:

Name: CANopen_Manager
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.4.0
Order Number: 1806

Append selected device as last child of
CAN 1

(You can select another target node in the navigator while this window is open.)

Figure 3.17: Add CANopen_manager



CANopen_master.project* - CoDeSys

Devices

- CANopen_master
  - Device (AWC 500 PCM51)
    - PLC Logic
      - **Application**
        - Library Manager
        - PLC_PRG (PRG)
        - Task Configuration
          - EtherCAT_Master
          - MainTask
    - EtherCAT_Master (EtherCAT Master)
      - PCM51 (PCM5-1 Power and Control Module)
        - IFM5_1 (IFM5-1 Interface Module)
          - CAN 1
            - CANopen_Manager (CANopen_Manager)
          - CAN 2

CANopen_Manager

Node ID 127

Network Management

☑ Autostart CANopenManager ☑ Polling of optional slaves

☑ Start Slaves

□ NMT Start All (if possible)

Sync

□ Enable Sync Producing

COB-ID: 128

Cycle Period (us): 1000

Window Length (us): 1200

□ Enable Sync Consuming

Heartbeat

☑ Enable Heartbeat Producing

Node ID: 127

Producer Time (ms): 200

1 error(s) 0 warning(s) 0 message(s)

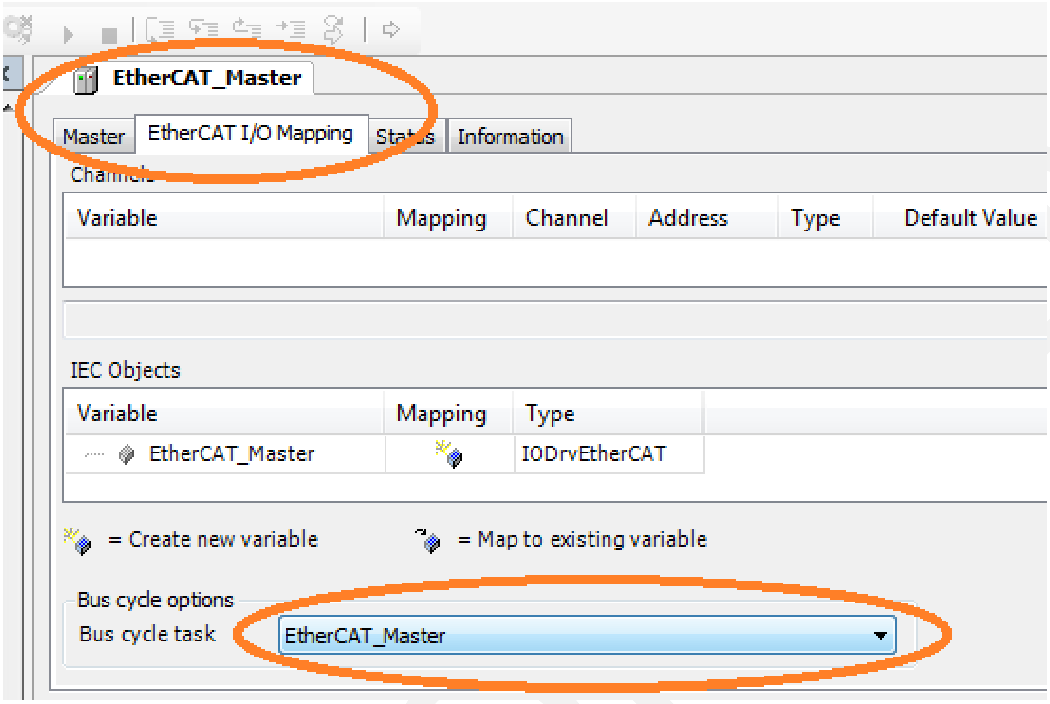
Description	Project	Object	Position

Precompile: OK

Current user: (nobody)

Figure 3.18: The Project after adding the CANopen Manager

Please note that it is importent that the Bus cycle of the EtherCAT master is set to "EtherCAT master task", instead of "parent buscycle"



EtherCAT_Master

Master | EtherCAT I/O Mapping | Status | Information

Channels

Variable	Mapping	Channel	Address	Type	Default Value

IEC Objects

Variable	Mapping	Type
EtherCAT_Master		IODrvEtherCAT

= Create new variable   = Map to existing variable

Bus cycle options

Bus cycle task: EtherCAT_Master

Figure 3.19: Setting the bus cycle

3.4.2 Scan for CANopen slaves

If online the CANopen master can now scan for devices on the network ("Scan for Devices"):

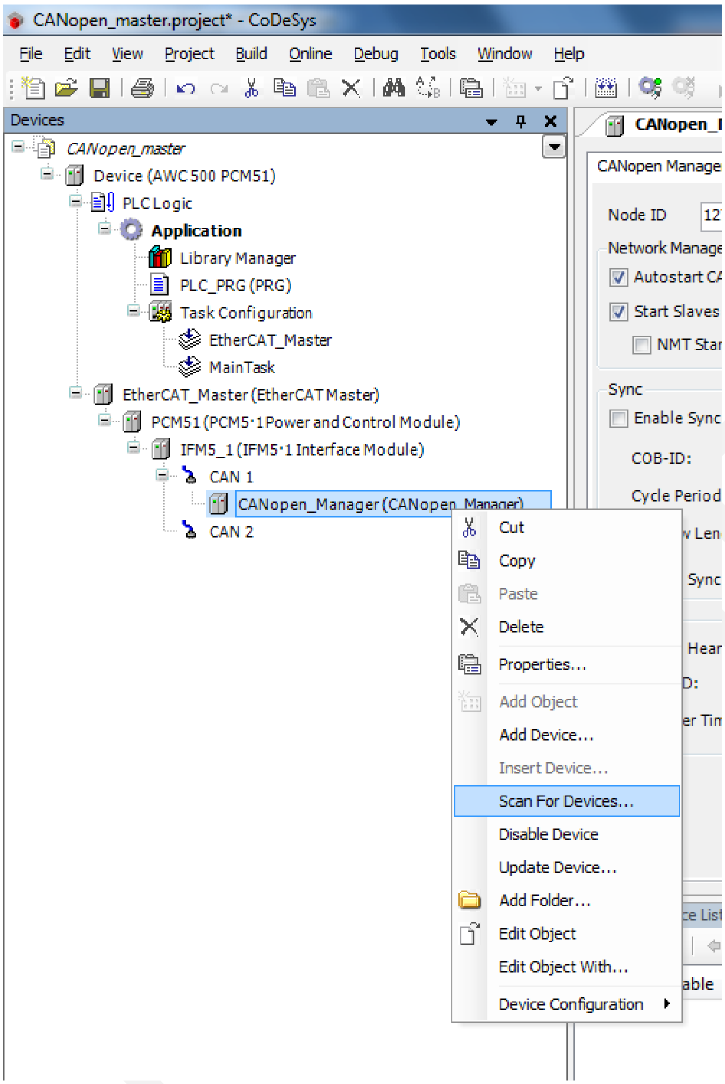
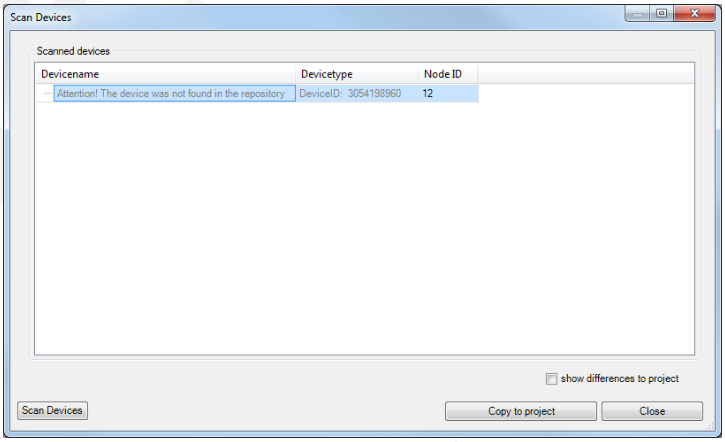


Figure 3.20: Scan for Devices

Then a dialog box with the found devices will be shown.



Scan Devices

Scanned devices

Devicename	Devicetype	Node ID
Attention! The device was not found in the repository	DeviceID: 3054198960	12

□ show differences to project

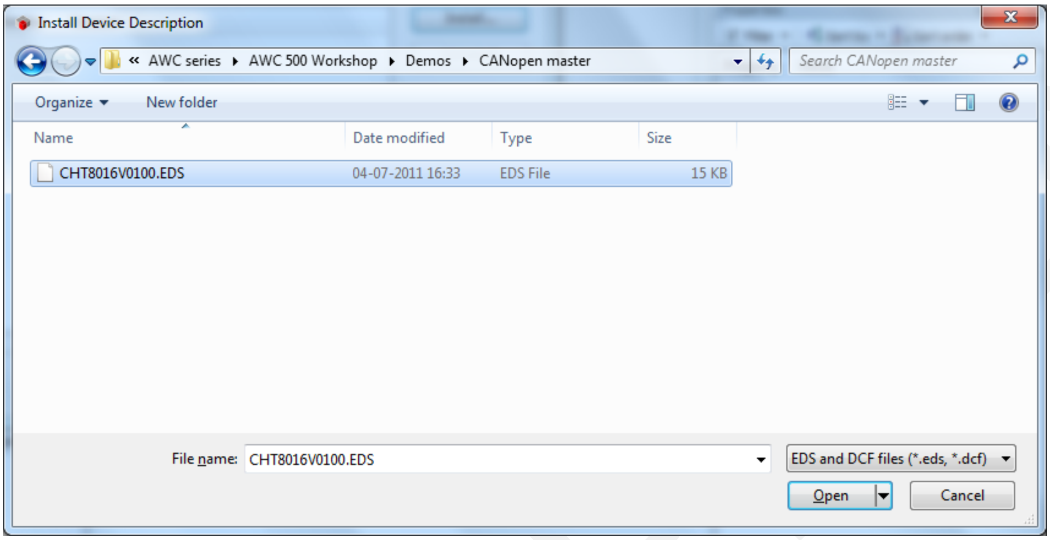
Figure 3.21: Devices found

It is only possible to add the device to Device tree, if it is installed in the Device repository

3.4.3 Manually added via EDS files

Alternatively CANopen slaves can be added to a configuration via EDS files.

Adding a CANopen slave EDS file.
Add a CANopen slaves EDS file via Device repository (Tools→Device repository→Install. . .) Change type to "EDS and DCF files (*.eds, *.dcf)", here an example on a CAN open slave:


Install Device Description
« AWC series ▸ AWC 500 Workshop ▸ Demos ▸ CANopen master
Search CANopen master
Organize ▾
New folder
Open
Cancel

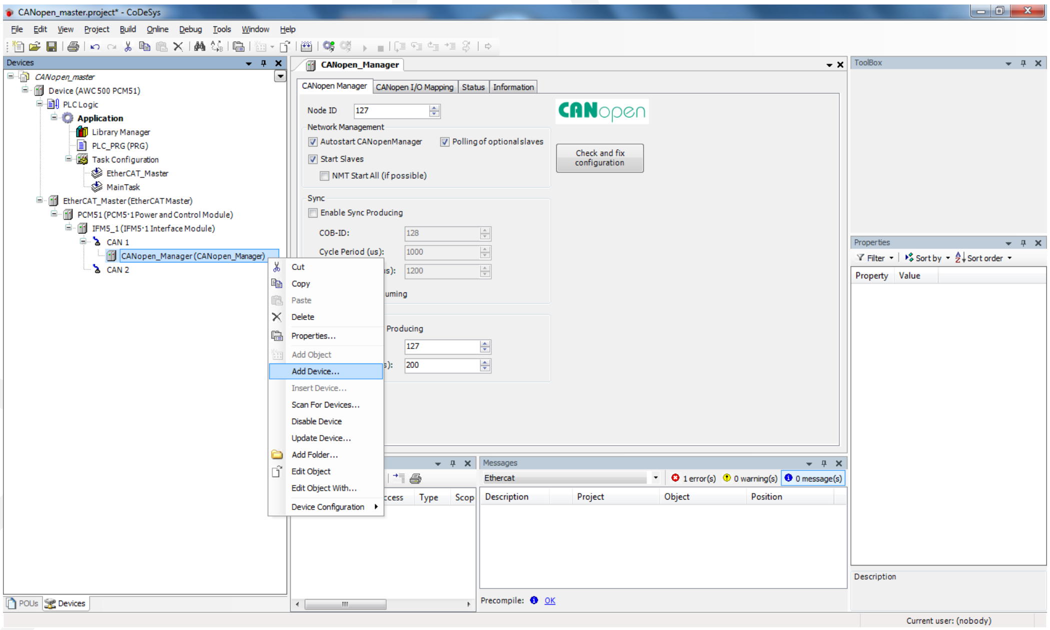
Name	Date modified	Type	Size
CHT8016V0100.EDS	04-07-2011 16:33	EDS File	15 KB

File name: CHT8016V0100.EDS

EDS and DCF files (*.eds, *.dcf)

Figure 3.22: Install Device Description

Now select "Add Device. . . ":



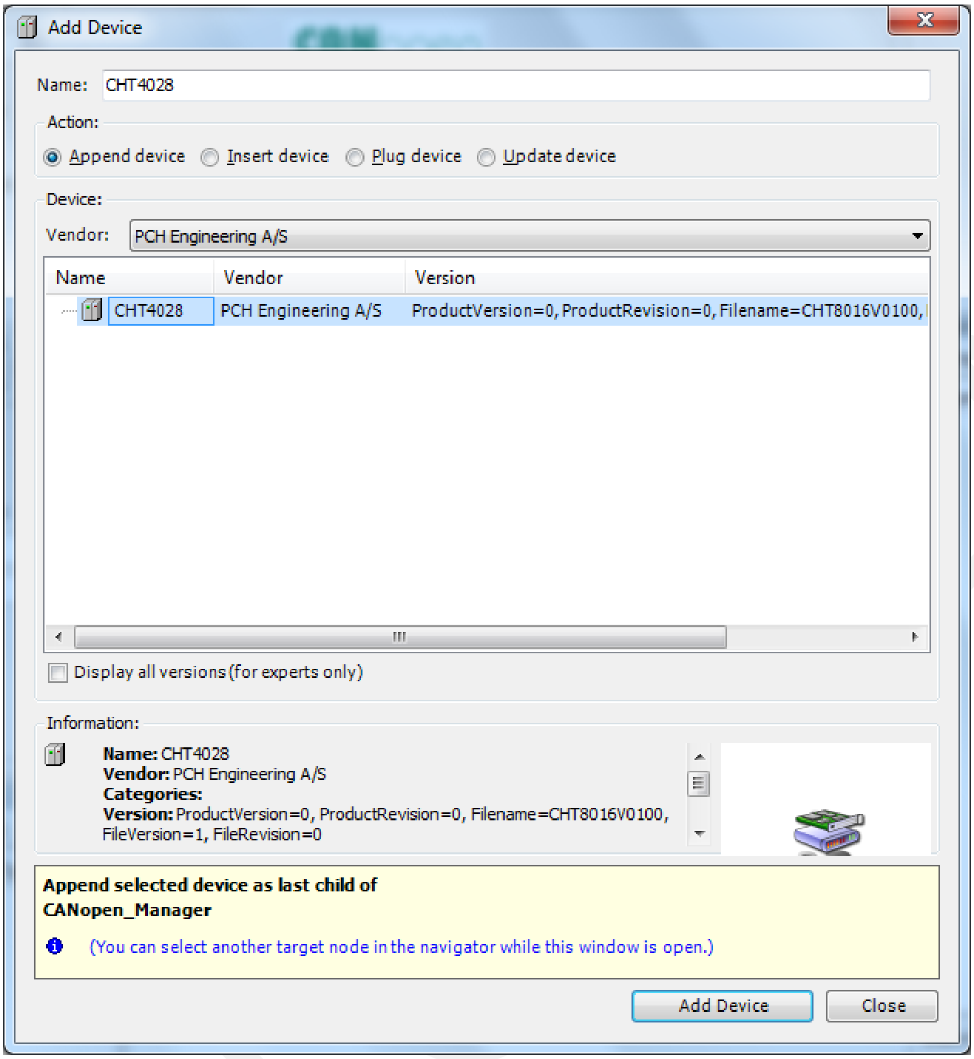
Node ID: 127

Network Management
☑ Autostart CANopenManager ☑ Polling of optional slaves
☑ Start Slaves
☐ NMT Start All (if possible)

Sync
☐ Enable Sync Producing
COB-ID: 128
Cycle Period (us): 1000
s): 1200

Figure 3.23: Add device

Locate the device:



Add Device

Name: CHT4028

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: PCH Engineering A/S

Name	Vendor	Version
CHT4028	PCH Engineering A/S	ProductVersion=0, ProductRevision=0, Filename=CHT8016V0100,

□ Display all versions (for experts only)

Information:

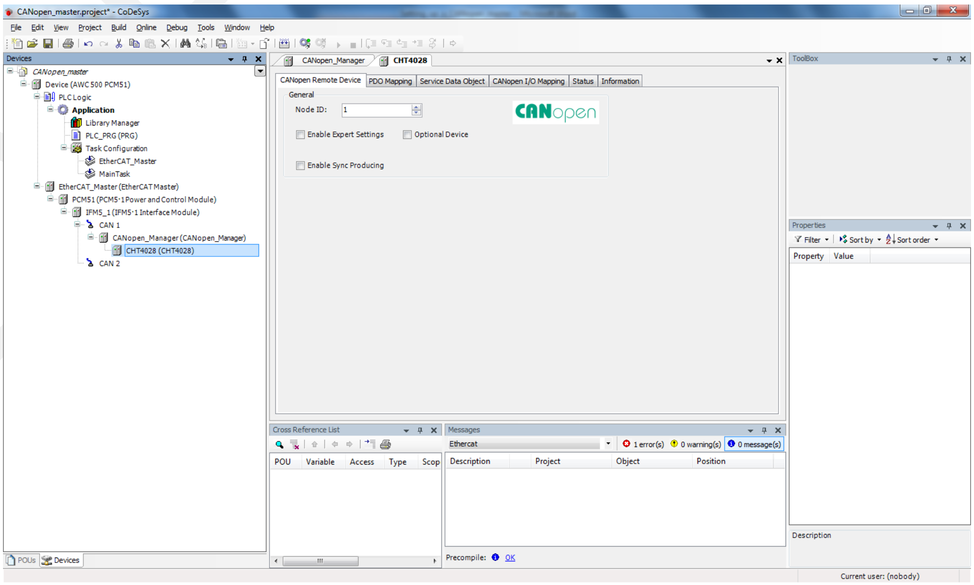
Name: CHT4028
Vendor: PCH Engineering A/S
Categories:
Version: ProductVersion=0, ProductRevision=0, Filename=CHT8016V0100, FileVersion=1, FileRevision=0

Append selected device as last child of CANopen_Manager

(You can select another target node in the navigator while this window is open.)

Figure 3.24: Locate the device

The device now shows up under the CANopen master:



Node ID: 1

□ Enable Expert Settings □ Optional Device

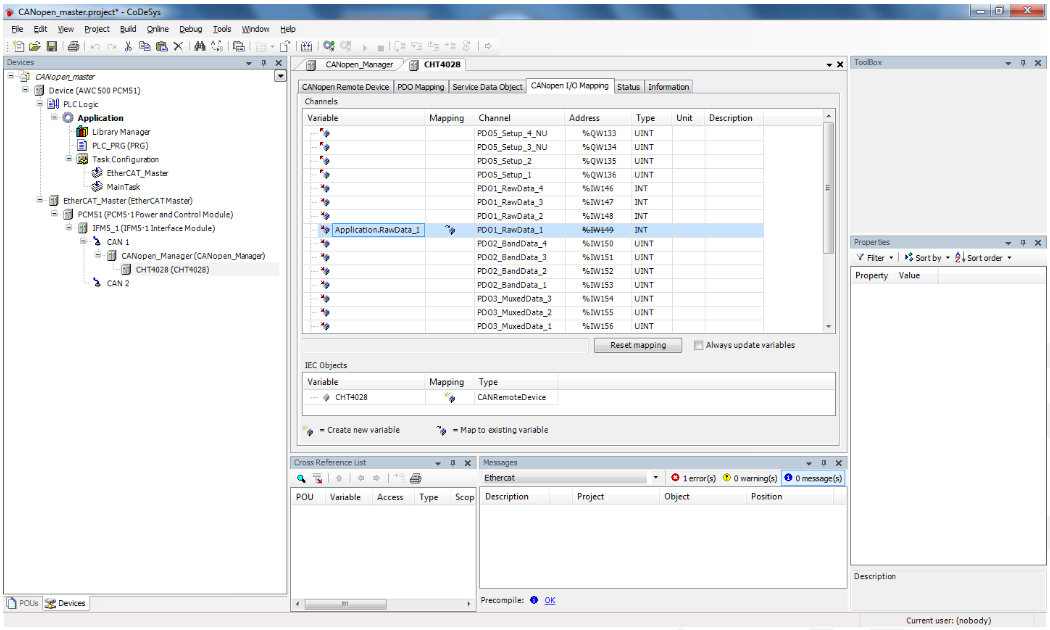
□ Enable Sync Producing

Messages: Ethercat — 1 error(s) 0 warning(s) 0 message(s)

Precompile: OK

Figure 3.25: The device is added under the CANopen master

The data in the PDOs are now ready to link to variables in the PLC program:



CANopen I/O Mapping

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		PDO5_Setup_4_NU	%QW133	UINT		
		PDO5_Setup_3_NU	%QW134	UINT		
		PDO5_Setup_2	%QW135	UINT		
		PDO5_Setup_1	%QW136	UINT		
		PDO1_RawData_4	%IW146	INT		
		PDO1_RawData_3	%IW147	INT		
		PDO1_RawData_2	%IW148	INT		
Application.RawData_1		PDO1_RawData_1	~~%IW149~~	INT		
		PDO2_BandData_4	%IW150	UINT		
		PDO2_BandData_3	%IW151	UINT		
		PDO2_BandData_2	%IW152	UINT		
		PDO2_BandData_1	%IW153	UINT		
		PDO3_MuxedData_3	%IW154	UINT		
		PDO3_MuxedData_2	%IW155	UINT		
		PDO3_MuxedData_1	%IW156	UINT		

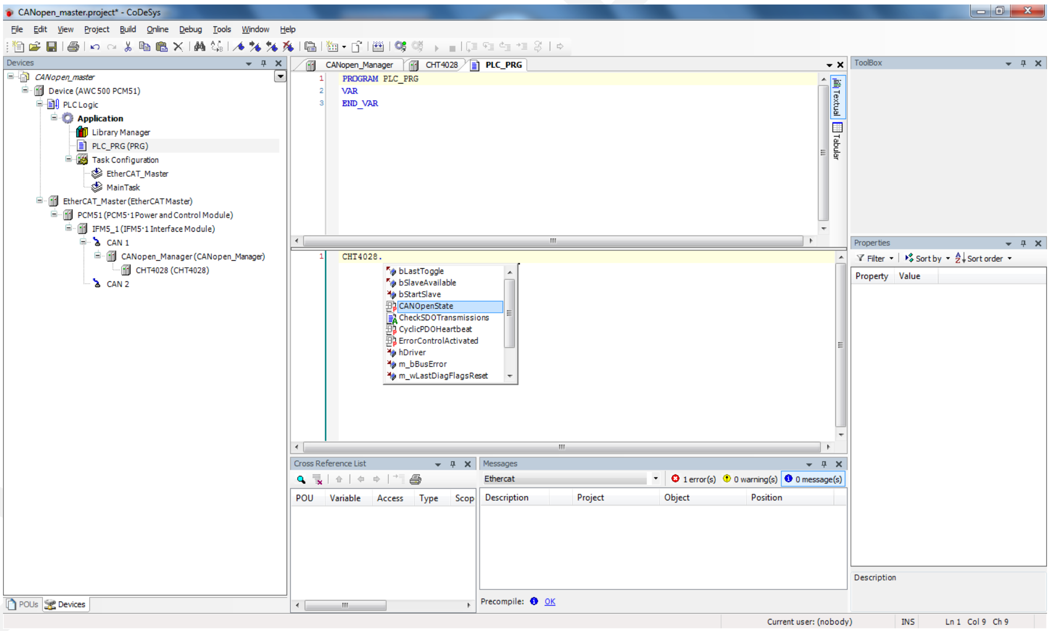
IEC Objects

Variable	Mapping	Type
CHT4028		CANRemoteDevice

Figure 3.26: PDO data ready for linking

3.4.4 Monitoring the CANopen slave status

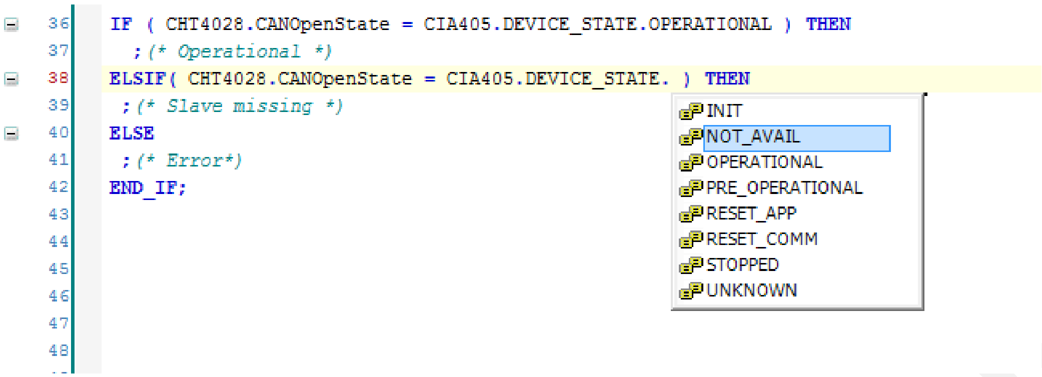
Each CANopen devices added can be access from PLC program via its IEC Object, here CHT4028.



```
PROGRAM PLC_PRG
VAR
END_VAR
```

```
CHT4028.
```

Figure 3.27: Accessing CANopen device from PLC program



```
IF ( CHT4028.CANOpenState = CIA405.DEVICE_STATE.OPERATIONAL ) THEN
  ; (* Operational *)
ELSIF( CHT4028.CANOpenState = CIA405.DEVICE_STATE. ) THEN
 ; (* Slave missing *)
ELSE
 ; (* Error*)
END_IF;
```

Figure 3.28: Accessing CANopen device code sample

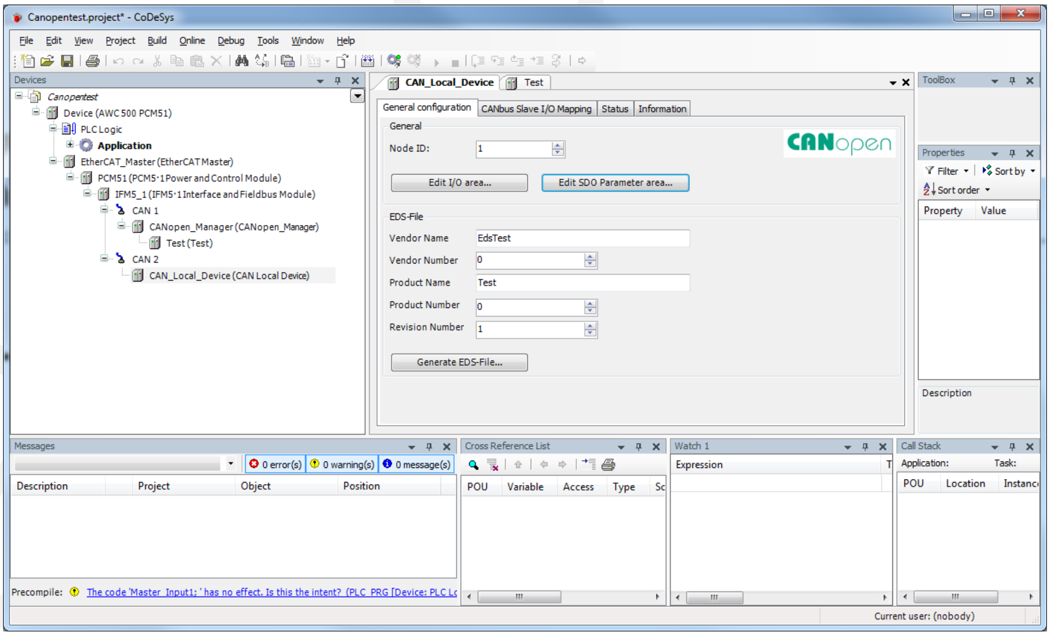
```
IF ( CHT4028.CANOpenState = CIA405.DEVICE_STATE.OPERATIONAL ) THEN
  ;(* Operational *)
ELSIF( CHT4028.CANOpenState = CIA405.DEVICE_STATE.NOT_AVAIL ) THEN
 ;(* Slave missing *)
ELSE
 ;(* Error*)
END_IF;
```

3.4.5 Demo

Optionally see the demo project `\CANopen Master\CANopen_master.project`

3.5 Usage of CANopen SDOs

On the CANopen slave, select "Edit SDO Parameter area. . . "



Node ID: 1

Vendor Name: EdsTest

Vendor Number: 0

Product Name: Test

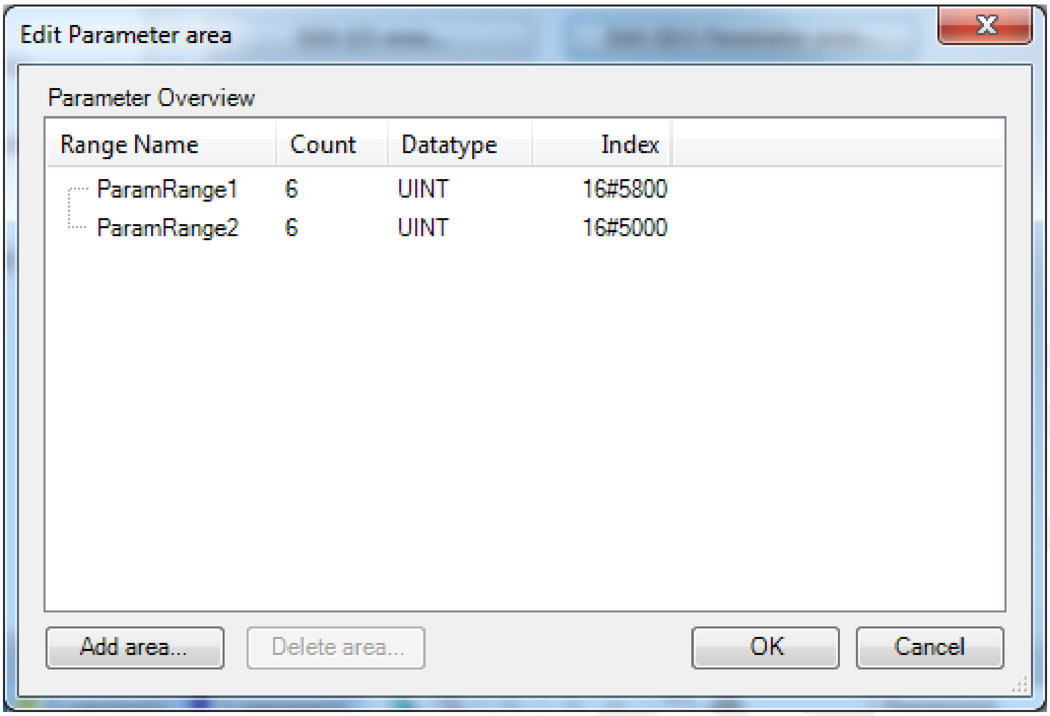
Product Number: 0

Revision Number: 1

Precompile: The code 'Master_Input1;' has no effect. Is this the intent? (PLC_PRG [Device: PLC Lo

Figure 3.29: Click "Edit SDO Parameter area"

Make the changes:



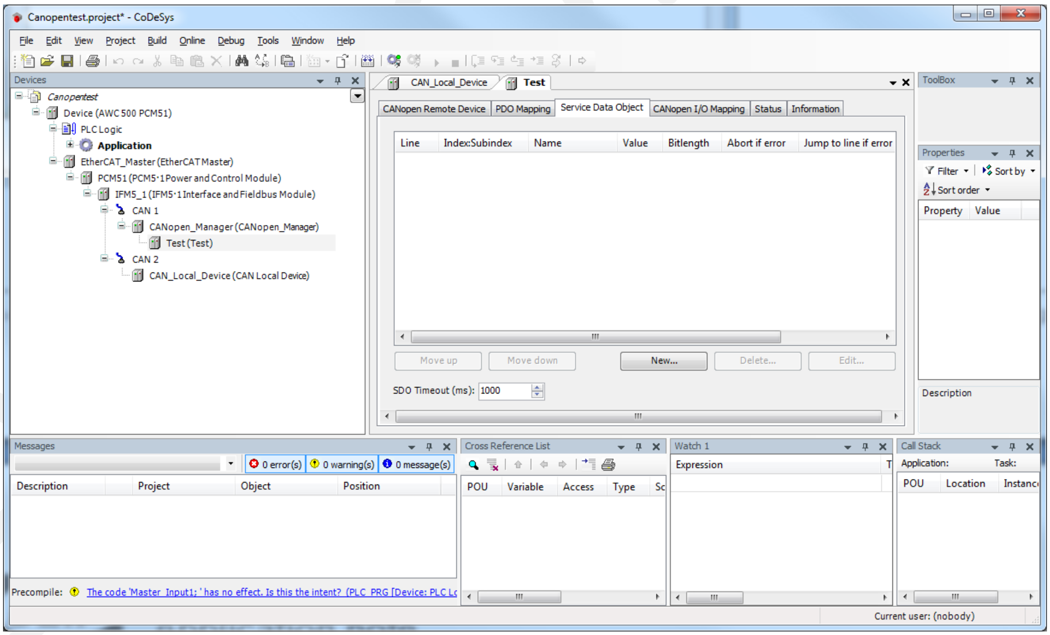
Edit Parameter area

Parameter Overview

Range Name	Count	Datatype	Index
ParamRange1	6	UINT	16#5800
ParamRange2	6	UINT	16#5000

Figure 3.30: Edit Parameter area

Then "Generate EDS-File. . . "
Import the generated EDS file.
To send SDOs to the CANopen slave at start up. Add the SDOs to Service Data Object page. Select "New. . . "



Canopentest.project* - CoDeSys

Devices

- Canopentest
  - Device (AWC 500 PCM51)
    - PLC Logic
      - Application
    - EtherCAT_Master (EtherCAT Master)
      - PCM51 (PCM5·1 Power and Control Module)
        - IFM5_1 (IFM5·1 Interface and Fieldbus Module)
          - CAN 1
            - CANopen_Manager (CANopen_Manager)
              - Test (Test)
          - CAN 2
            - CAN_Local_Device (CAN Local Device)

CAN_Local_Device | Test

CANopen Remote Device | PDO Mapping | Service Data Object | CANopen I/O Mapping | Status | Information

Line	Index:Subindex	Name	Value	Bitlength	Abort if error	Jump to line if error

SDO Timeout (ms): 1000

Messages: 0 error(s) 0 warning(s) 0 message(s)

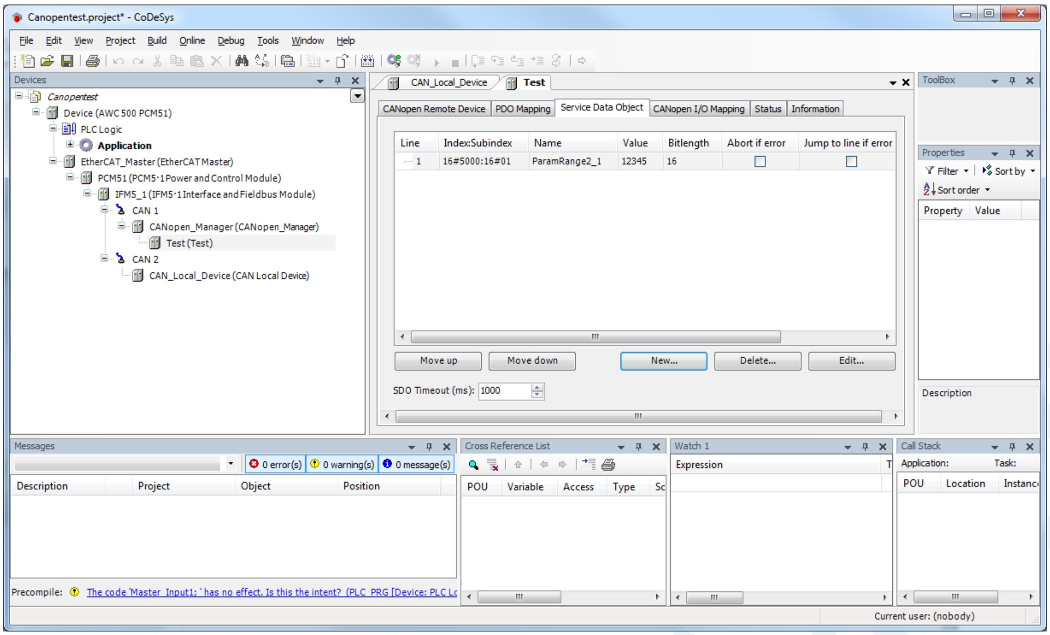
Description	Project	Object	Position

Precompile: The code 'Master_Input1; ' has no effect. Is this the intent? (PLC_PRG [Device: PLC Lc

Current user: (nobody)

Figure 3.31: Select New

From this list the SDO can be selected:



Line	Index:Subindex	Name	Value	Bitlength	Abort if error	Jump to line if error
1	16#5000:16#01	ParamRange2_1	12345	16	□	□

SDO Timeout (ms): 1000

Precompile: The code 'Master_Input1; ' has no effect. Is this the intent? (PLC_PRG [Device: PLC Lc

Figure 3.32: Select SDO

Then the SDOs are sent to the CANopen slave at start up.

For continues exchange for the SDOs to a CANopen slave, see the CoDeSys help on topics
CIA405.SDO_READ (FB)
CIA405.SDO_WRITE (FB)

3.6 Doing CAN layer II communication

3.6.1 Demo

See the demo project `\CAN layer II\testproject-can_layer2_echo.project`

3.7 CAN telegrams debug logging

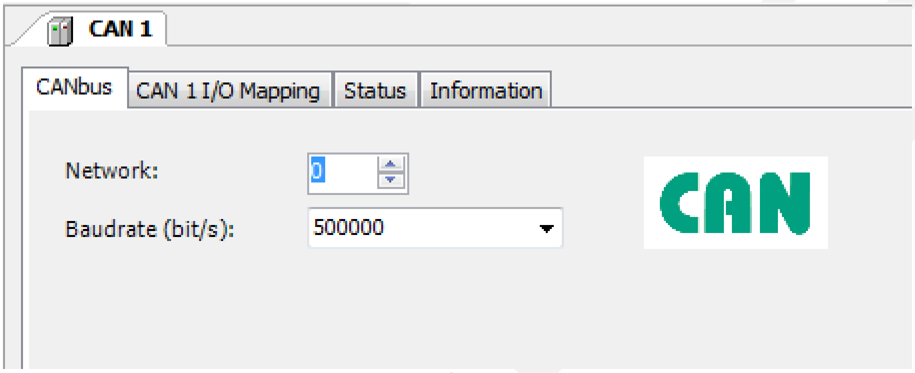
The AWC has build in CANopen logging feature, logging CAN telegrams to the CoDeSys log. It is enabled using the function :

```
FUNCTION IFM51_EnableDebugLog : BOOL
VAR_INPUT
usiNetID: USINT; // Which CAN line you want to filter on (Network ID)
usiSize : USINT; // How may telegrams you want to log before stopping
END_VAR
```

Example: The function must be called once, to setup and initiate the logging.

```
IFM51_EnableDebugLog(usiNetID:= 0, usiSize := 1000);
```

This logs 1000 telegrams from NetID 0, corresponding to Network ID under IFM5.1->CAN1:

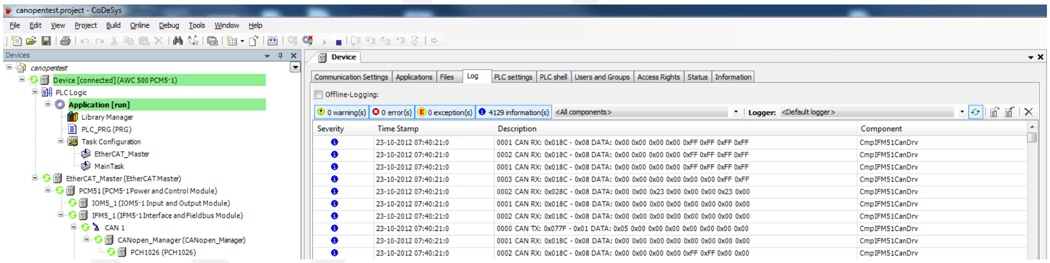


Network: 0

Baudrate (bit/s): 500000

Figure 3.33: CAN network id

Here after the function can be called again to perform another 1000 logs.



□ Offline-Logging:

0 warning(s) 0 error(s) 0 exception(s) 4129 information(s) <All components> Logger: <Default logger>

Severity	Time Stamp	Description	Component
ℹ	23-10-2012 07:40:21:0	0001 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0xFF 0xFF 0xFF 0xFF	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0002 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0xFF 0xFF 0xFF 0xFF	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0001 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0xFF 0xFF 0xFF 0xFF	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0003 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFF	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0002 CAN RX: 0x028C - 0x08 DATA: 0x00 0x00 0x23 0x00 0x00 0x00 0x23 0x00	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0001 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0002 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0000 CAN TX: 0x077F - 0x01 DATA: 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0001 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	CmpIFM51CanDrv
ℹ	23-10-2012 07:40:21:0	0002 CAN RX: 0x018C - 0x08 DATA: 0x00 0x00 0x00 0x00 0xFF 0xFF 0x00 0x00	CmpIFM51CanDrv

Figure 3.34: CAN telegrams logged to the CoDeSys log

ℹ **It is only possible to log from one netID at a time.**

3.8 Setting up OPC connection interface

3.8.1 Overview of concept

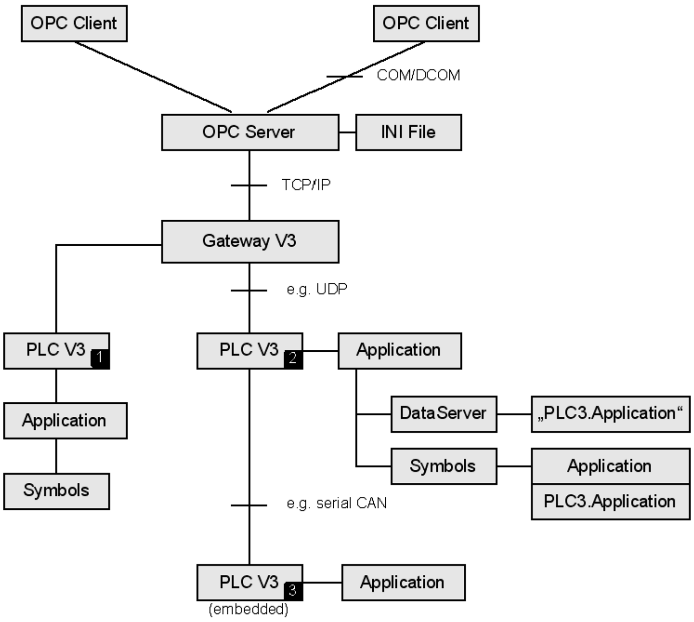
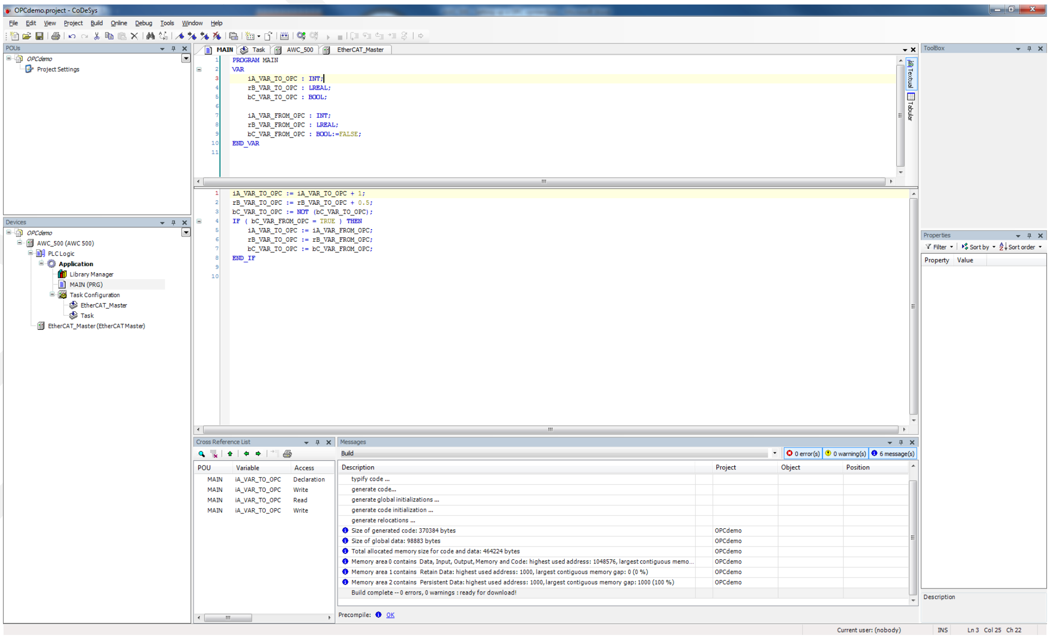


Figure 3.35: Concept overview

3.8.2 Example

Create a PLC project with variables to be exchanged via OPC



```
PROGRAM MAIN
VAR
    iA_VAR_TO_OPC : INT;
    rB_VAR_TO_OPC : LREAL;
    bC_VAR_TO_OPC : BOOL;

    iA_VAR_FROM_OPC : INT;
    rB_VAR_FROM_OPC : LREAL;
    bC_VAR_FROM_OPC : BOOL:=FALSE;
END_VAR
```

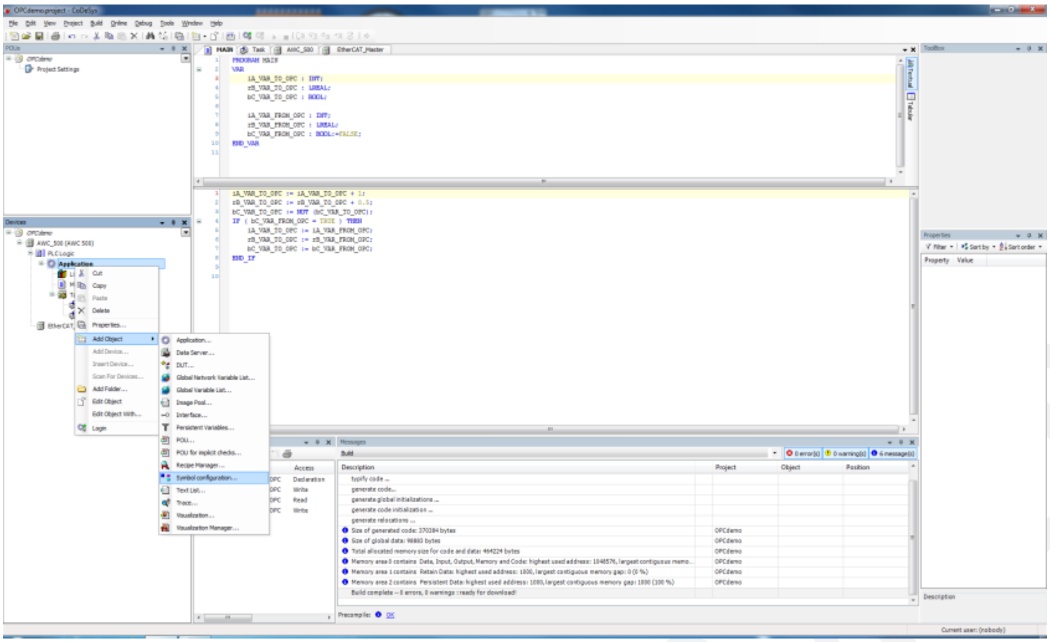
```
iA_VAR_TO_OPC := iA_VAR_TO_OPC + 1;
rB_VAR_TO_OPC := rB_VAR_TO_OPC + 0.5;
bC_VAR_TO_OPC := NOT (bC_VAR_TO_OPC);
IF ( bC_VAR_FROM_OPC = TRUE ) THEN
    iA_VAR_TO_OPC := iA_VAR_FROM_OPC;
    rB_VAR_TO_OPC := rB_VAR_FROM_OPC;
    bC_VAR_TO_OPC := bC_VAR_FROM_OPC;
END_IF
```

POU	Variable	Access
MAIN	iA_VAR_TO_OPC	Declaration
MAIN	iA_VAR_TO_OPC	Write
MAIN	iA_VAR_TO_OPC	Read
MAIN	iA_VAR_TO_OPC	Write

Description	Project	Object	Position
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 370384 bytes	OPCdemo		
Size of global data: 98883 bytes	OPCdemo		
Total allocated memory size for code and data: 464224 bytes	OPCdemo		
Memory area 0 contains Data, Input, Output, Memory and Code: highest used address: 1048576, largest contiguous memo...	OPCdemo		
Memory area 1 contains Retain Data: highest used address: 1000, largest contiguous memory gap: 0 (0 %)	OPCdemo		
Memory area 2 contains Persistent Data: highest used address: 1000, largest contiguous memory gap: 1000 (100 %)	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

Figure 3.36: Project with variables for OPC exchange

Then on Application Add object→Symbol configuration



```
PROGRAM MAIN
VAR
    iA_VAR_TO_OPC : INT;
    rB_VAR_TO_OPC : LREAL;
    bC_VAR_TO_OPC : BOOL;

    iA_VAR_FROM_OPC : INT;
    rB_VAR_FROM_OPC : LREAL;
    bC_VAR_FROM_OPC : BOOL:=FALSE;
END_VAR
```

```
iA_VAR_TO_OPC := iA_VAR_TO_OPC + 1;
rB_VAR_TO_OPC := rB_VAR_TO_OPC + 0.5;
bC_VAR_TO_OPC := NOT (bC_VAR_TO_OPC);
IF ( bC_VAR_FROM_OPC = TRUE ) THEN
    iA_VAR_TO_OPC := iA_VAR_FROM_OPC;
    rB_VAR_TO_OPC := rB_VAR_FROM_OPC;
    bC_VAR_TO_OPC := bC_VAR_FROM_OPC;
END_IF
```

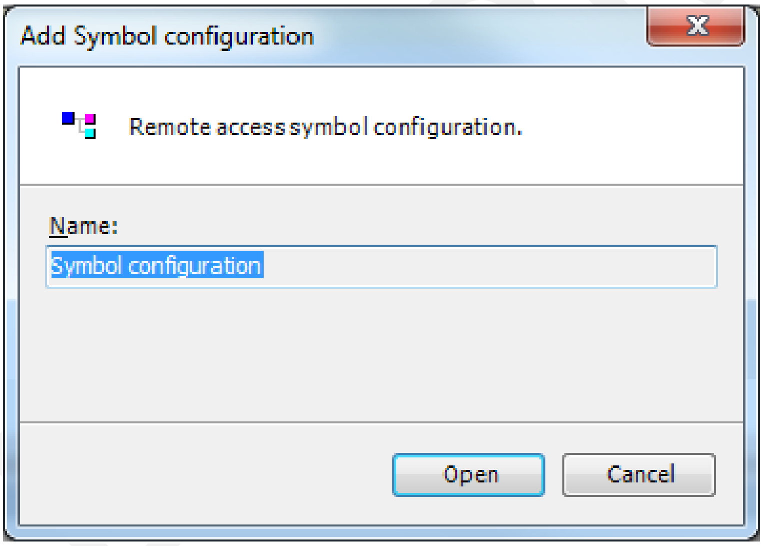
Messages

Build 0 error(s) 0 warning(s) 6 message(s)

Description	Project	Object	Position
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 370384 bytes	OPCdemo		
Size of global data: 98883 bytes	OPCdemo		
Total allocated memory size for code and data: 464224 bytes	OPCdemo		
Memory area 0 contains Data, Input, Output, Memory and Code: highest used address: 1048576, largest contiguous memo...	OPCdemo		
Memory area 1 contains Retain Data: highest used address: 1000, largest contiguous memory gap: 0 (0 %)	OPCdemo		
Memory area 2 contains Persistent Data: highest used address: 1000, largest contiguous memory gap: 1000 (100 %)	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

Precompile: OK

Figure 3.37: Add symbol configuration



Add Symbol configuration

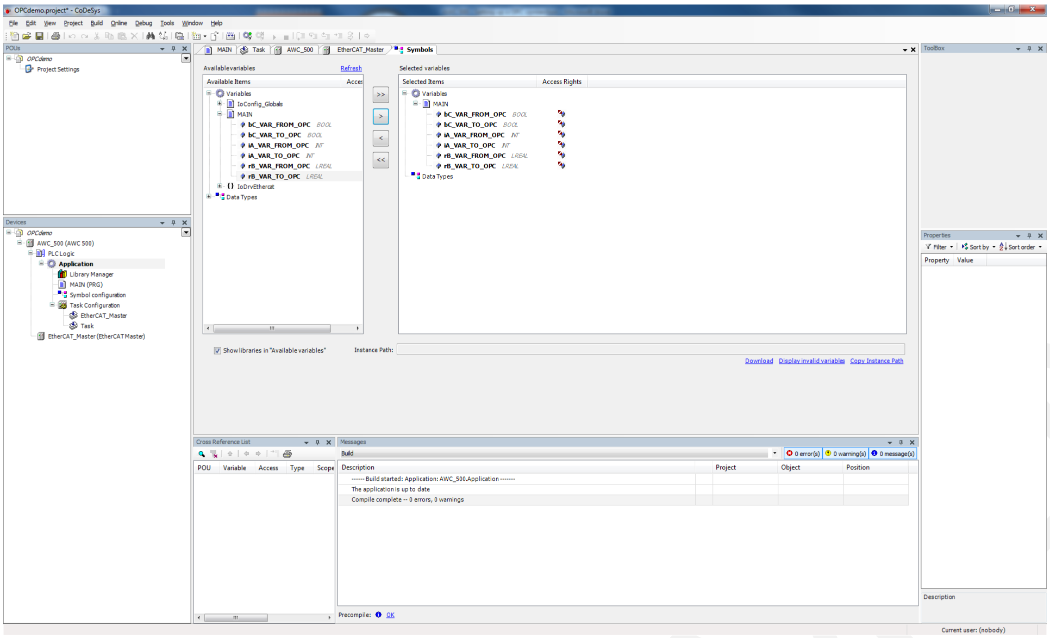
Remote access symbol configuration.

Name:

Symbol configuration

Figure 3.38: Set symbol configuration name

Now select the Items for exchange:



Selected variables

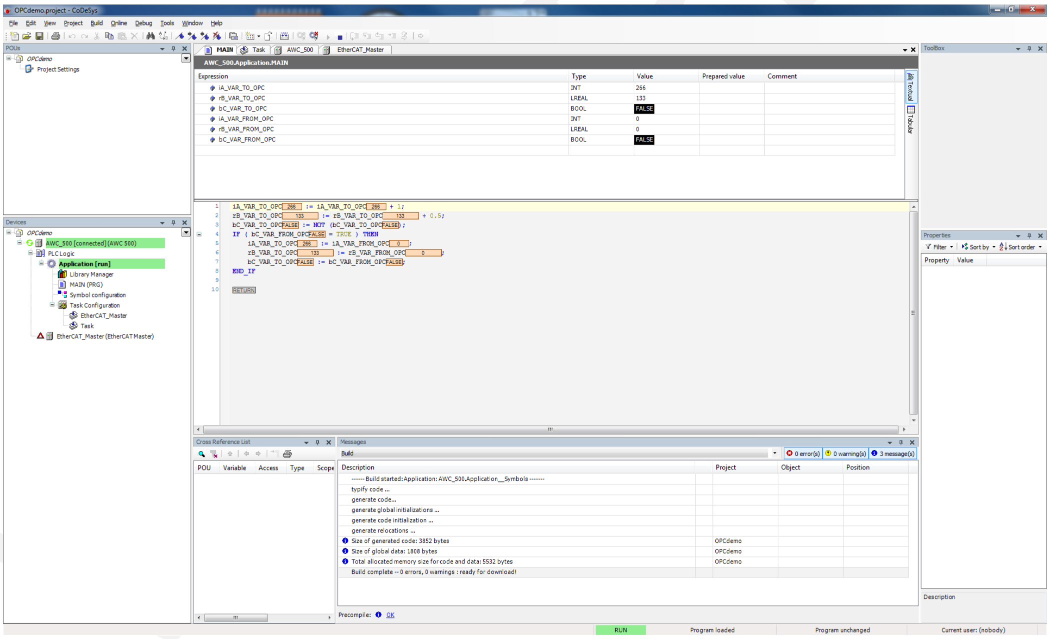
Selected Items	Access Rights
Variables	
MAIN	
bC_VAR_FROM_OPC BOOL	
bC_VAR_TO_OPC BOOL	
iA_VAR_FROM_OPC INT	
iA_VAR_TO_OPC INT	
rB_VAR_FROM_OPC LREAL	
rB_VAR_TO_OPC LREAL	
Data Types	

Description	Project	Object	Position
------ Build started: Application: AWC_500.Application -------			
The application is up to date			
Compile complete -- 0 errors, 0 warnings			

Precompile: OK

Figure 3.39: Select items to exchange

Save project, Build, Download the project and run



AWC_500.Application.MAIN

Expression	Type	Value	Prepared value	Comment
iA_VAR_TO_OPC	INT	266		
rB_VAR_TO_OPC	LREAL	133		
bC_VAR_TO_OPC	BOOL	FALSE		
iA_VAR_FROM_OPC	INT	0		
rB_VAR_FROM_OPC	LREAL	0		
bC_VAR_FROM_OPC	BOOL	FALSE		

```
iA_VAR_TO_OPC := iA_VAR_TO_OPC + 1;
rB_VAR_TO_OPC := rB_VAR_TO_OPC + 0.5;
bC_VAR_TO_OPC := NOT (bC_VAR_TO_OPC);
IF ( bC_VAR_FROM_OPC = TRUE ) THEN
    iA_VAR_TO_OPC := iA_VAR_FROM_OPC;
    rB_VAR_TO_OPC := rB_VAR_FROM_OPC;
    bC_VAR_TO_OPC := bC_VAR_FROM_OPC;
END_IF

RETURN
```

Description	Project	Object	Position
------ Build started:Application: AWC_500.Application__Symbols -------			
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 3852 bytes	OPCdemo		
Size of global data: 1808 bytes	OPCdemo		
Total allocated memory size for code and data: 5532 bytes	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

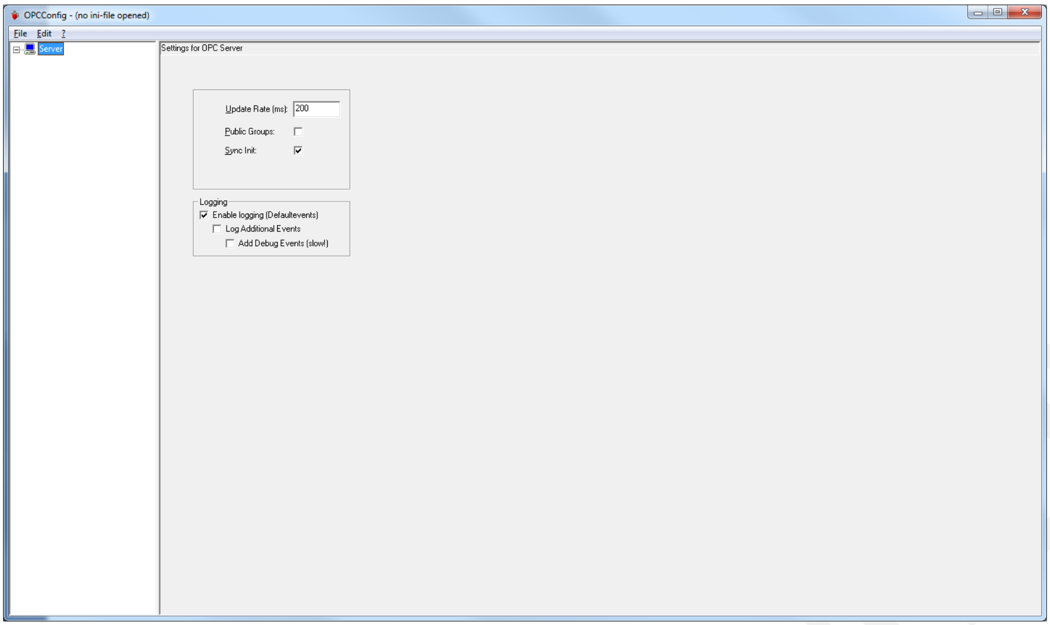
Precompile: OK

RUN Program loaded Program unchanged

Figure 3.40: Running project with OPC

Setting up the OPC server

Open 3S CoDeSys → CoDeSys OPC Server 3→ OPCConfiguration. Select New.



Settings for OPC Server

Update Rate (ms): 200

Public Groups: □

Sync Init: ☑

Logging

☑ Enable logging (Defaultevents)

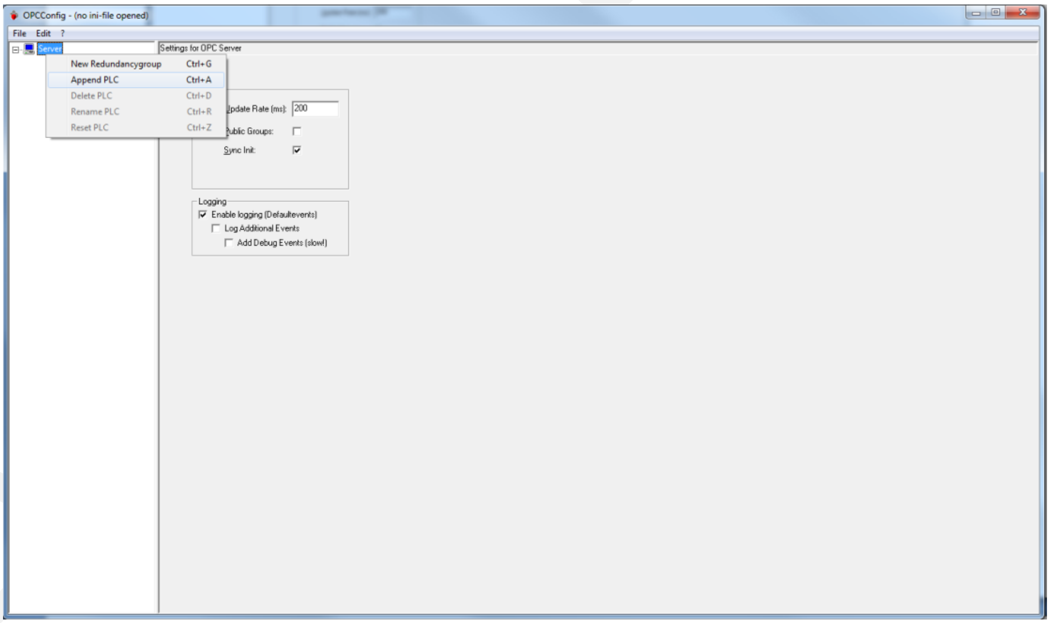
□ Log Additional Events

□ Add Debug Events (slow!)

Figure 3.41: OPC configuration

Change Update Rate (ms) from 200 ms to eg. 1000 ms or 2000 ms, if many variables are to be exchanged via OPC.

Add a PLC



Settings for OPC Server

Update Rate (ms): 200

Public Groups: □

Sync Init: ☑

Logging

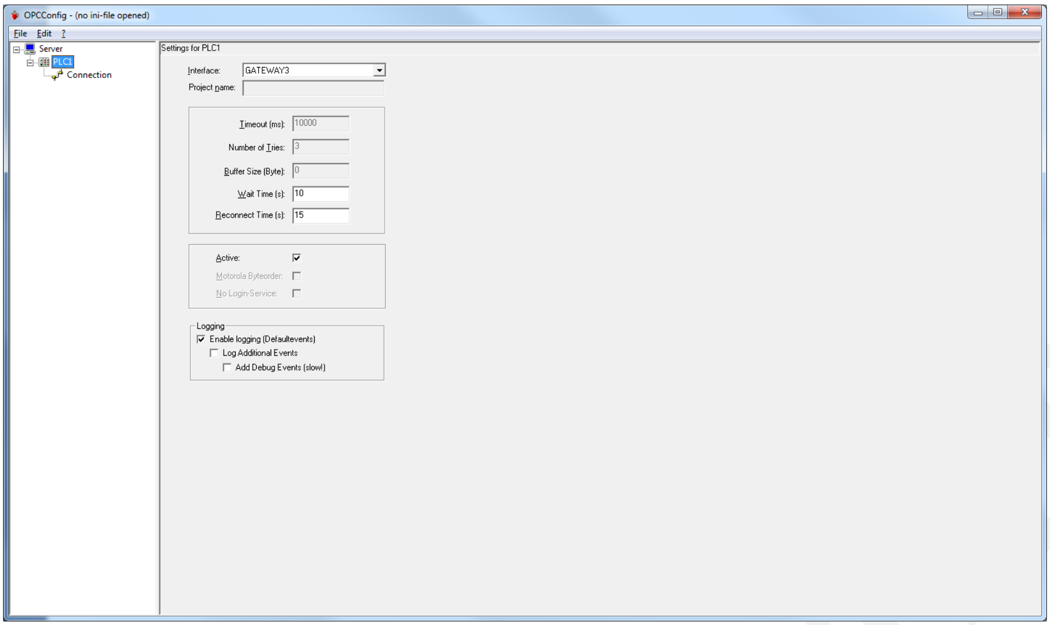
☑ Enable logging (Defaultevents)

□ Log Additional Events

□ Add Debug Events (slow!)

Figure 3.42: Append PLC

Use default settings:



Settings for PLC1

Interface: GATEWAY3

Project name:

Timeout (ms): 10000

Number of Tries: 3

Buffer Size (Byte): 0

Wait Time (s): 10

Reconnect Time (s): 15

Active: ☑

Motorola Byteorder: □

No Login-Service: □

Logging

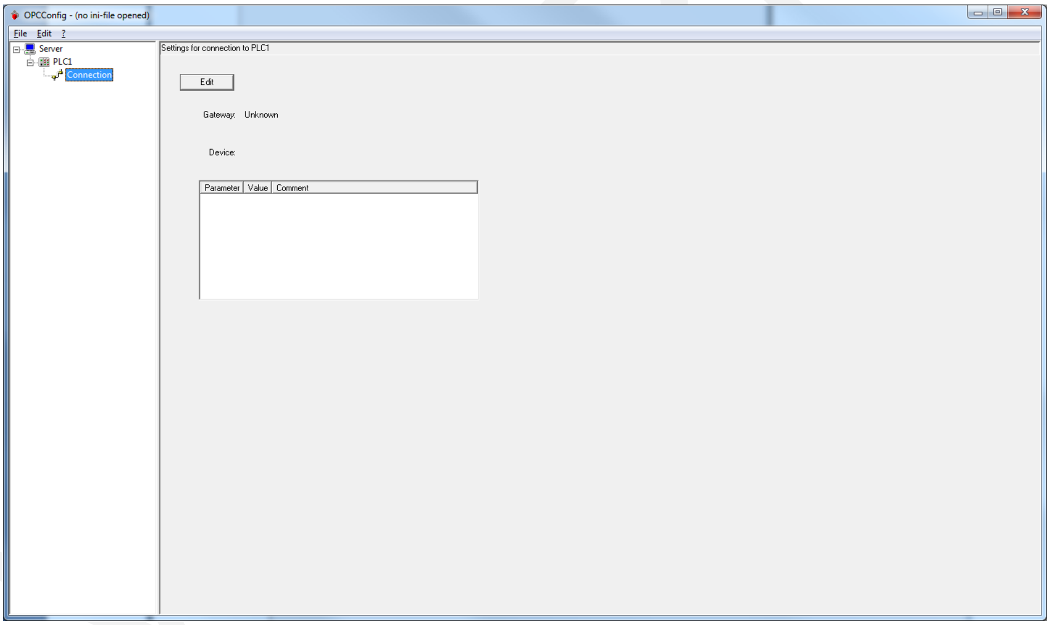
☑ Enable logging (Defaultevents)

□ Log Additional Events

□ Add Debug Events (slow!)

Figure 3.43: Use default settings

Select Edit.



Settings for connection to PLC1

Gateway: Unknown

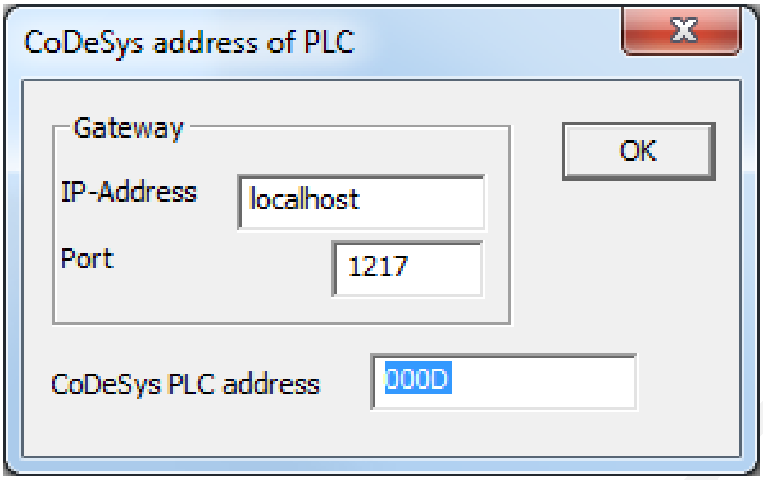
Device:

Parameter	Value	Comment

Figure 3.44: Edit settings

Use the CoDeSys PLC address from the Device Settings in CoDeSys[1]

[1] Please Note these settings are only valid if the AWC 500 is connected to the same subnet.



CoDeSys address of PLC

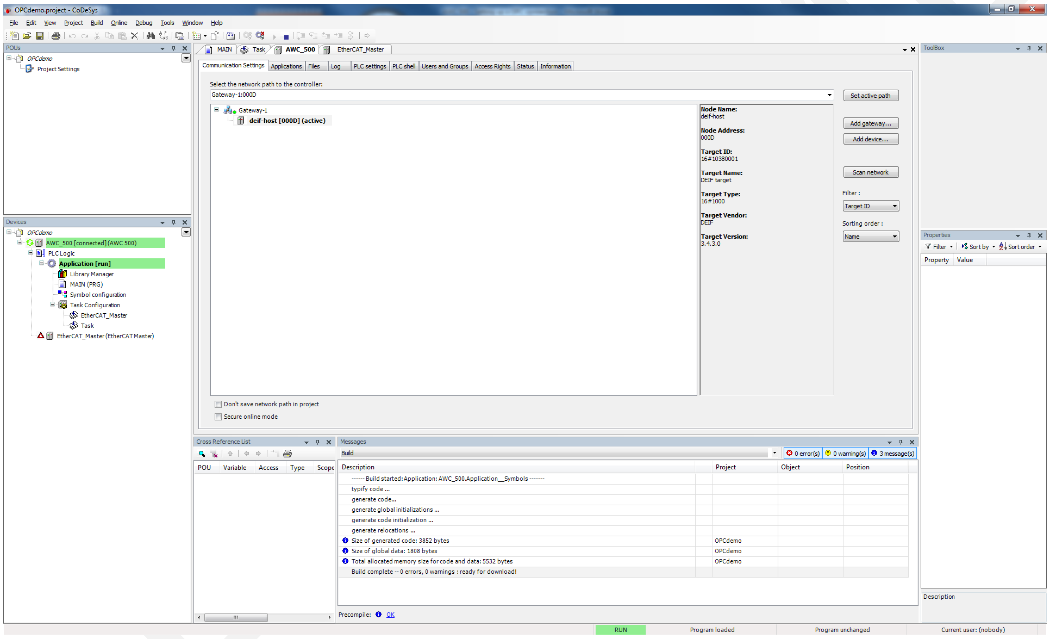
Gateway

IP-Address: localhost

Port: 1217

CoDeSys PLC address: 000D

Figure 3.45: Set the PLC address



POUs:
- OPCdemo
  - Project Settings

Devices:
- OPCdemo
  - AWC_500 [connected] (AWC 500)
    - PLC Logic
      - Application [run]
        - Library Manager
        - MAIN (PRG)
        - Symbol configuration
        - Task Configuration
          - EtherCAT_Master
          - Task
    - EtherCAT_Master (EtherCAT Master)

Select the network path to the controller: Gateway-1:000D

- Gateway-1
  - deif-host [000D] (active)

Node Name: deif-host

Node Address: 000D

Target ID: 16#10380001

Target Name: DEIF target

Target Type: 16#1000

Target Vendor: DEIF

Target Version: 3.4.3.0

Filter: Target ID

Sorting order: Name

□ Don't save network path in project

□ Secure online mode

Build — 0 error(s) 0 warning(s) 3 message(s)

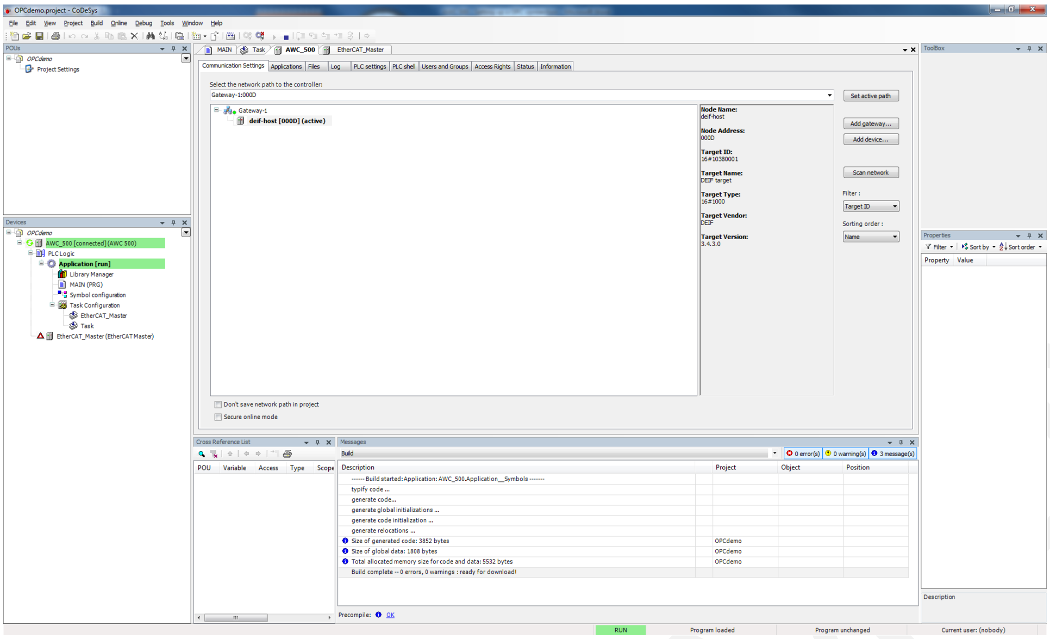
Description	Project	Object	Position
------ Build started: Application: AWC_500.Application__Symbols -------			
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 3852 bytes	OPCdemo		
Size of global data: 1808 bytes	OPCdemo		
Total allocated memory size for code and data: 5532 bytes	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

Precompile: OK

RUN | Program loaded | Program unchanged | Current user: (nobody)

Figure 3.46: The project after configuration

IMPORTANT: Save the configuration to: `C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\OPCServer.ini`
Additionally more PLCs can be added to the OPC server configuration.



Select the network path to the controller: Gateway-1:000D

Node Name: deif-host

Node Address: 000D

Target ID: 16#10380001

Target Name: DEIF target

Target Type: 16#1000

Target Vendor: DEIF

Target Version: 3.4.3.0

Filter : Target ID

Sorting order : Name

□ Don't save network path in project

□ Secure online mode

Build — 0 error(s) 0 warning(s) 3 message(s)

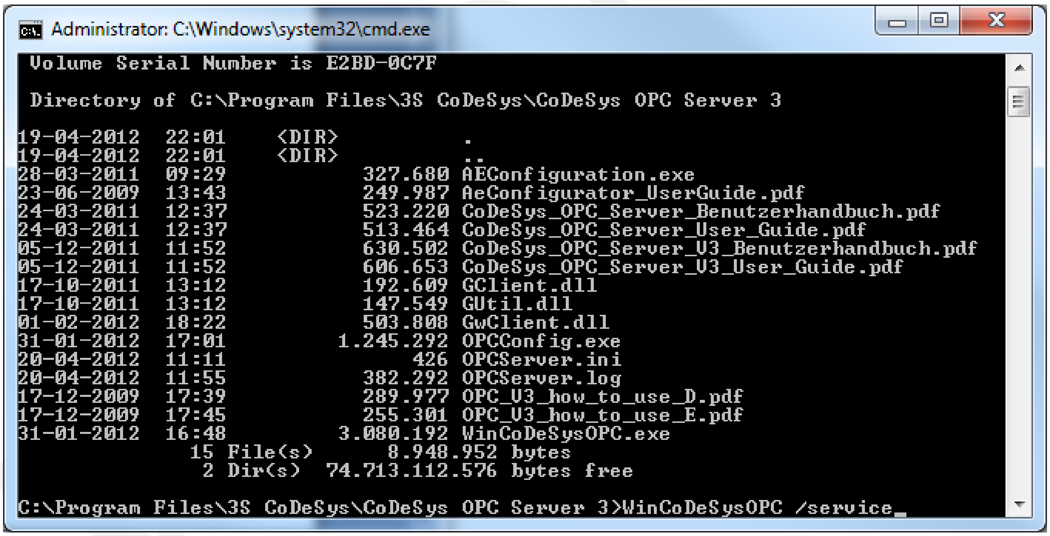
Description	Project	Object	Position
------ Build started:Application: AWC_500.Application__Symbols -------			
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 3852 bytes	OPCdemo		
Size of global data: 1808 bytes	OPCdemo		
Total allocated memory size for code and data: 5532 bytes	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

Precompile: OK

Figure 3.47: More PLCs can be added to the OPC server configuration

Starting the OPC server

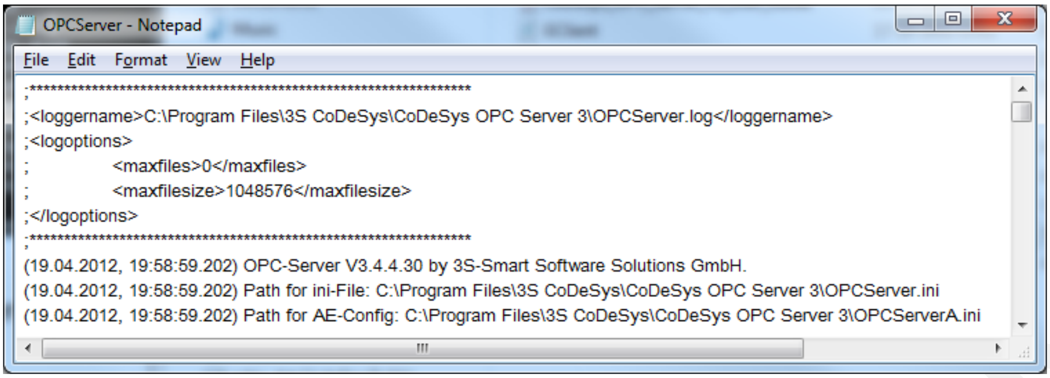
WinCoDeSysOPC /service



```
 Volume Serial Number is E2BD-0C7F

 Directory of C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3

19-04-2012  22:01    <DIR>          .
19-04-2012  22:01    <DIR>          ..
28-03-2011  09:29           327.680 AEConfiguration.exe
23-06-2009  13:43           249.987 AeConfigurator_UserGuide.pdf
24-03-2011  12:37           523.220 CoDeSys_OPC_Server_Benutzerhandbuch.pdf
24-03-2011  12:37           513.464 CoDeSys_OPC_Server_User_Guide.pdf
05-12-2011  11:52           630.502 CoDeSys_OPC_Server_V3_Benutzerhandbuch.pdf
05-12-2011  11:52           606.653 CoDeSys_OPC_Server_V3_User_Guide.pdf
17-10-2011  13:12           192.609 GClient.dll
17-10-2011  13:12           147.549 GUtil.dll
01-02-2012  18:22           503.808 GwClient.dll
31-01-2012  17:01         1.245.292 OPCConfig.exe
20-04-2012  11:11               426 OPCServer.ini
20-04-2012  11:55           382.292 OPCServer.log
17-12-2009  17:39           289.977 OPC_V3_how_to_use_D.pdf
17-12-2009  17:45           255.301 OPC_V3_how_to_use_E.pdf
31-01-2012  16:48         3.080.192 WinCoDeSysOPC.exe
              15 File(s)      8.948.952 bytes
               2 Dir(s)  74.713.112.576 bytes free

C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3>WinCoDeSysOPC /service_
```

Figure 3.48: Starting the OPC server

Logs can be found in:
`C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\OPCServer.txt`



```
;*****************************************************************
;<loggername>C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\OPCServer.log</loggername>
;<logoptions>
;              <maxfiles>0</maxfiles>
;              <maxfilesize>1048576</maxfilesize>
;</logoptions>
;*****************************************************************
(19.04.2012, 19:58:59.202) OPC-Server V3.4.4.30 by 3S-Smart Software Solutions GmbH.
(19.04.2012, 19:58:59.202) Path for ini-File: C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\OPCServer.ini
(19.04.2012, 19:58:59.202) Path for AE-Config: C:\Program Files\3S CoDeSys\CoDeSys OPC Server 3\OPCServerA.ini
```

Figure 3.49: Log example

Testing the connection

Start the OPC test client:

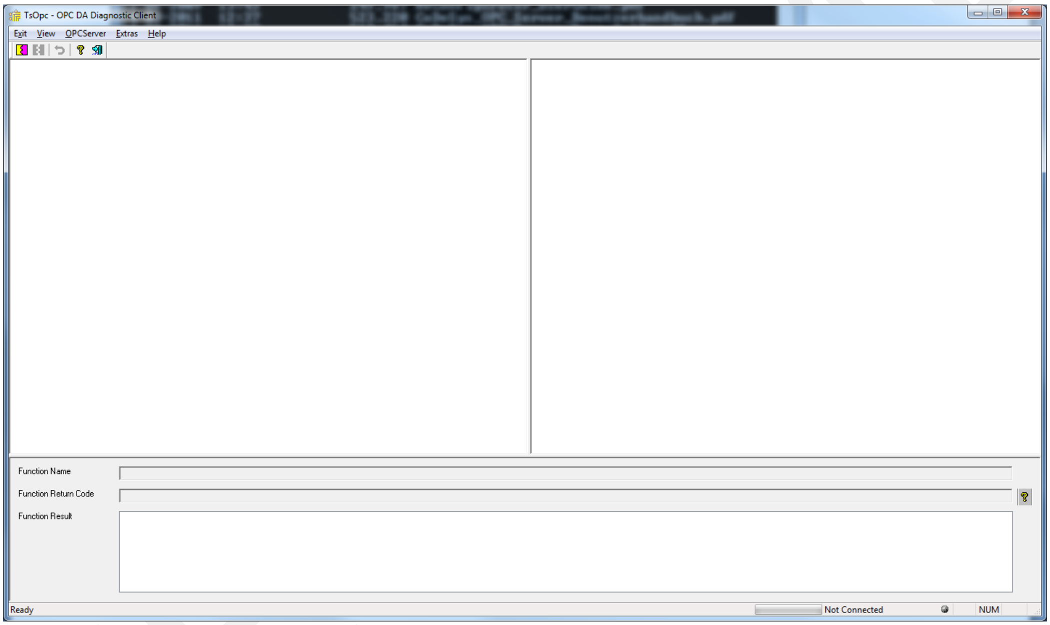
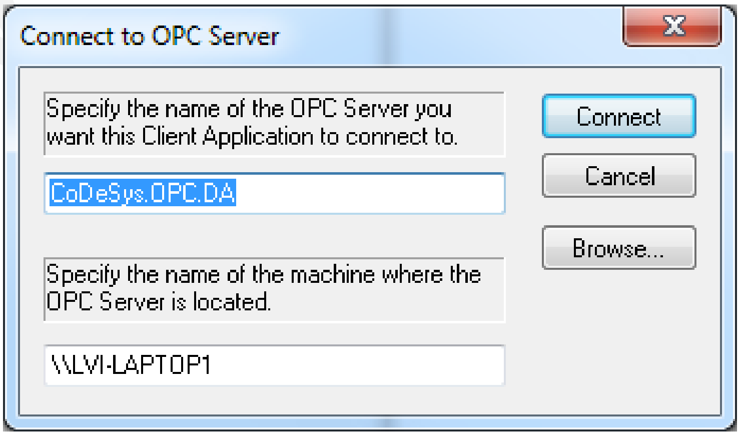


Figure 3.50: TsOpc test client

Connect to CoDeSys.OPC.DA OPCServer →Connect



Specify the name of the OPC Server you want this Client Application to connect to.

CoDeSys.OPC.DA

Specify the name of the machine where the OPC Server is located.

\\LVI-LAPTOP1

Figure 3.51: Connect to OPC Server

Right click Private Groups in right pane:

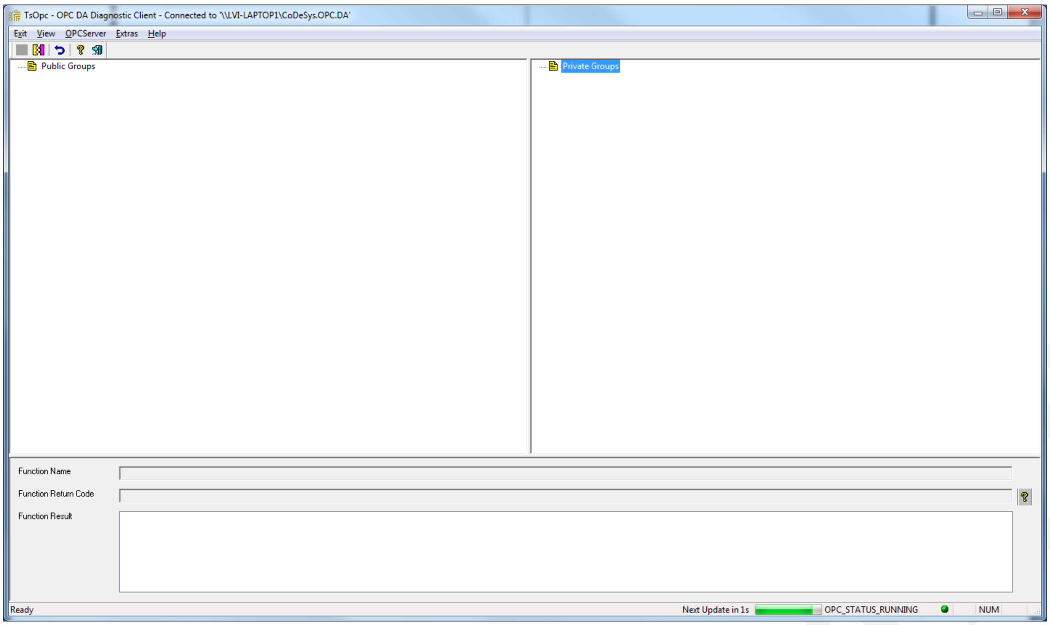
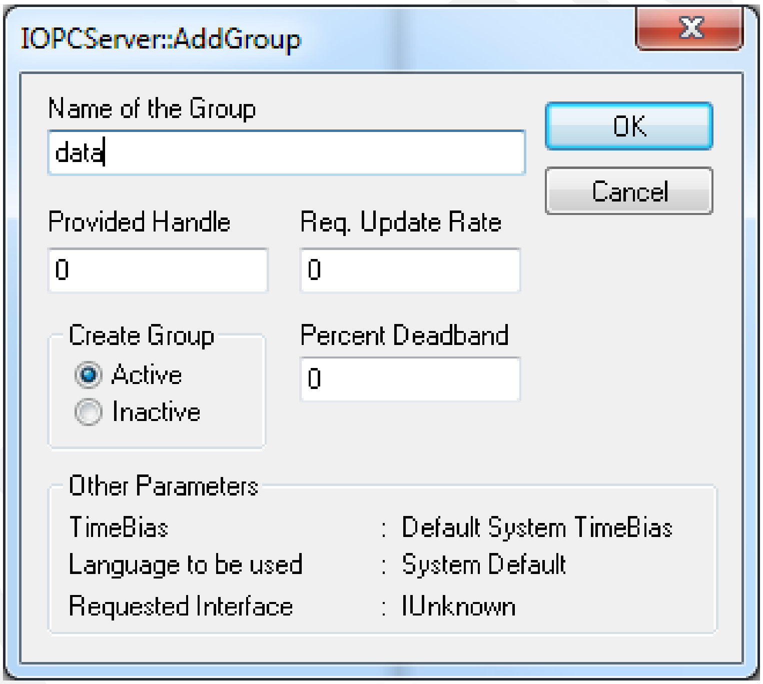


Figure 3.52: Right click Private Groups



Name of the Group: data

Provided Handle: 0

Req. Update Rate: 0

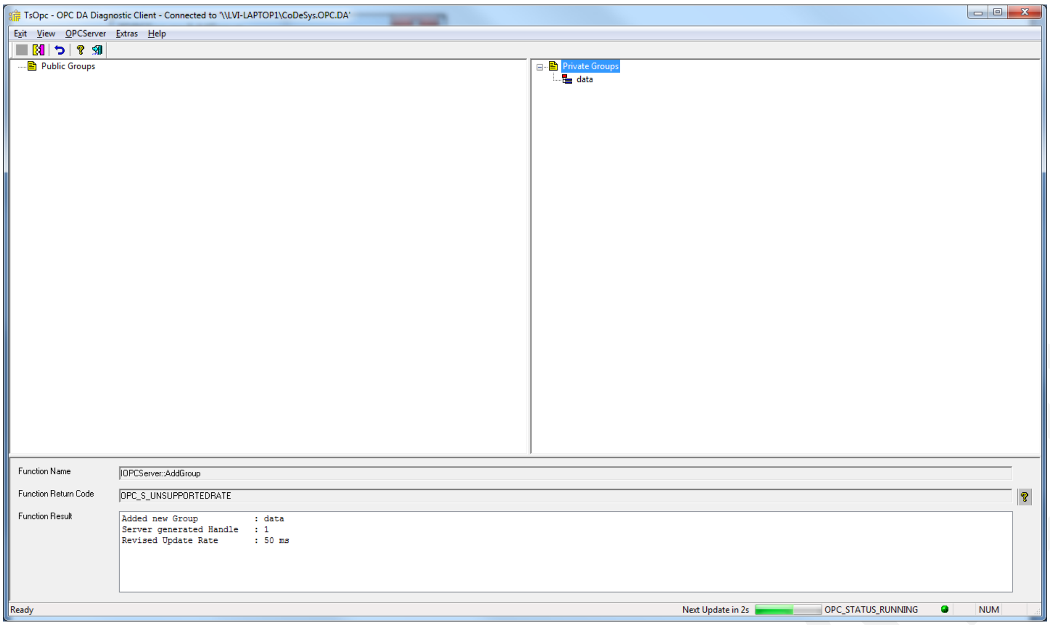
Create Group
● Active
○ Inactive

Percent Deadband: 0

Other Parameters

TimeBias	: Default System TimeBias
Language to be used	: System Default
Requested Interface	: IUnknown

Figure 3.53: Add group


TsOpc - OPC DA Diagnostic Client - Connected to '\\LVI-LAPTOP1\CoDeSys.OPC.DA'
Exit View OPCServer Extras Help
Public Groups
Private Groups
data
Ready
Next Update in 2s
OPC_STATUS_RUNNING
NUM

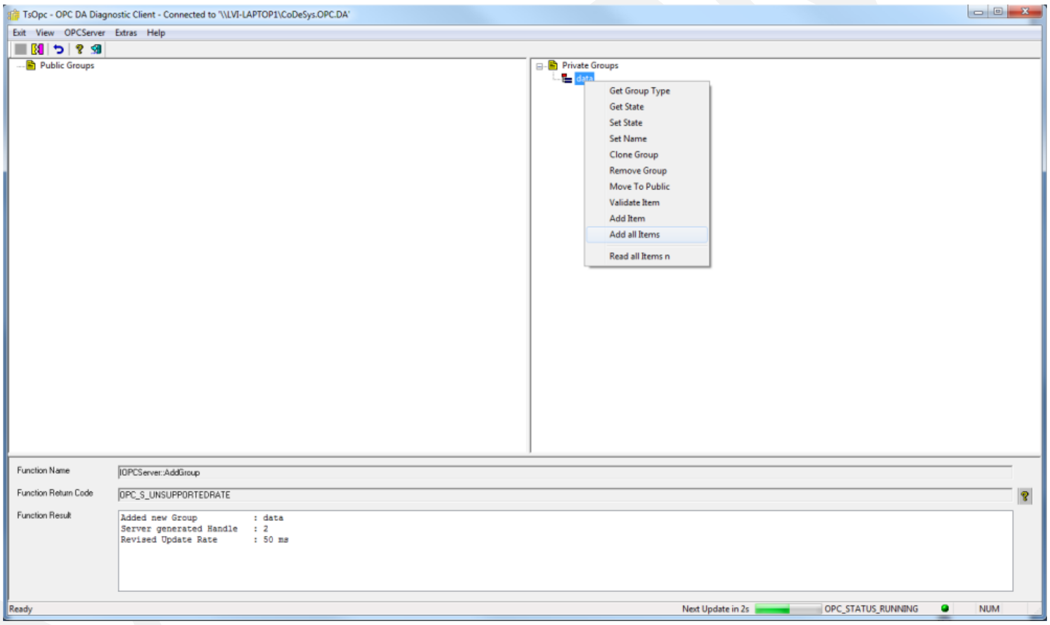
Function Name: IOPCServer::AddGroup

Function Return Code: OPC_S_UNSUPPORTEDRATE

Function Result:

```
Added new Group          : data
Server generated Handle  : 1
Revised Update Rate      : 50 ms
```

Figure 3.54: Group added to Private Groups



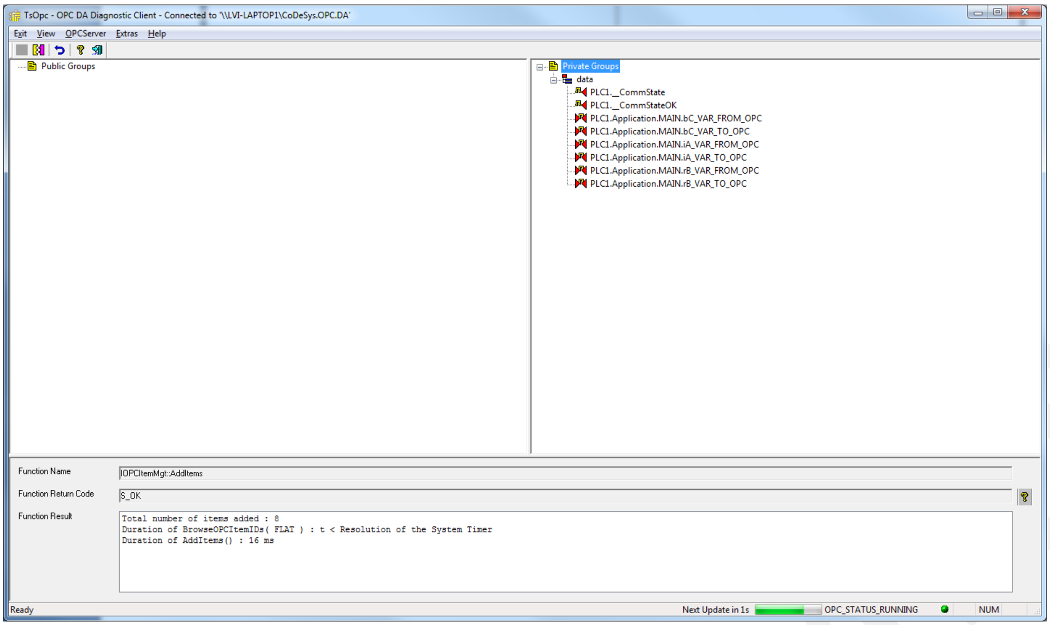
Function Name: IOPCServer::AddGroup

Function Return Code: OPC_S_UNSUPPORTEDRATE

Function Result:

```
Added new Group          : data
Server generated Handle  : 2
Revised Update Rate      : 50 ms
```

Figure 3.55: Add all Items


TsOpc - OPC DA Diagnostic Client - Connected to '\\LVI-LAPTOP1\CoDeSys.OPC.DA'
Exit
View
OPCServer
Extras
Help

Public Groups

Private Groups
data
PLC1.__CommState
PLC1.__CommStateOK
PLC1.Application.MAIN.bC_VAR_FROM_OPC
PLC1.Application.MAIN.bC_VAR_TO_OPC
PLC1.Application.MAIN.iA_VAR_FROM_OPC
PLC1.Application.MAIN.iA_VAR_TO_OPC
PLC1.Application.MAIN.rB_VAR_FROM_OPC
PLC1.Application.MAIN.rB_VAR_TO_OPC

Function Name: IOPCItemMgt::AddItems

Function Return Code: S_OK

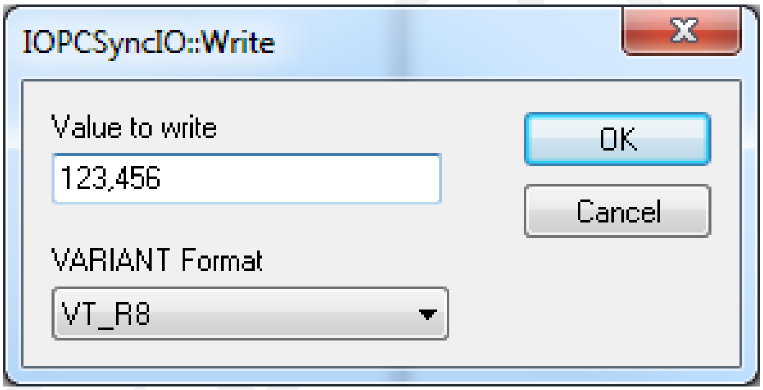
Function Result:

```
Total number of items added : 8
Duration of BrowseOPCItemIDs( FLAT ) : t < Resolution of the System Timer
Duration of AddItems() : 16 ms
```

Ready   Next Update in 1s   OPC_STATUS_RUNNING   NUM

Figure 3.56: Items

Right click each item to read or write E.g. rB_VAR_FROM_OPC

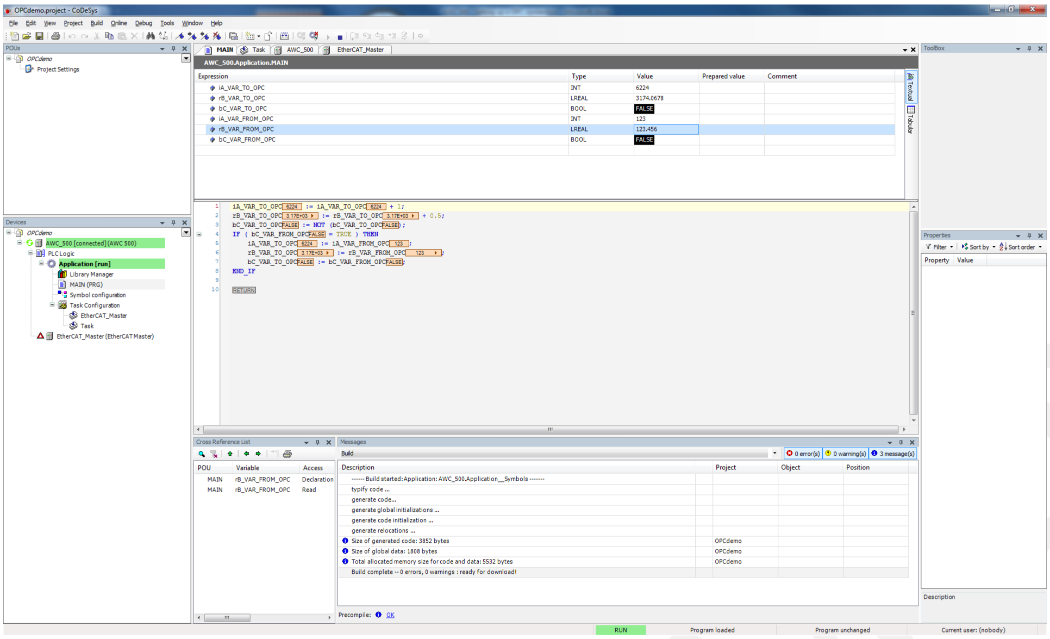


Value to write: 123,456

VARIANT Format: VT_R8

Figure 3.57: rB_VAR_FROM_OPC

Then the item becomes set in PLC:



AWC_500.Application.MAIN

Expression	Type	Value	Prepared value	Comment
iA_VAR_TO_OPC	INT	6224		
rB_VAR_TO_OPC	LREAL	3174.0678		
bC_VAR_TO_OPC	BOOL	FALSE		
iA_VAR_FROM_OPC	INT	123		
rB_VAR_FROM_OPC	LREAL	123.456		
bC_VAR_FROM_OPC	BOOL	FALSE		

```
iA_VAR_TO_OPC 6224 := iA_VAR_TO_OPC 6224 + 1;
rB_VAR_TO_OPC 3.17E+03 := rB_VAR_TO_OPC 3.17E+03 + 0.5;
bC_VAR_TO_OPC FALSE := NOT (bC_VAR_TO_OPC FALSE);
IF ( bC_VAR_FROM_OPC FALSE = TRUE ) THEN
    iA_VAR_TO_OPC 6224 := iA_VAR_FROM_OPC 123;
    rB_VAR_TO_OPC 3.17E+03 := rB_VAR_FROM_OPC 123;
    bC_VAR_TO_OPC FALSE := bC_VAR_FROM_OPC FALSE;
END_IF

RETURN
```

Cross Reference List

POU	Variable	Access
MAIN	rB_VAR_FROM_OPC	Declaration
MAIN	rB_VAR_FROM_OPC	Read

Messages — Build — 0 error(s) 0 warning(s) 3 message(s)

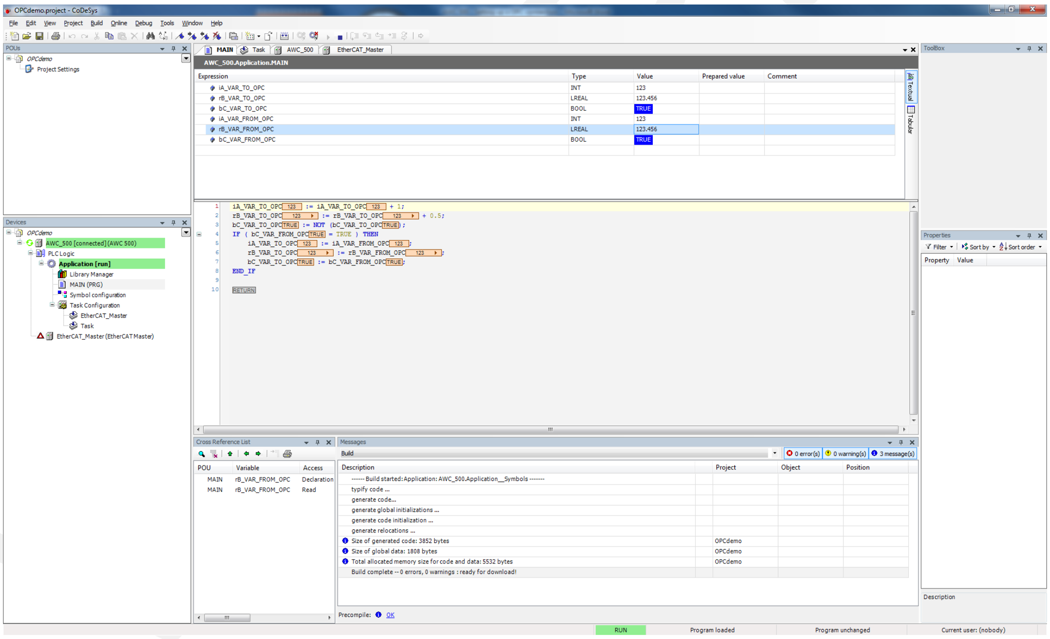
Description	Project	Object	Position
------ Build started: Application: AWC_500.Application__Symbols -------			
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 3852 bytes	OPCdemo		
Size of global data: 1808 bytes	OPCdemo		
Total allocated memory size for code and data: 5532 bytes	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

Precompile: OK

RUN Program loaded Program unchanged Current user: (nobody)

Figure 3.58: The item becomes set in the PLC

Now set item bC_VAR_FROM_OPC 1



AWC_500.Application.MAIN

Expression	Type	Value	Prepared value	Comment
iA_VAR_TO_OPC	INT	123		
rB_VAR_TO_OPC	LREAL	123.456		
bC_VAR_TO_OPC	BOOL	TRUE		
iA_VAR_FROM_OPC	INT	123		
rB_VAR_FROM_OPC	LREAL	123.456		
bC_VAR_FROM_OPC	BOOL	TRUE		

```
iA_VAR_TO_OPC 123 := iA_VAR_TO_OPC 123 + 1;
rB_VAR_TO_OPC 123 := rB_VAR_TO_OPC 123 + 0.5;
bC_VAR_TO_OPC TRUE := NOT (bC_VAR_TO_OPC TRUE);
IF ( bC_VAR_FROM_OPC TRUE = TRUE ) THEN
    iA_VAR_TO_OPC 123 := iA_VAR_FROM_OPC 123;
    rB_VAR_TO_OPC 123 := rB_VAR_FROM_OPC 123;
    bC_VAR_TO_OPC TRUE := bC_VAR_FROM_OPC TRUE;
END_IF

RETURN
```

Cross Reference List

POU	Variable	Access
MAIN	rB_VAR_FROM_OPC	Declaration
MAIN	rB_VAR_FROM_OPC	Read

Messages — Build — 0 error(s) 0 warning(s) 3 message(s)

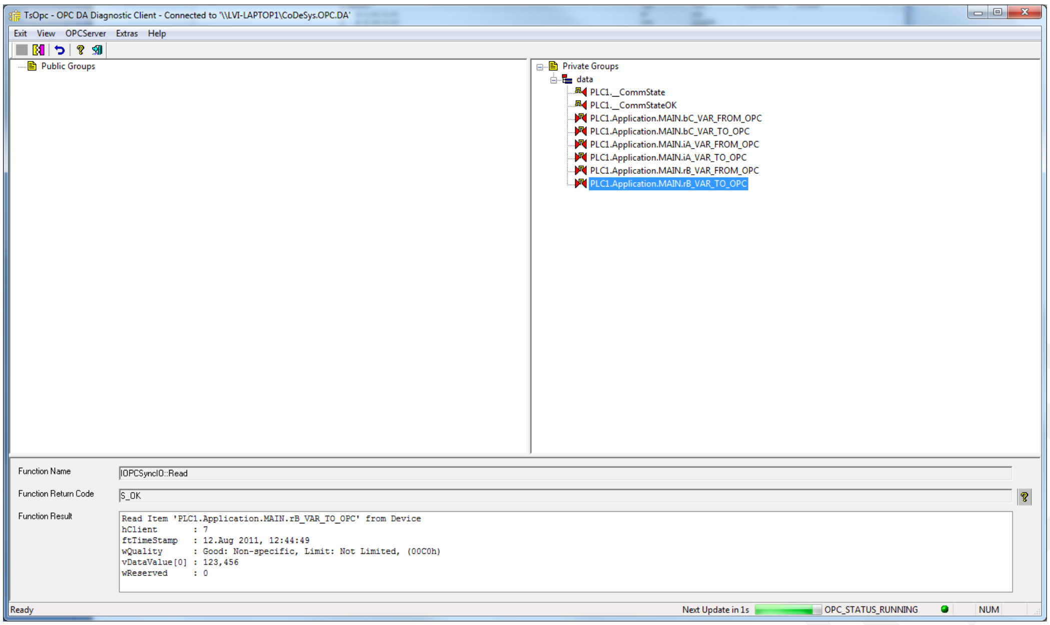
Description	Project	Object	Position
------ Build started: Application: AWC_500.Application__Symbols -------			
typify code ...			
generate code...			
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 3852 bytes	OPCdemo		
Size of global data: 1808 bytes	OPCdemo		
Total allocated memory size for code and data: 5532 bytes	OPCdemo		
Build complete -- 0 errors, 0 warnings : ready for download!			

Precompile: OK

RUN Program loaded Program unchanged Current user: (nobody)

Figure 3.59: bC_VAR_FROM_OPC

and a read of rB_VAR_TO_OPC shows the value:


TsOpc - OPC DA Diagnostic Client - Connected to '\\LVI-LAPTOP1\CoDeSys.OPC.DA'
Exit
View
OPCServer
Extras
Help
Public Groups
Private Groups
data
PLC1._CommState
PLC1._CommStateOK
PLC1.Application.MAIN.bC_VAR_FROM_OPC
PLC1.Application.MAIN.bC_VAR_TO_OPC
PLC1.Application.MAIN.iA_VAR_FROM_OPC
PLC1.Application.MAIN.iA_VAR_TO_OPC
PLC1.Application.MAIN.rB_VAR_FROM_OPC
PLC1.Application.MAIN.rB_VAR_TO_OPC
Ready
Next Update in 1s
OPC_STATUS_RUNNING
NUM

Function Name: IOPCSyncIO::Read

Function Return Code: S_OK

Function Result:

```
Read Item 'PLC1.Application.MAIN.rB_VAR_TO_OPC' from Device
hClient       : 7
ftTimeStamp   : 12.Aug 2011, 12:44:49
wQuality      : Good: Non-specific, Limit: Not Limited, (00C0h)
vDataValue[0] : 123,456
wReserved     : 0
```

Figure 3.60: reading rB_VAR_TO_OPC

Stop/Uninstalling the OPC Server

With command
WinCoDeSysOPC /UnRegServer
all entries of the OPC Server will be removed from the registry.

3.9 Creating a MODBUS TCPIP slave interface

Here we show how to create a MODBUS TCP/IP Slave interface on the AWC 500 .
The steps are:

- Add the Ethernet device to the AWC 500 PCM 51 device
- Add the Modbus Slave Device
- Configure the Modbus slave variables for exchange with the Modbus master

3.9.1 Add the Ethernet device to the AWC 500 PCM 51 device ("→Add Device. . . ")

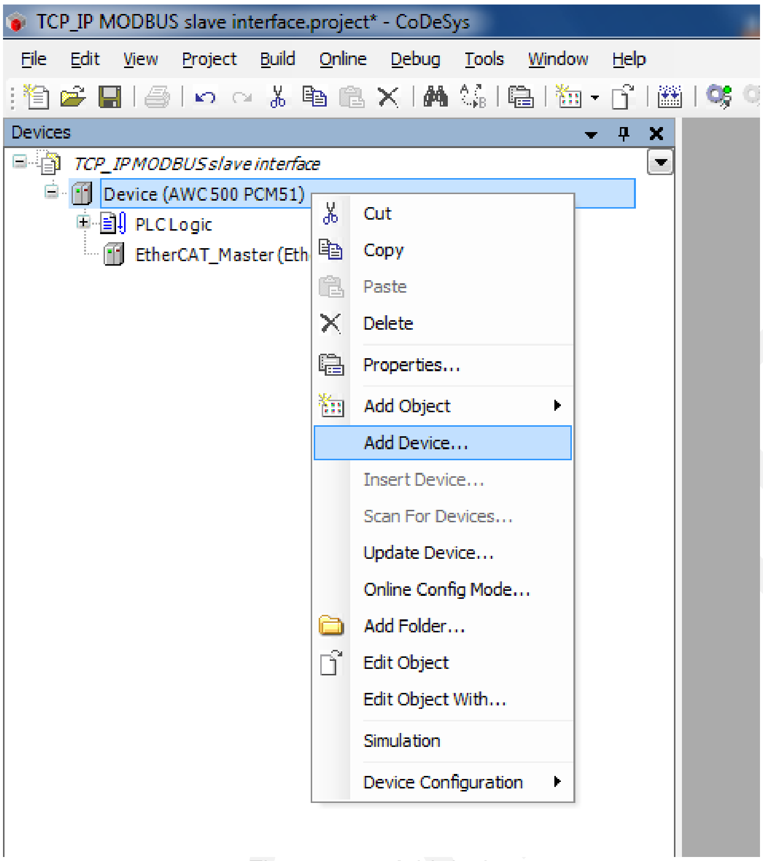
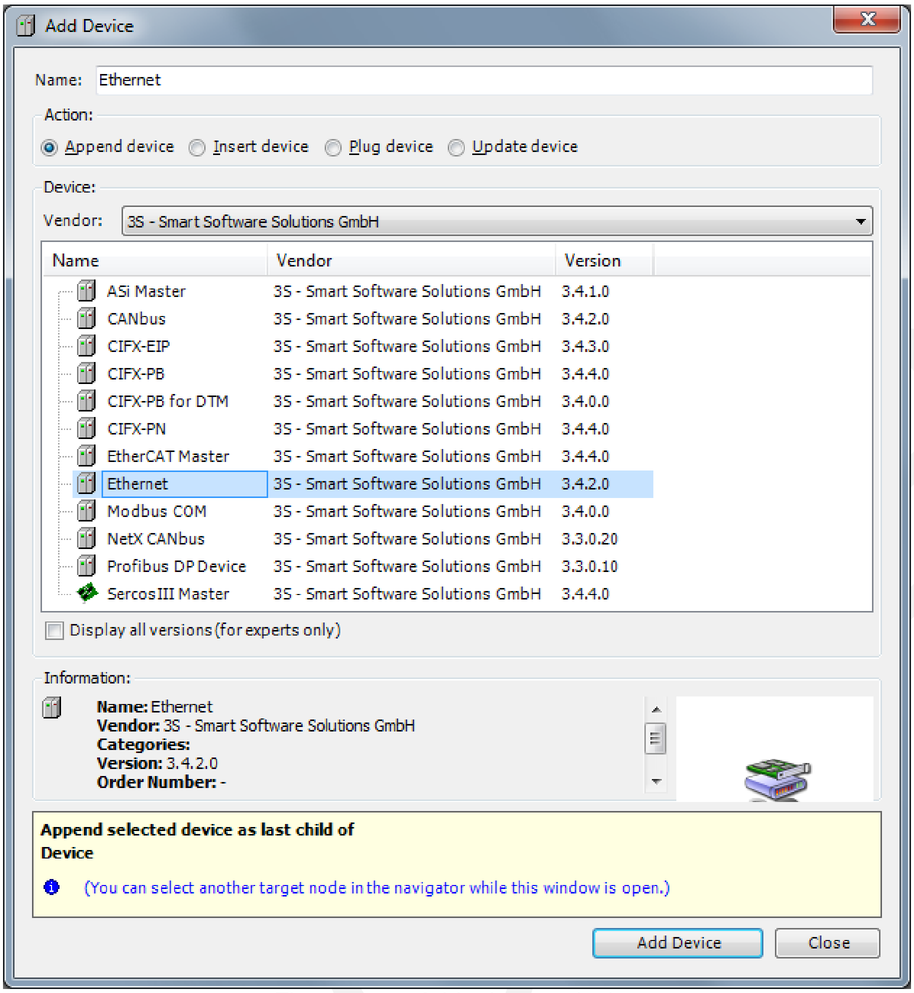


Figure 3.61: Add device

Change vendor to 3S – Smart Software Solutions GmbH and select the Ethernet device:



Add Device

Name: Ethernet

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
ASi Master	3S - Smart Software Solutions GmbH	3.4.1.0
CANbus	3S - Smart Software Solutions GmbH	3.4.2.0
CIFX-EIP	3S - Smart Software Solutions GmbH	3.4.3.0
CIFX-PB	3S - Smart Software Solutions GmbH	3.4.4.0
CIFX-PB for DTM	3S - Smart Software Solutions GmbH	3.4.0.0
CIFX-PN	3S - Smart Software Solutions GmbH	3.4.4.0
EtherCAT Master	3S - Smart Software Solutions GmbH	3.4.4.0
Ethernet	3S - Smart Software Solutions GmbH	3.4.2.0
Modbus COM	3S - Smart Software Solutions GmbH	3.4.0.0
NetX CANbus	3S - Smart Software Solutions GmbH	3.3.0.20
Profibus DP Device	3S - Smart Software Solutions GmbH	3.3.0.10
SercosIII Master	3S - Smart Software Solutions GmbH	3.4.4.0

□ Display all versions (for experts only)

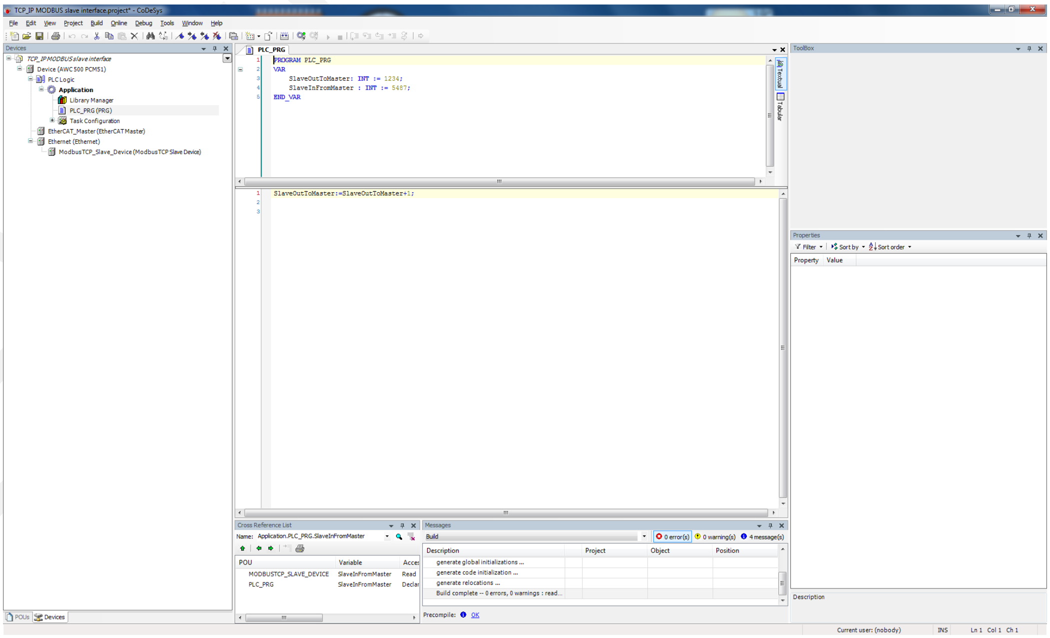
Information:

Name: Ethernet
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.2.0
Order Number: -

Append selected device as last child of
Device

(You can select another target node in the navigator while this window is open.)

Figure 3.62: Add ethernet device



TCP_IP MODBUS slave interface.project* - CoDeSys

TCP_IP MODBUS slave interface
Device (AWC 500 PCM51)
PLC Logic
Application
Library Manager
PLC_PRG (PRG)
Task Configuration
EtherCAT_Master (EtherCAT Master)
Ethernet (Ethernet)
ModbusTCP_Slave_Device (ModbusTCP Slave Device)

PLC_PRG

```
PROGRAM PLC_PRG
VAR
    SlaveOutToMaster: INT := 1234;
    SlaveInFromMaster : INT := 5487;
END_VAR
```

```
SlaveOutToMaster:=SlaveOutToMaster+1;
```

Cross Reference List

Name: Application.PLC_PRG.SlaveInFromMaster

POU	Variable	Acces
MODBUSTCP_SLAVE_DEVICE	SlaveInFromMaster	Read
PLC_PRG	SlaveInFromMaster	Declar

Messages

Build 0 error(s) 0 warning(s) 4 message(s)

Description	Project	Object	Position
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Build complete -- 0 errors, 0 warnings : read...			

Precompile: OK

Current user: (nobody) INS Ln 1 Col 1 Ch 1

Figure 3.63: Ethernet device added

Add the Modbus Slave (Add Device. . . →Modbus TCP Slave Device)

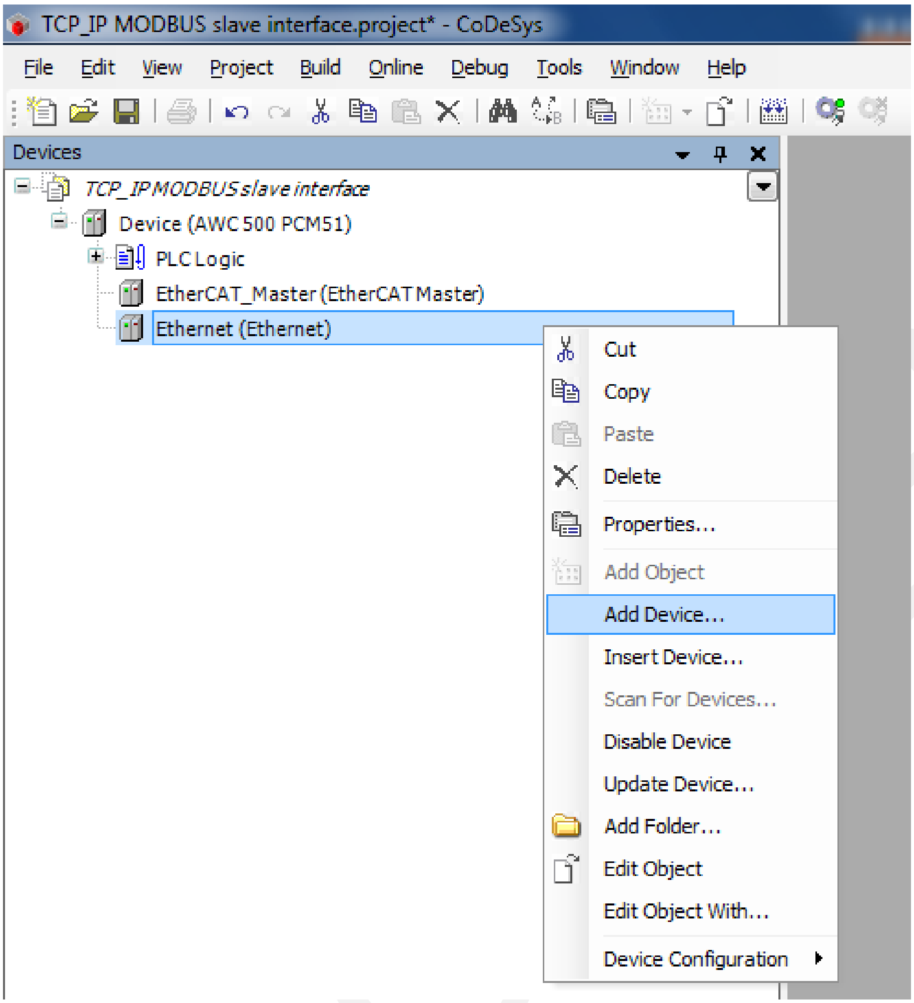
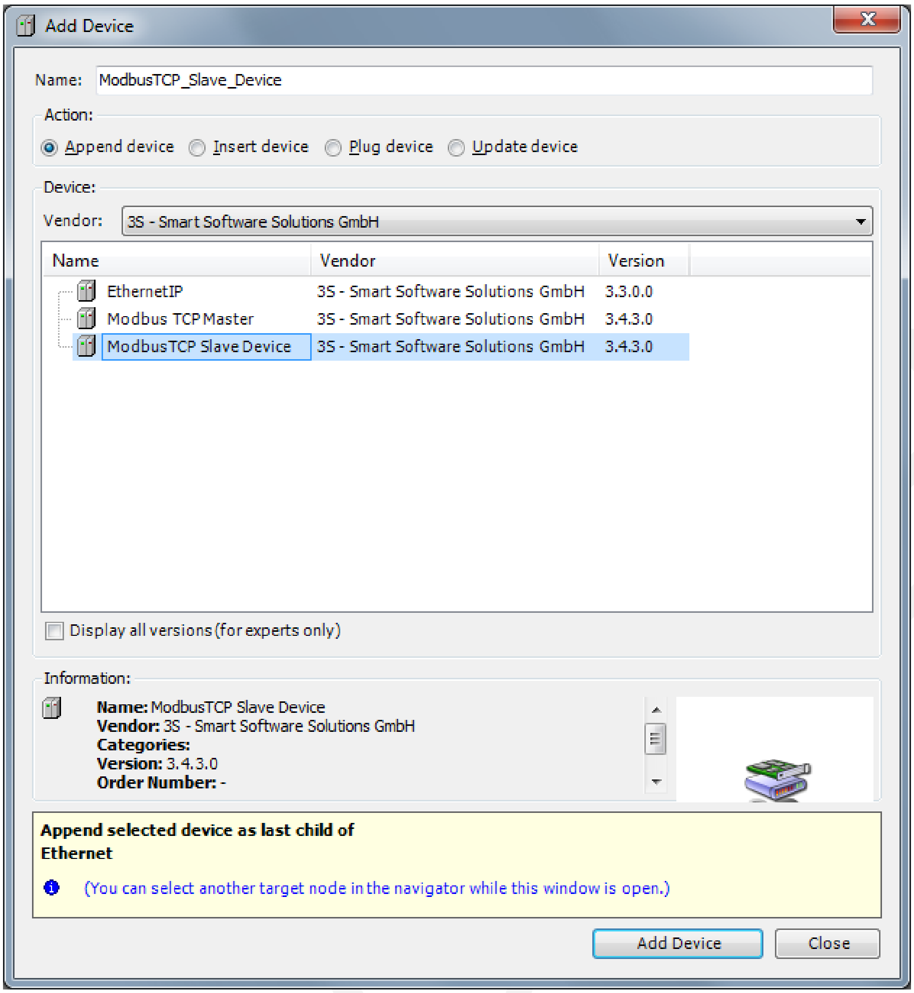


Figure 3.64: Add the Modbus Slave device

From the list select "MODBUS TCP Slave Device":



Add Device

Name: ModbusTCP_Slave_Device

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
EthernetIP	3S - Smart Software Solutions GmbH	3.3.0.0
Modbus TCP Master	3S - Smart Software Solutions GmbH	3.4.3.0
ModbusTCP Slave Device	3S - Smart Software Solutions GmbH	3.4.3.0

□ Display all versions (for experts only)

Information:

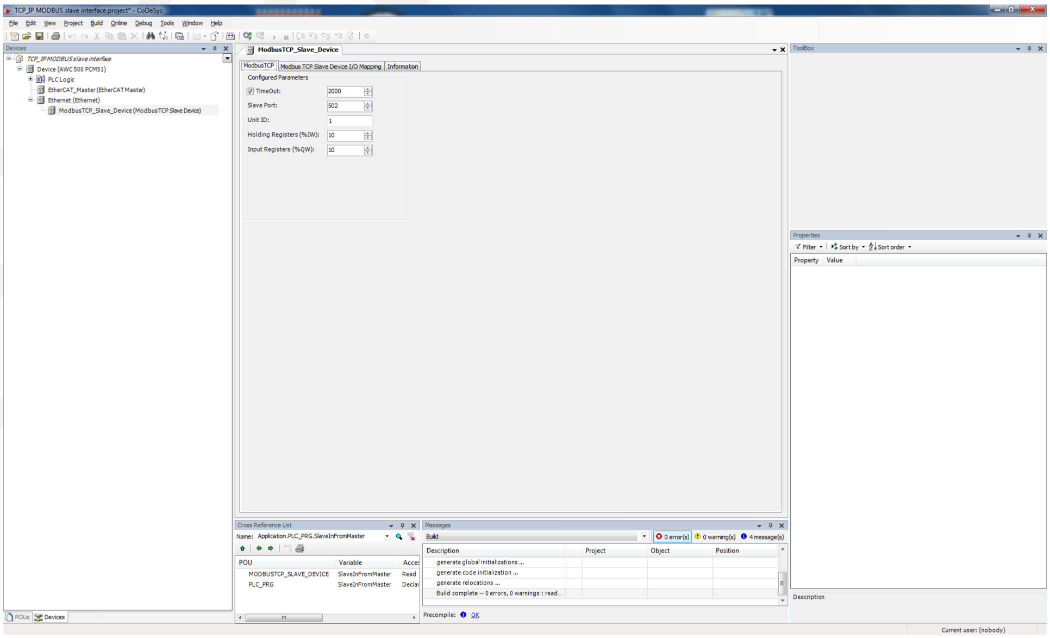
Name: ModbusTCP Slave Device
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.3.0
Order Number: -

Append selected device as last child of
Ethernet

(You can select another target node in the navigator while this window is open.)

Figure 3.65: Select "MODBUS TCP Slave Device"

Configure the overall Response time and socket timeout (optional just leave default). The Unit Id is used to identify the AWC 500 as modbus slave, and shall be used in the configuration of the Modbus master:



TCP_IP MODBUS slave interface.project* - CoDeSys

- TCP_IP MODBUS slave interface
  - Device (AWC 500 PCM51)
    - PLC Logic
    - EtherCAT_Master (EtherCAT Master)
    - Ethernet (Ethernet)
      - ModbusTCP_Slave_Device (ModbusTCP Slave Device)

ModbusTCP_Slave_Device

ModbusTCP | Modbus TCP Slave Device I/O Mapping | Information

Configured Parameters

☑ TimeOut: 2000
Slave Port: 502
Unit ID: 1
Holding Registers (%IW): 10
Input Registers (%QW): 10

Cross Reference List

Name: Application.PLC_PRG.SlaveInFromMaster

POU	Variable	Acces
MODBUSTCP_SLAVE_DEVICE	SlaveInFromMaster	Read
PLC_PRG	SlaveInFromMaster	Declar

Messages

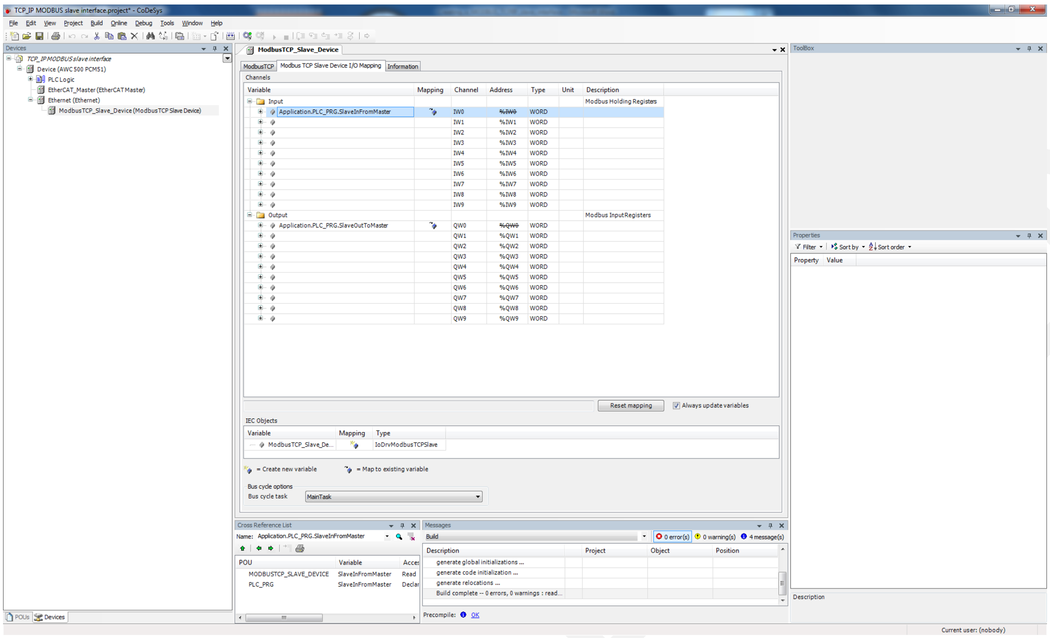
Build — 0 error(s) 0 warning(s) 4 message(s)

Description		Project	Object	Position
generate global initializations ...				
generate code initialization ...				
generate relocations ...				
Build complete -- 0 errors, 0 warnings : read...				

Precompile: OK

Figure 3.66: Modbus TCP setup

The settings above create:
10 MODBUS Holding registers from offset 0, to be written with MODBUS Function 06 or 16. They can be read with Function 03 (4x).
10 MODBUS Input registers from offset 0. They can be read by MODBUS Function 04 (3x)
I/O Mapping:



ModbusTCP_Slave_Device

ModbusTCP | Modbus TCP Slave Device I/O Mapping | Information

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
Input						Modbus Holding Registers
Application.PLC_PRG.SlaveInFromMaster		IW0	~~%IW0~~	WORD		
		IW1	%IW1	WORD		
		IW2	%IW2	WORD		
		IW3	%IW3	WORD		
		IW4	%IW4	WORD		
		IW5	%IW5	WORD		
		IW6	%IW6	WORD		
		IW7	%IW7	WORD		
		IW8	%IW8	WORD		
		IW9	%IW9	WORD		
Output						Modbus InputRegisters
Application.PLC_PRG.SlaveOutToMaster		QW0	~~%QW0~~	WORD		
		QW1	%QW1	WORD		
		QW2	%QW2	WORD		
		QW3	%QW3	WORD		
		QW4	%QW4	WORD		
		QW5	%QW5	WORD		
		QW6	%QW6	WORD		
		QW7	%QW7	WORD		
		QW8	%QW8	WORD		
		QW9	%QW9	WORD		

Reset mapping ☑ Always update variables

IEC Objects

Variable	Mapping	Type
ModbusTCP_Slave_De...		IoDrvModbusTCPSlave

= Create new variable = Map to existing variable

Bus cycle options
Bus cycle task: MainTask

Cross Reference List
Name: Application.PLC_PRG.SlaveInFromMaster

POU	Variable	Acces
MODBUSTCP_SLAVE_DEVICE	SlaveInFromMaster	Read
PLC_PRG	SlaveInFromMaster	Declar

Messages
Build — 0 error(s) 0 warning(s) 4 message(s)

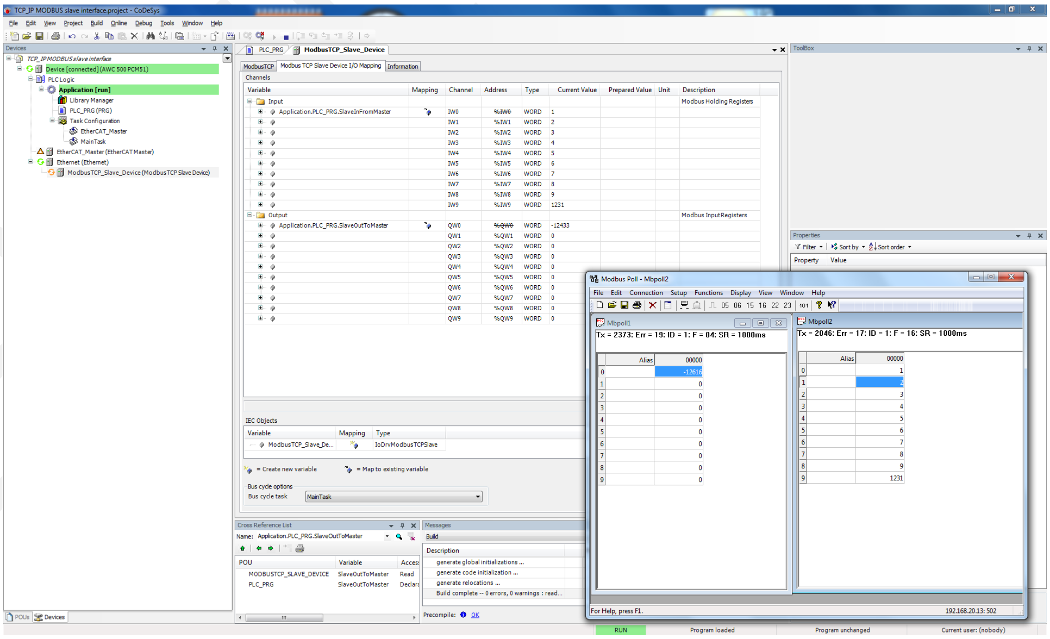
```
generate global initializations ...
generate code initialization ...
generate relocations ...
Build complete -- 0 errors, 0 warnings : read...
```

Precompile: OK

Figure 3.67: I/O Mapping

Set "Always update variables" if the input variable is not actually used in the program. Specify Bus cycle task to control what task, should be responsible to execute the MODBUS slave.

Use a modbus master to test the interface e.g. Modbus Poll:



PLC_PRG | ModbusTCP_Slave_Device

ModbusTCP | Modbus TCP Slave Device I/O Mapping | Information

Channels

Variable	Mapping	Channel	Address	Type	Current Value	Prepared Value	Unit	Description
Input								Modbus Holding Registers
Application.PLC_PRG.SlaveInFromMaster		IW0	~~%IW0~~	WORD	1			
		IW1	%IW1	WORD	2			
		IW2	%IW2	WORD	3			
		IW3	%IW3	WORD	4			
		IW4	%IW4	WORD	5			
		IW5	%IW5	WORD	6			
		IW6	%IW6	WORD	7			
		IW7	%IW7	WORD	8			
		IW8	%IW8	WORD	9			
		IW9	%IW9	WORD	1231			
Output								Modbus InputRegisters
Application.PLC_PRG.SlaveOutToMaster		QW0	~~%QW0~~	WORD	-12433			
		QW1	%QW1	WORD	0			
		QW2	%QW2	WORD	0			
		QW3	%QW3	WORD	0			
		QW4	%QW4	WORD	0			
		QW5	%QW5	WORD	0			
		QW6	%QW6	WORD	0			
		QW7	%QW7	WORD	0			
		QW8	%QW8	WORD	0			
		QW9	%QW9	WORD	0			

Modbus Poll - Mbpoll2

Mbpoll1: Tx = 2373: Err = 19: ID = 1: F = 04: SR = 1000ms

	Alias	00000
0		-12616
1		0
2		0
3		0
4		0
5		0
6		0
7		0
8		0
9		0

Mbpoll2: Tx = 2046: Err = 17: ID = 1: F = 16: SR = 1000ms

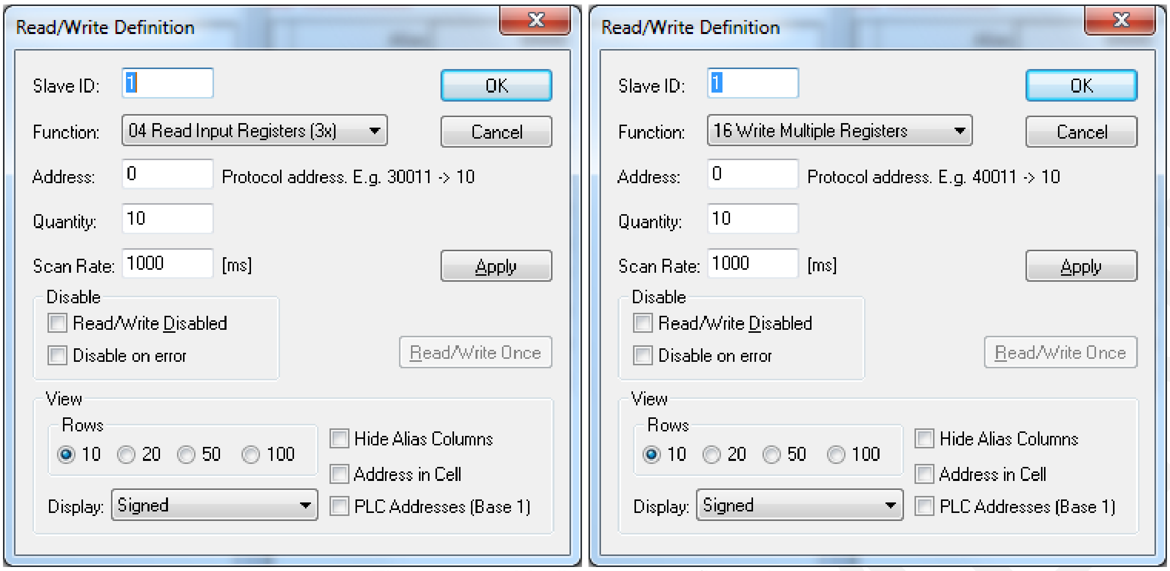
	Alias	00000
0		1
1		2
2		3
3		4
4		5
5		6
6		7
7		8
8		9
9		1231

For Help, press F1. 192.168.20.13: 502

RUN Program loaded Program unchanged Current user: (nobody)

Figure 3.68: Modbus Poll for testing modbus interface

The used configurations in Modbus poll are shown here:



Read/Write Definition

Slave ID: 1

Function: 04 Read Input Registers (3x)

Address: 0 Protocol address. E.g. 30011 -> 10

Quantity: 10

Scan Rate: 1000 [ms]

Disable: □ Read/Write Disabled □ Disable on error

View: Rows ● 10 ○ 20 ○ 50 ○ 100

Display: Signed

□ Hide Alias Columns □ Address in Cell □ PLC Addresses (Base 1)

Read/Write Definition

Slave ID: 1

Function: 16 Write Multiple Registers

Address: 0 Protocol address. E.g. 40011 -> 10

Quantity: 10

Scan Rate: 1000 [ms]

Disable: □ Read/Write Disabled □ Disable on error

View: Rows ● 10 ○ 20 ○ 50 ○ 100

Display: Signed

□ Hide Alias Columns □ Address in Cell □ PLC Addresses (Base 1)

Figure 3.69: Modbus poll configuration

3.9.2 Extending the number of registers

By default the Modbus Slave is limited to max 40. To extend the registers edit the device description file: `c:\ProgramData\CoDeSys\Devices\115\0000 0002\3.4.3.0\device.xml`

Locate:

```
<RangeType basetype="std:USINT" name="AssemblySizeRangeType">
   <!— Number IO—Parameters (WORD) —>
   <Min>2</Min>
   <Max>40</Max> Change to e.g. 500.
   <Default>10</Default>
</RangeType>
```

There is a known bug that is limiting the number of registers to 1000.

3.9.3 Diagnostics

The state of the modbus slave can be monitored by accessing the IEC Objects in the PLC program.
In each cycle the server is executed once. In case of a socket error the reset flag is set, normal request processing then should continue in the next cycle:

```
m_ModbusServer();                    // execute request processing
IF(m_ModbusServer.xError)THEN        // any errors?
      m_ModbusServer.xReset := TRUE; // set reset flag
      IoDrvStartBusCycle := Errors.ERR_SOCK_NOTCONNECTED;
ELSE
IoDrvStartBusCycle := Errors.ERR_OK;
END_IF
```

3.9.4 Demo

See further the demo project "TCP_IP Modbus master loopback.project"

3.9.5 Troubleshooting

If you are unable to connect from the Master to the Modbus slave, try ping the AWC 500 .

3.10 Setting up a MODBUS TCPIP master

To create a MODBUS TCP/IP master the steps are :

- Add the Ethernet device to the AWC 500 PCM 51 device
- Add the Modbus master
- Add the Modbus slave to the master
- Configure the Modbus slave variables for exchange with AWC 500

3.10.1 Add the Ethernet device to the AWC 500 PCM 51 device ("→Add Device. . . ")

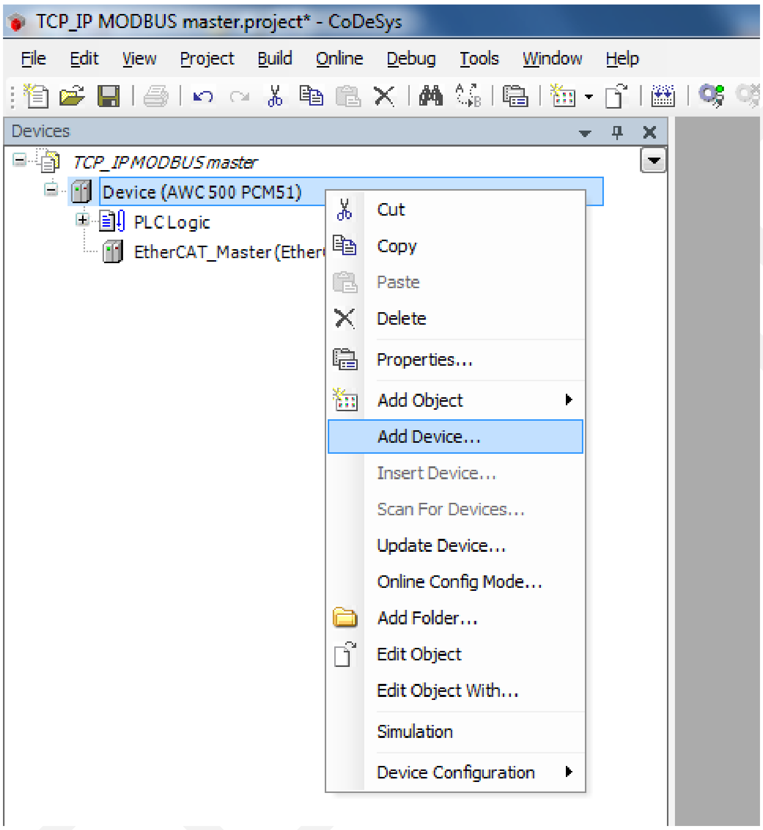
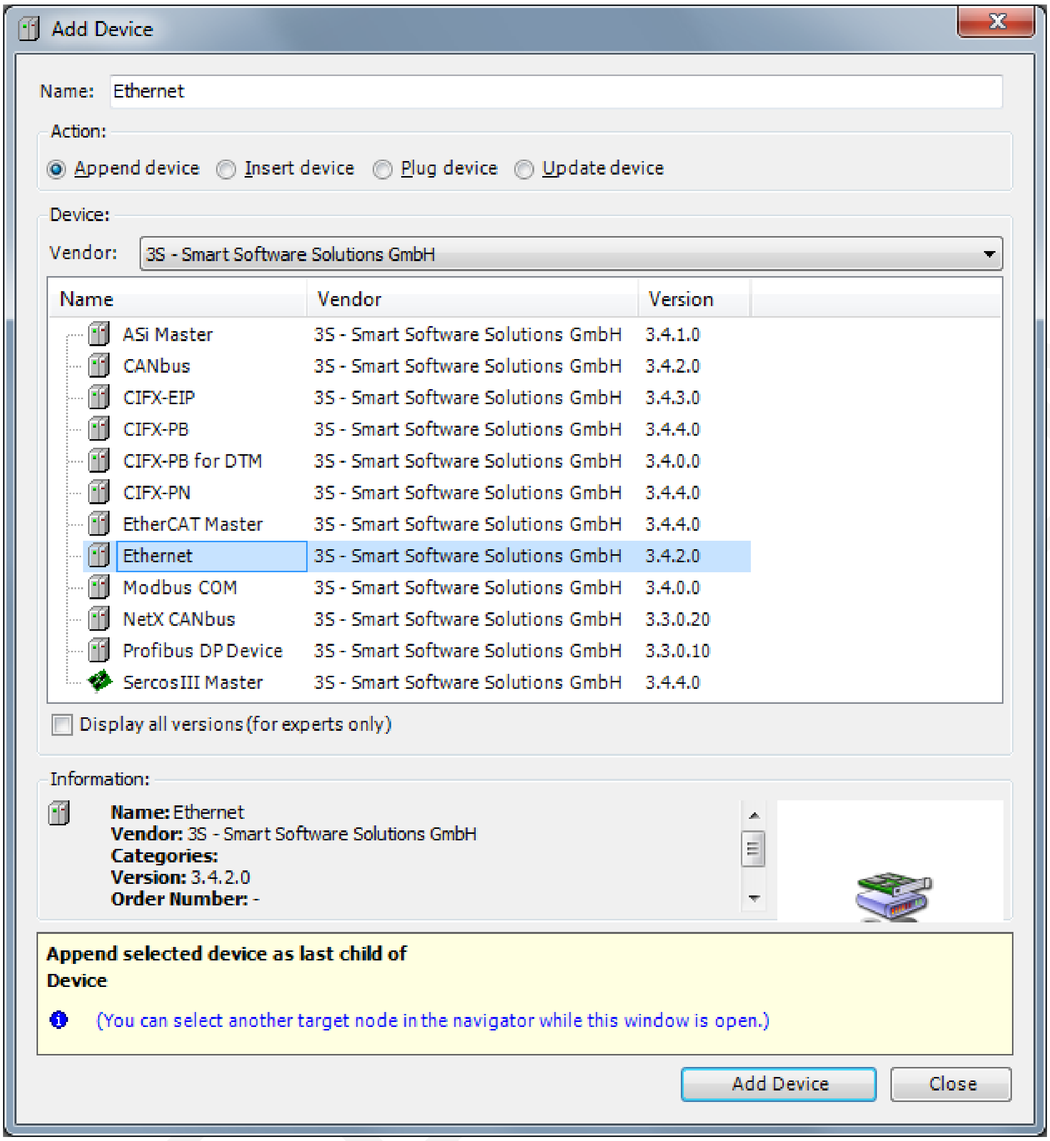


Figure 3.70: Add device

Change vendor to 3S – Smart Software Solutions GmbH and select the Ethernet device:



Add Device

Name: Ethernet

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
ASi Master	3S - Smart Software Solutions GmbH	3.4.1.0
CANbus	3S - Smart Software Solutions GmbH	3.4.2.0
CIFX-EIP	3S - Smart Software Solutions GmbH	3.4.3.0
CIFX-PB	3S - Smart Software Solutions GmbH	3.4.4.0
CIFX-PB for DTM	3S - Smart Software Solutions GmbH	3.4.0.0
CIFX-PN	3S - Smart Software Solutions GmbH	3.4.4.0
EtherCAT Master	3S - Smart Software Solutions GmbH	3.4.4.0
Ethernet	3S - Smart Software Solutions GmbH	3.4.2.0
Modbus COM	3S - Smart Software Solutions GmbH	3.4.0.0
NetX CANbus	3S - Smart Software Solutions GmbH	3.3.0.20
Profibus DP Device	3S - Smart Software Solutions GmbH	3.3.0.10
SercosIII Master	3S - Smart Software Solutions GmbH	3.4.4.0

□ Display all versions (for experts only)

Information:

Name: Ethernet
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.2.0
Order Number: -

Append selected device as last child of
Device

(You can select another target node in the navigator while this window is open.)

Figure 3.71: Add ethernet device

3.10.2 Add the Modbus master (Add Device. . . →Modbus TCP Master)

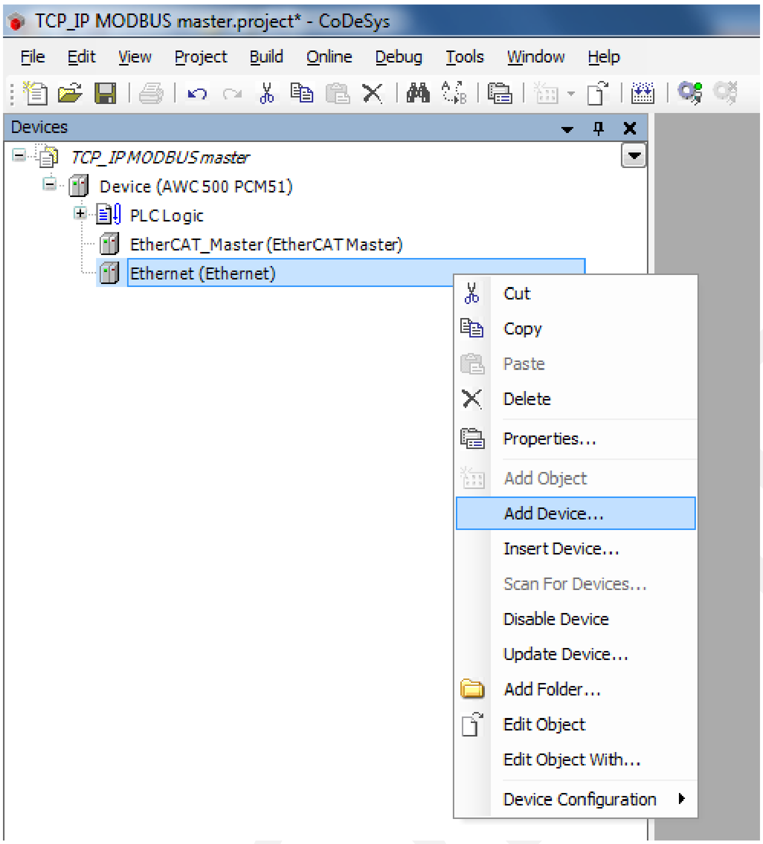
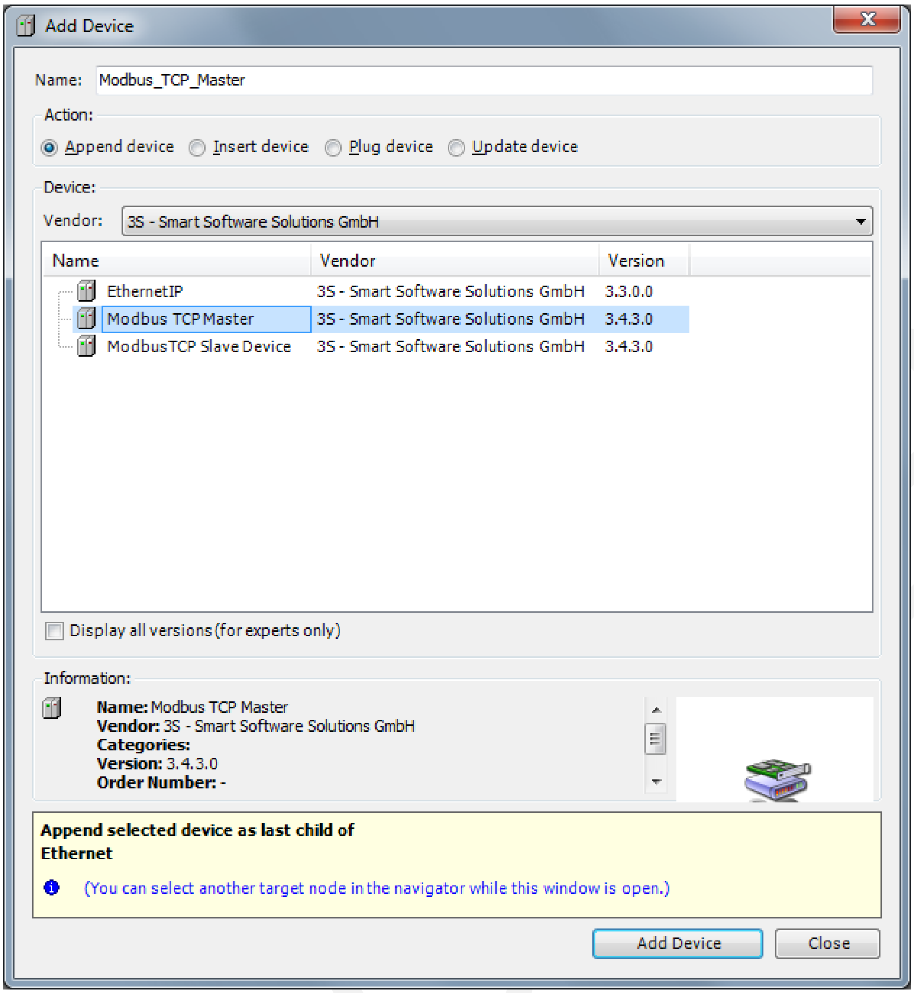


Figure 3.72: Add the Modbus master

From the list select "MODBUS TCP Master":



Add Device

Name: Modbus_TCP_Master

Action:

● Append device ○ Insert device ○ Plug device ○ Update device

Device:

Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
EthernetIP	3S - Smart Software Solutions GmbH	3.3.0.0
Modbus TCP Master	3S - Smart Software Solutions GmbH	3.4.3.0
ModbusTCP Slave Device	3S - Smart Software Solutions GmbH	3.4.3.0

□ Display all versions (for experts only)

Information:

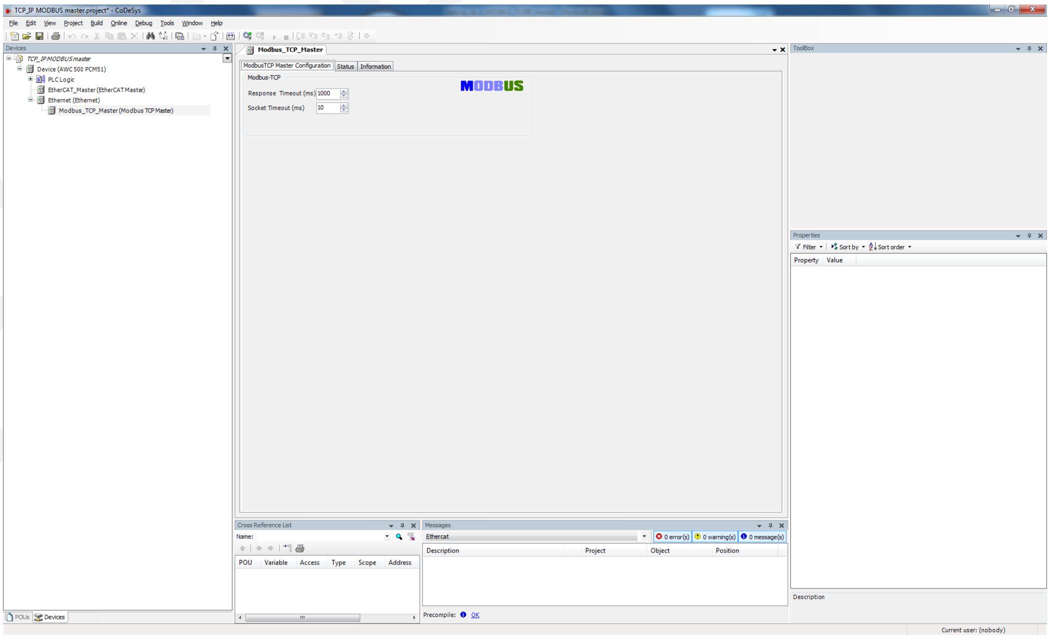
Name: Modbus TCP Master
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.3.0
Order Number: -

Append selected device as last child of
Ethernet

(You can select another target node in the navigator while this window is open.)

Figure 3.73: Select "MODBUS TCP Master"

Configure the overall Respone time and socket timeout (optional just leave default):



TCP_IP MODBUS master.project* - CoDeSys

Response Timeout (ms) 1000

Socket Timeout (ms) 10

Figure 3.74: Configure the overall Respone time

3.10.3 Add the Modbus TCP slave(Add Device. . . →Modbus TCP slave)

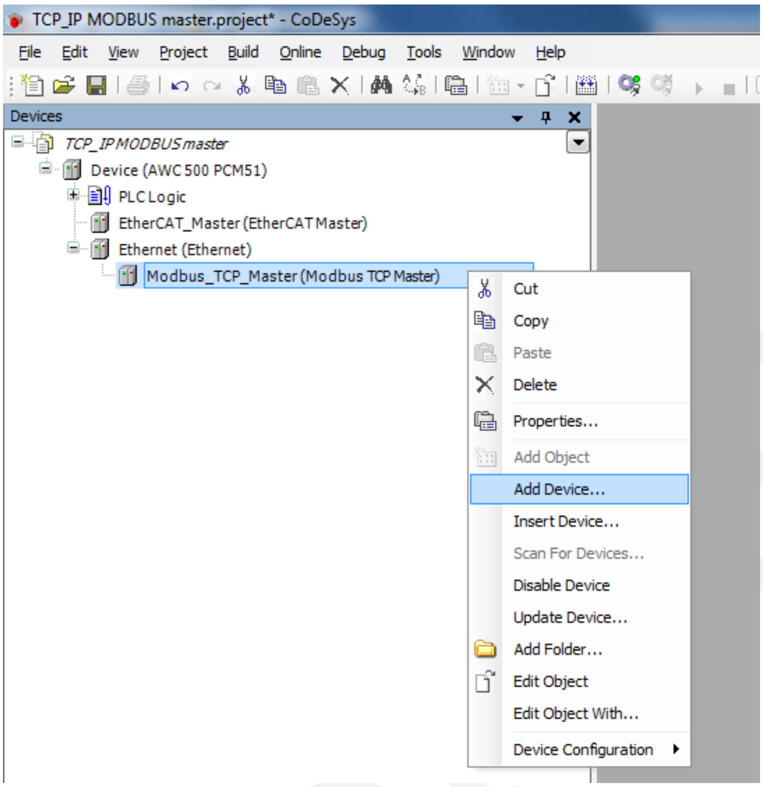
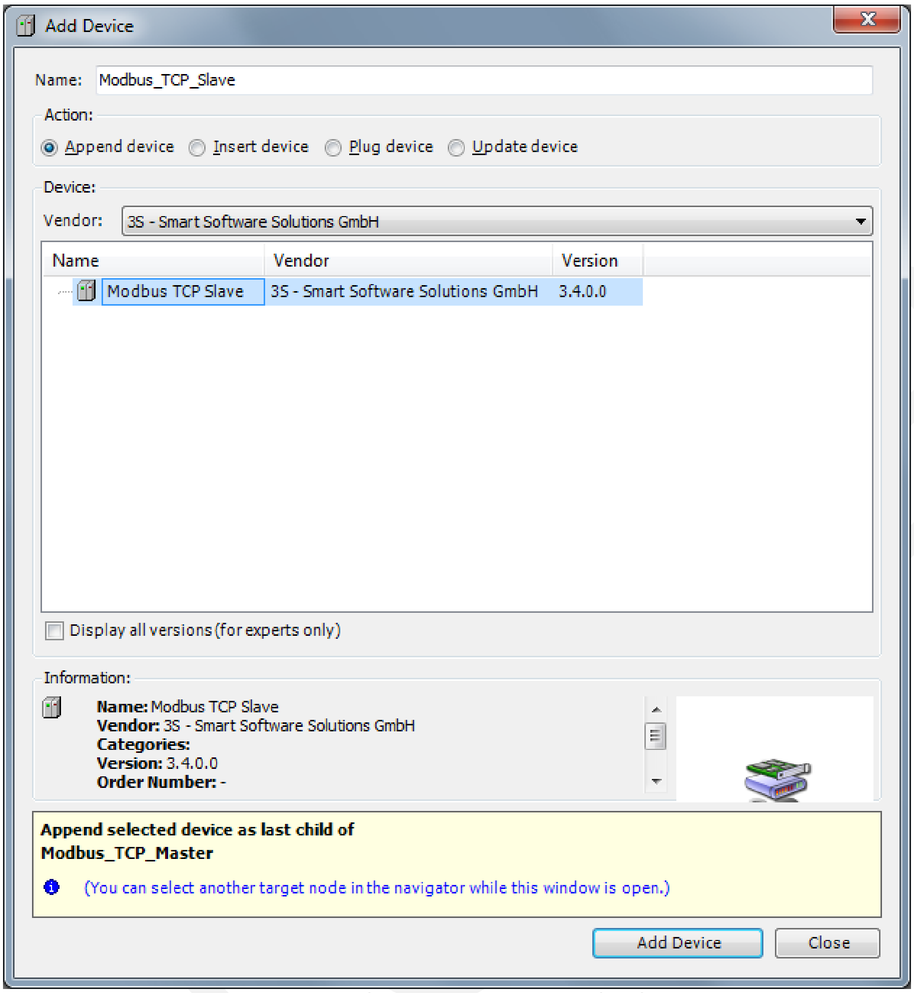


Figure 3.75: Add the Modbus TCP slave

Select the Modbus TCP Slave from the list and "Add Device":



Add Device

Name: Modbus_TCP_Slave

Action:
● Append device ○ Insert device ○ Plug device ○ Update device

Device:
Vendor: 3S - Smart Software Solutions GmbH

Name	Vendor	Version
Modbus TCP Slave	3S - Smart Software Solutions GmbH	3.4.0.0

□ Display all versions (for experts only)

Information:

Name: Modbus TCP Slave
Vendor: 3S - Smart Software Solutions GmbH
Categories:
Version: 3.4.0.0
Order Number: -

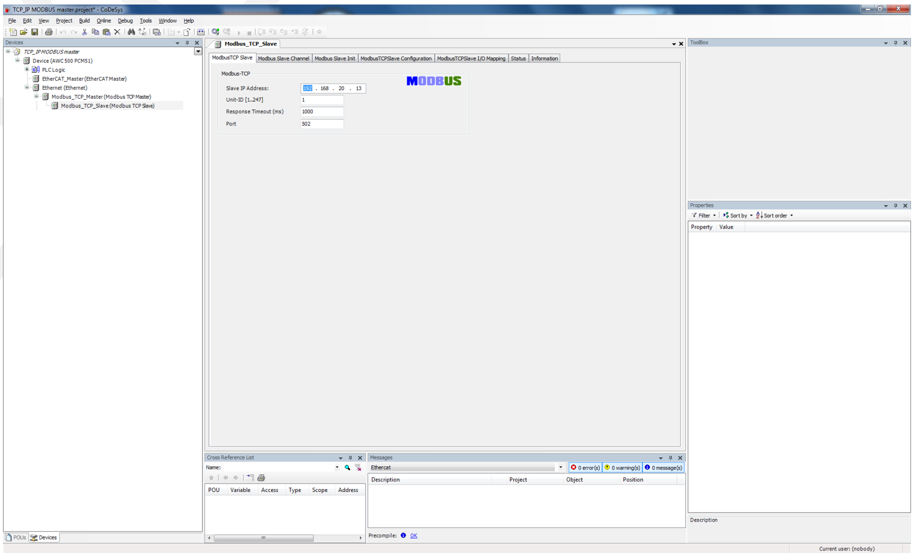
Append selected device as last child of
Modbus_TCP_Master

(You can select another target node in the navigator while this window is open.)

Figure 3.76: Select the Modbus TCP Slave

3.10.4 Configure the Modbus slave variables for exchange with AWC 500

Slave : First specify the IP address of the Modbus slave (here the local IP address is shown) and unit id:



TCP_IP MODBUS master.project* - CoDeSys

TCP_IP MODBUS master
- Device (AWC 500 PCM51)
  - PLC Logic
  - EtherCAT_Master (EtherCAT Master)
  - Ethernet (Ethernet)
    - Modbus_TCP_Master (Modbus TCP Master)
      - Modbus_TCP_Slave (Modbus TCP Slave)

Modbus_TCP_Slave

ModbusTCP Slave | Modbus Slave Channel | Modbus Slave Init | ModbusTCPSlave Configuration | ModbusTCPSlave I/O Mapping | Status | Information

Modbus-TCP

Slave IP Address: 192 . 168 . 20 . 13
Unit-ID [1..247]: 1
Response Timeout (ms): 1000
Port: 502

Figure 3.77: Specify the IP address of the Modbus slave

Channel: On the Modbus Slave Channel page a list of modbus function calls (channels) are add to do the exchange of variables:

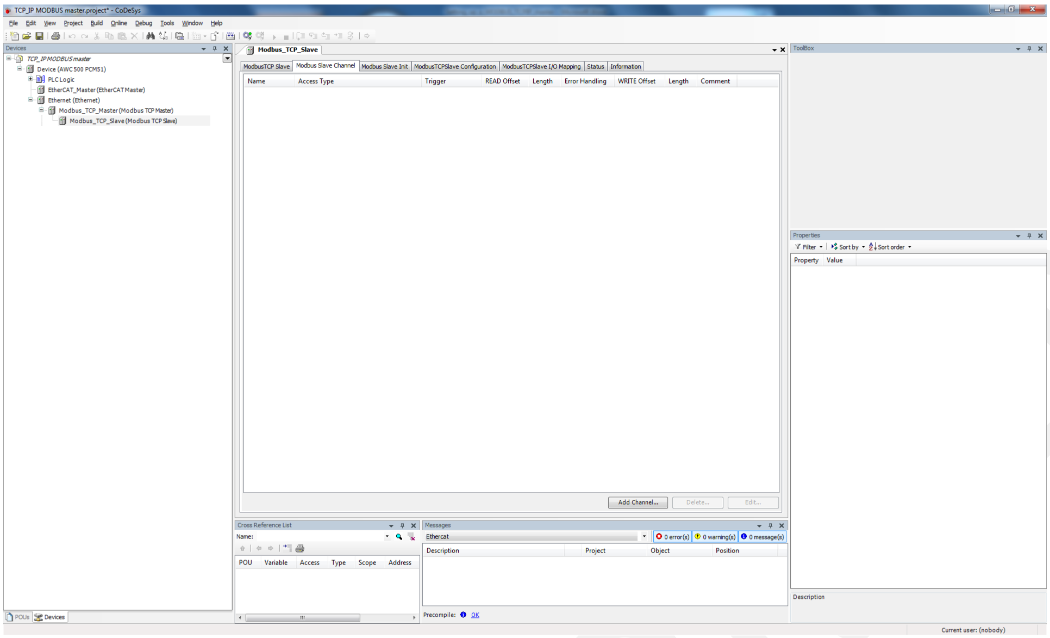
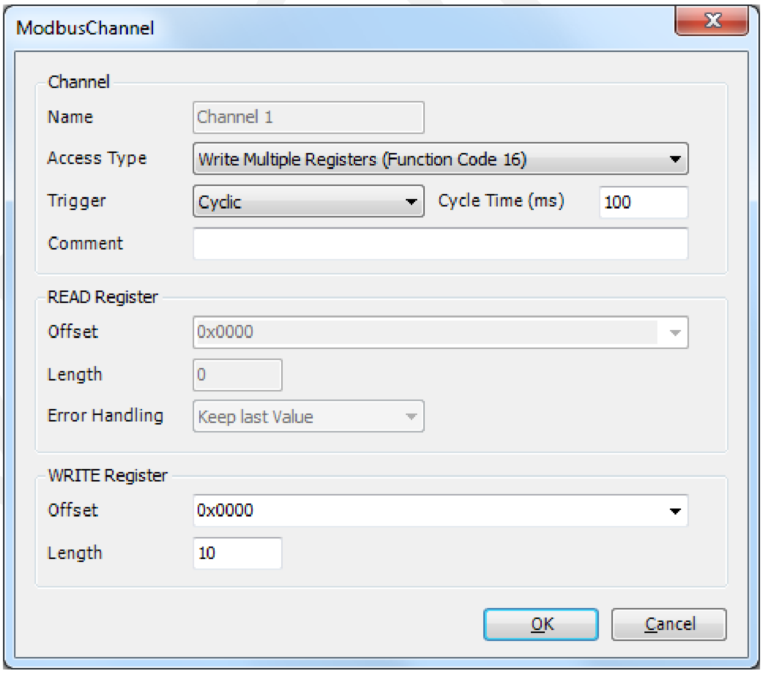


Figure 3.78: Modbus slave channels

Add a channel by selecting "Add Channel. . . ", these settings shall match the provided modbus features of the Modbus slave eg. here are 10 variables written from the AWC 500 to 10 holding registers in the modbus slave:



ModbusChannel

Channel

Name: Channel 1

Access Type: Write Multiple Registers (Function Code 16)

Trigger: Cyclic | Cycle Time (ms): 100

Comment:

READ Register

Offset: 0x0000

Length: 0

Error Handling: Keep last Value

WRITE Register

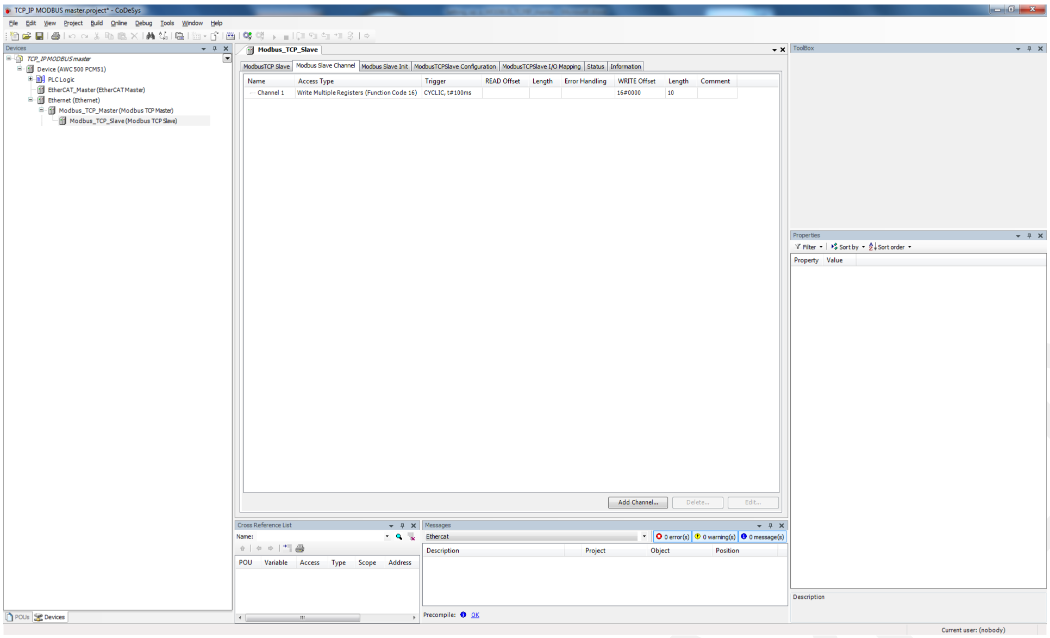
Offset: 0x0000

Length: 10

Figure 3.79: Configure Modbus channels

Changing the trigger from Cyclic to RISING_EDGE, creates a boolean variable that can be toggled from the program to control exchange of variables.

Here after the channel is added to the list:



TCP_IP MODBUS master.project* - CoDeSys

Devices

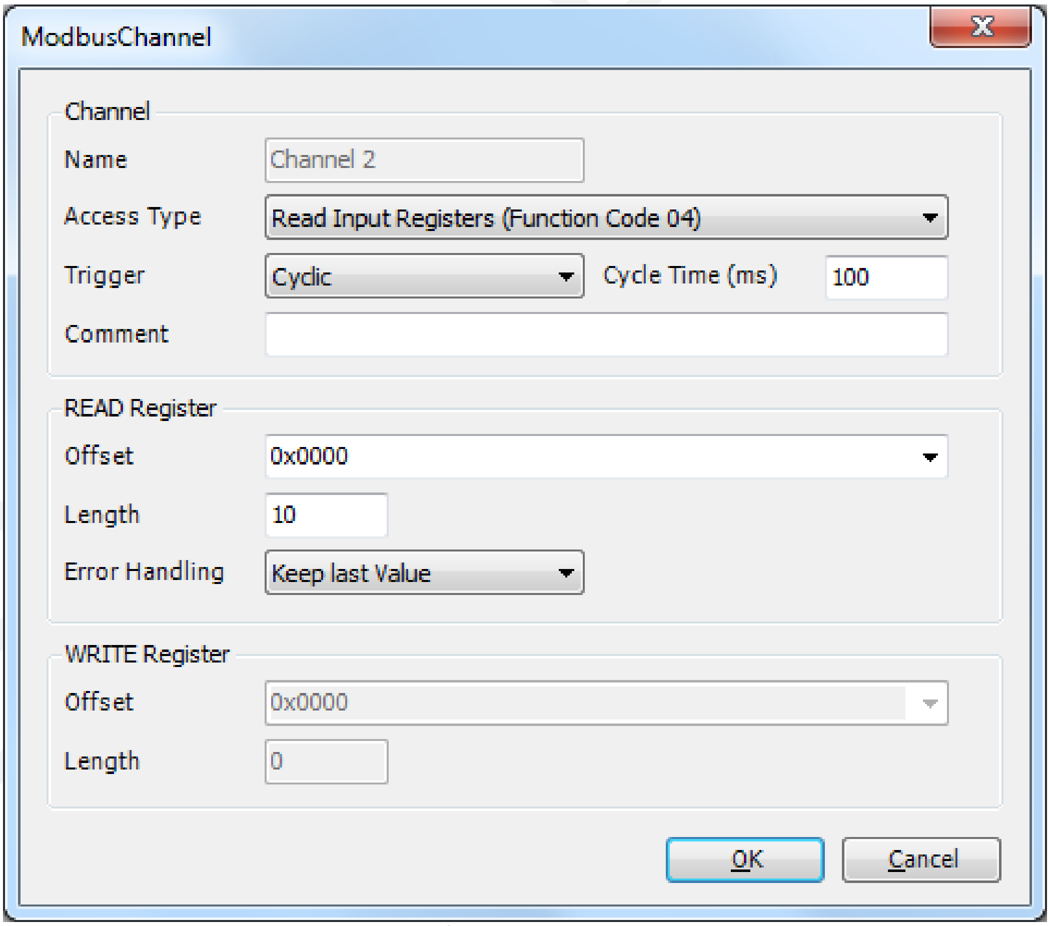
```
TCP_IP MODBUS master
  Device (AWC 500 PCM51)
    PLC Logic
    EtherCAT_Master (EtherCAT Master)
    Ethernet (Ethernet)
      Modbus_TCP_Master (Modbus TCP Master)
        Modbus_TCP_Slave (Modbus TCP Slave)
```

Modbus_TCP_Slave — Modbus Slave Channel

Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset	Length	Comment
Channel 1	Write Multiple Registers (Function Code 16)	CYCLIC, t#100ms				16#0000	10	

Figure 3.80: Channel added to the list

Add additional channels to eg read input registers or holding registers from the Modbus slave (eg. here are 10 (length) registers read, starting by offset 0x0000:



ModbusChannel

Channel

Name: Channel 2

Access Type: Read Input Registers (Function Code 04)

Trigger: Cyclic

Cycle Time (ms): 100

Comment:

READ Register

Offset: 0x0000

Length: 10

Error Handling: Keep last Value

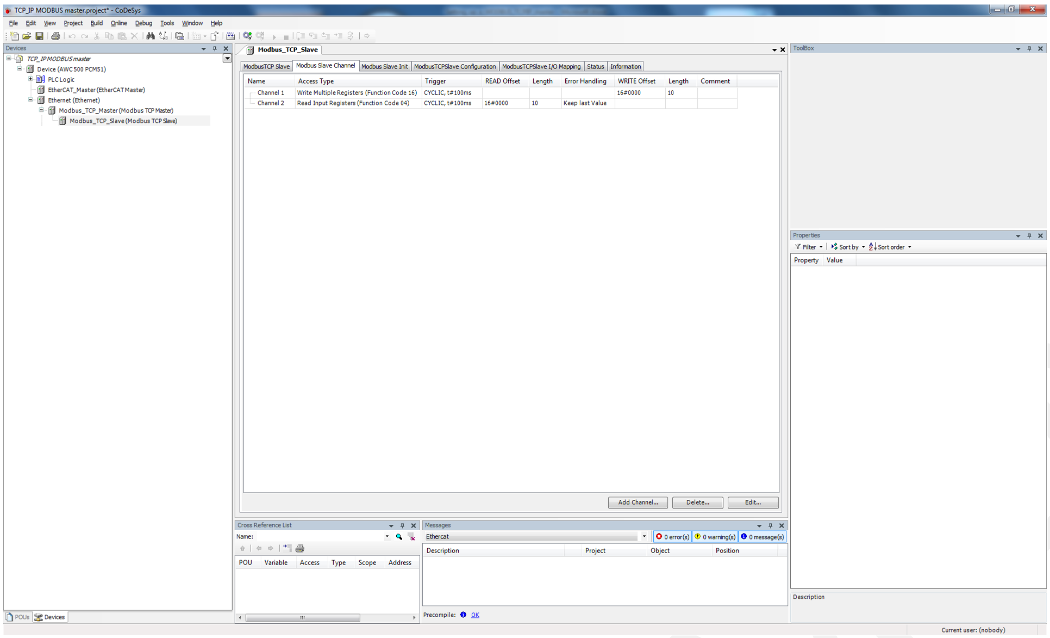
WRITE Register

Offset: 0x0000

Length: 0

Figure 3.81: Configure Modbus channels

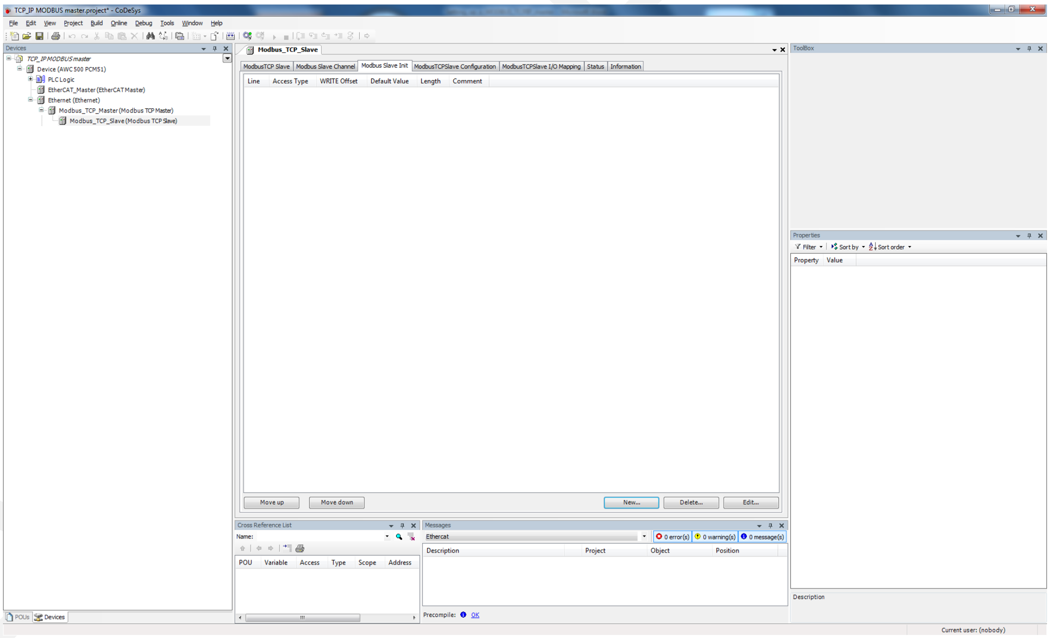
Same way the function is added to the list:



Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset	Length	Comment
Channel 1	Write Multiple Registers (Function Code 16)	CYCLIC, t#100ms				16#0000	10	
Channel 2	Read Input Registers (Function Code 04)	CYCLIC, t#100ms	16#0000	10	Keep last Value			

Figure 3.82: Channel added to the list

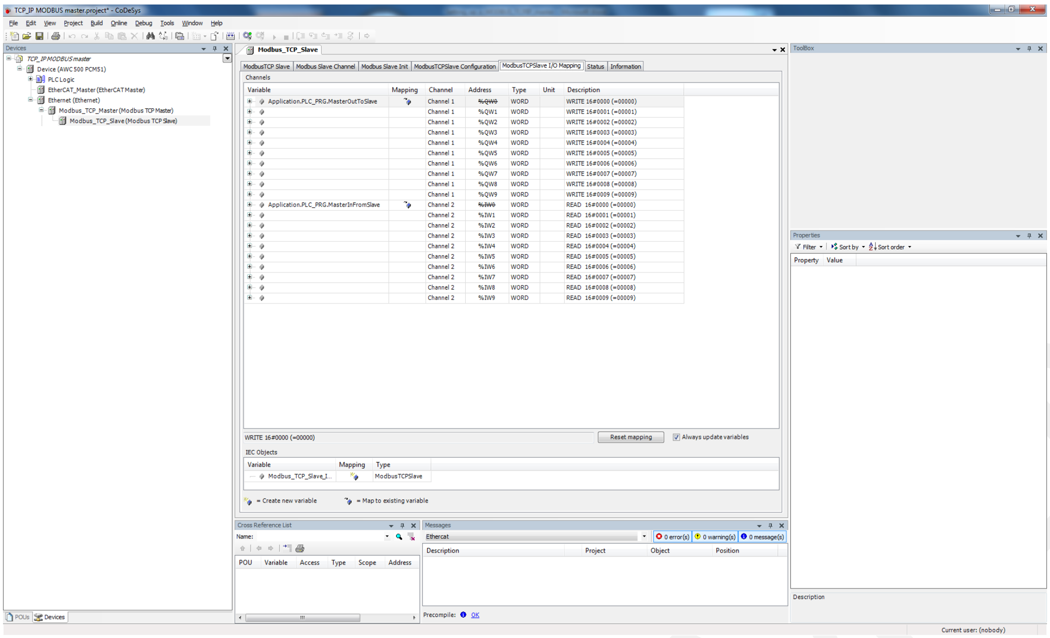
Init: Here modbus functions can be added to the list, to set registers in the Modbus slave on power up. Add them to the list using "New. . ." if required.



Line	Access Type	WRITE Offset	Default Value	Length	Comment

Figure 3.83: Modbus TCP Slave init tab

I/O Mapping: Here the PLC variables for excahnge can be linked to the modbus function(channel):


TCP_IP MODBUS master.project* - CoDeSys

Devices
- TCP_IP MODBUS master
  - Device (AWC 500 PCM51)
    - PLC Logic
    - EtherCAT_Master (EtherCAT Master)
    - Ethernet (Ethernet)
      - Modbus_TCP_Master (Modbus TCP Master)
        - Modbus_TCP_Slave (Modbus TCP Slave)

Modbus_TCP_Slave

ModbusTCP Slave | Modbus Slave Channel | Modbus Slave Init | ModbusTCPSlave Configuration | ModbusTCPSlave I/O Mapping | Status | Information

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
Application.PLC_PRG.MasterOutToSlave		Channel 1	~~%QW0~~	WORD		WRITE 16#0000 (=00000)
		Channel 1	%QW1	WORD		WRITE 16#0001 (=00001)
		Channel 1	%QW2	WORD		WRITE 16#0002 (=00002)
		Channel 1	%QW3	WORD		WRITE 16#0003 (=00003)
		Channel 1	%QW4	WORD		WRITE 16#0004 (=00004)
		Channel 1	%QW5	WORD		WRITE 16#0005 (=00005)
		Channel 1	%QW6	WORD		WRITE 16#0006 (=00006)
		Channel 1	%QW7	WORD		WRITE 16#0007 (=00007)
		Channel 1	%QW8	WORD		WRITE 16#0008 (=00008)
		Channel 1	%QW9	WORD		WRITE 16#0009 (=00009)
Application.PLC_PRG.MasterInFromSlave		Channel 2	~~%IW0~~	WORD		READ 16#0000 (=00000)
		Channel 2	%IW1	WORD		READ 16#0001 (=00001)
		Channel 2	%IW2	WORD		READ 16#0002 (=00002)
		Channel 2	%IW3	WORD		READ 16#0003 (=00003)
		Channel 2	%IW4	WORD		READ 16#0004 (=00004)
		Channel 2	%IW5	WORD		READ 16#0005 (=00005)
		Channel 2	%IW6	WORD		READ 16#0006 (=00006)
		Channel 2	%IW7	WORD		READ 16#0007 (=00007)
		Channel 2	%IW8	WORD		READ 16#0008 (=00008)
		Channel 2	%IW9	WORD		READ 16#0009 (=00009)

WRITE 16#0000 (=00000)

☑ Always update variables

IEC Objects

Variable	Mapping	Type
Modbus_TCP_Slave_I...		ModbusTCPSlave

= Create new variable   = Map to existing variable

Messages: Ethercat — 0 error(s) | 0 warning(s) | 0 message(s)

Precompile: OK

Current user: (nobody)

Figure 3.84: Linking PLC Variables to the modbus function

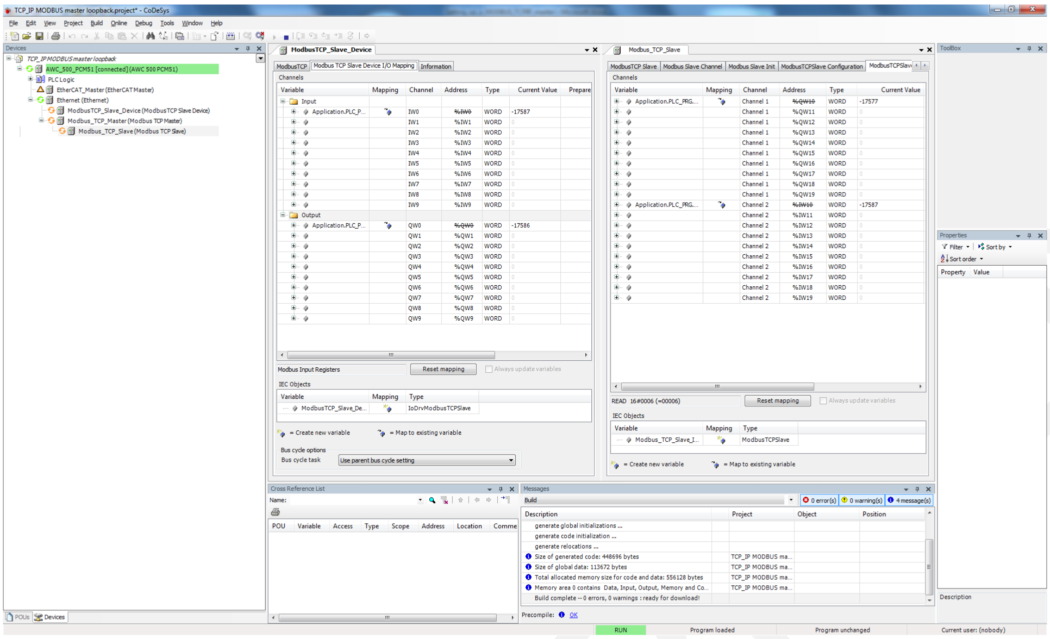
3.10.5 Diagnostics

The state of the modbus master and each slave can be monitored by accessing the IEC Objects in the PLC program.

```
VAR
    MasterError  :  BOOL;
    MasterErrorOnSlave  :  BOOL;
END_VAR;
MasterError  :=  Modbus_TCP_Master_Instance.xSlaveError;  (*TRUE if error on the modbus master*)
MasterErrorOnSlave  :=  Modbus_TCP_Slave_Instance.xError;  (*TRUE if error on the particular slave
```

3.10.6 Demo

See further the demo project \TCPIP MODBUS Master\TCP_IP_Modbus_master_loopback.project



ModbusTCP_Slave_Device — Channels

Variable	Mapping	Channel	Address	Type	Current Value	Prepare
Input						
Application.PLC_P...		IW0	~~%IW0~~	WORD	-17587	
		IW1	%IW1	WORD	0	
		IW2	%IW2	WORD	0	
		IW3	%IW3	WORD	0	
		IW4	%IW4	WORD	0	
		IW5	%IW5	WORD	0	
		IW6	%IW6	WORD	0	
		IW7	%IW7	WORD	0	
		IW8	%IW8	WORD	0	
		IW9	%IW9	WORD	0	
Output						
Application.PLC_P...		QW0	~~%QW0~~	WORD	-17586	
		QW1	%QW1	WORD	0	
		QW2	%QW2	WORD	0	
		QW3	%QW3	WORD	0	
		QW4	%QW4	WORD	0	
		QW5	%QW5	WORD	0	
		QW6	%QW6	WORD	0	
		QW7	%QW7	WORD	0	
		QW8	%QW8	WORD	0	
		QW9	%QW9	WORD	0	

Modbus Input Registers

Reset mapping

□ Always update variables

IEC Objects

Variable	Mapping	Type
ModbusTCP_Slave_De...		IoDrvModbusTCPSlave

= Create new variable = Map to existing variable

Bus cycle options

Bus cycle task: Use parent bus cycle setting

Modbus_TCP_Slave — Channels

Variable	Mapping	Channel	Address	Type	Current Value
Application.PLC_PRG....		Channel 1	~~%QW10~~	WORD	-17577
		Channel 1	%QW11	WORD	0
		Channel 1	%QW12	WORD	0
		Channel 1	%QW13	WORD	0
		Channel 1	%QW14	WORD	0
		Channel 1	%QW15	WORD	0
		Channel 1	%QW16	WORD	0
		Channel 1	%QW17	WORD	0
		Channel 1	%QW18	WORD	0
		Channel 1	%QW19	WORD	0
Application.PLC_PRG....		Channel 2	~~%IW10~~	WORD	-17587
		Channel 2	%IW11	WORD	0
		Channel 2	%IW12	WORD	0
		Channel 2	%IW13	WORD	0
		Channel 2	%IW14	WORD	0
		Channel 2	%IW15	WORD	0
		Channel 2	%IW16	WORD	0
		Channel 2	%IW17	WORD	0
		Channel 2	%IW18	WORD	0
		Channel 2	%IW19	WORD	0

READ 16#0006 (=00006)

Reset mapping

□ Always update variables

IEC Objects

Variable	Mapping	Type
Modbus_TCP_Slave_I...		ModbusTCPSlave

= Create new variable = Map to existing variable

Messages — Build

Description	Project	Object	Position
generate global initializations ...			
generate code initialization ...			
generate relocations ...			
Size of generated code: 448696 bytes	TCP_IP MODBUS ma...		
Size of global data: 113672 bytes	TCP_IP MODBUS ma...		
Total allocated memory size for code and data: 556128 bytes	TCP_IP MODBUS ma...		
Memory area 0 contains Data, Input, Output, Memory and Co...	TCP_IP MODBUS ma...		
Build complete -- 0 errors, 0 warnings : ready for download!			

Figure 3.85: Modbus master demo

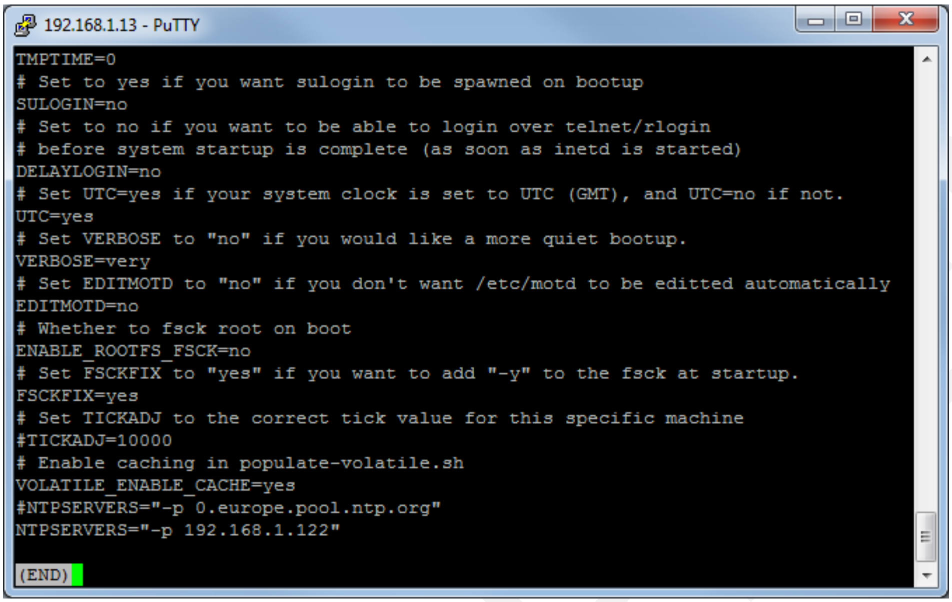
3.10.7 Troubleshooting

Make sure the IP adress range of the Modbus slave fits the address range of the AWC 500 . To check you can use ping [slaveip] via SHH connection to test.

3.11 Setting up NTP (Network Time Protocol) client

The AWC 500 supports NTP (Network Time protocol). This is a client that can access a NTP Server to request time.

In the file `/etc/default/rcS`:



```
TMPTIME=0
# Set to yes if you want sulogin to be spawned on bootup
SULOGIN=no
# Set to no if you want to be able to login over telnet/rlogin
# before system startup is complete (as soon as inetd is started)
DELAYLOGIN=no
# Set UTC=yes if your system clock is set to UTC (GMT), and UTC=no if not.
UTC=yes
# Set VERBOSE to "no" if you would like a more quiet bootup.
VERBOSE=very
# Set EDITMOTD to "no" if you don't want /etc/motd to be editted automatically
EDITMOTD=no
# Whether to fsck root on boot
ENABLE_ROOTFS_FSCK=no
# Set FSCKFIX to "yes" if you want to add "-y" to the fsck at startup.
FSCKFIX=yes
# Set TICKADJ to the correct tick value for this specific machine
#TICKADJ=10000
# Enable caching in populate-volatile.sh
VOLATILE_ENABLE_CACHE=yes
#NTPSERVERS="-p 0.europe.pool.ntp.org"
NTPSERVERS="-p 192.168.1.122"

(END)
```

Figure 3.86: Edit /etc/default/rcS

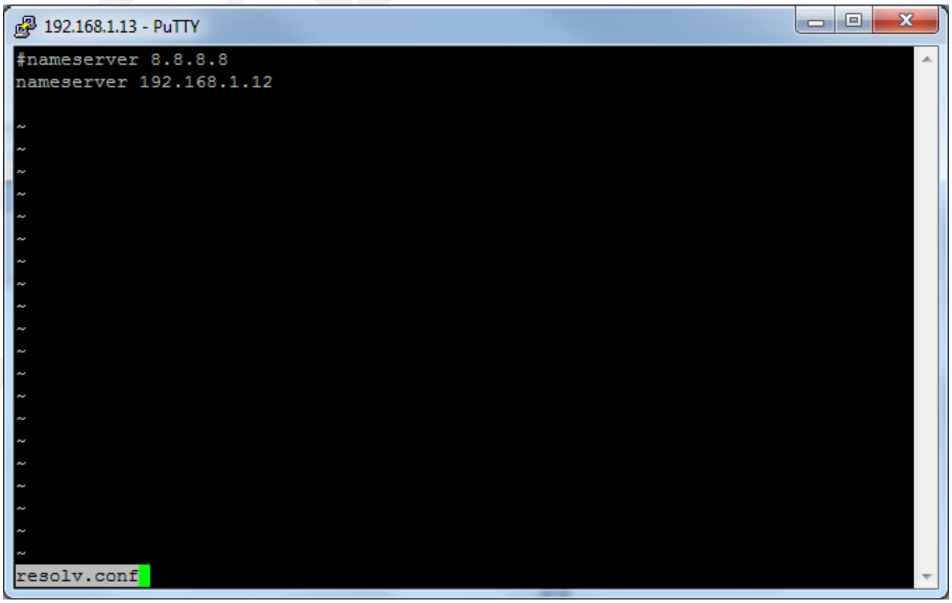
Uncomment the line:

#NTPSERVERS="−p 0.europe.pool.ntp.org"

And type in the ip or url of the local NTP server (in this case 192.168.1.122).

For testing with the internet pool of NTP servers set up DNS in /etc/resolv.conf.

Add the nameserver here:



```
#nameserver 8.8.8.8
nameserver 192.168.1.12

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
resolv.conf
```

Figure 3.87: Set name server in resolv.conf

Now the NPT Client has been configured so that every time the AWC 500 is rebooted it will try to update the system time from the NTP server defined in /etc/default/rcS.

3.11.1 Testing the NTP Client from the command line

To update the system time from the command line type:

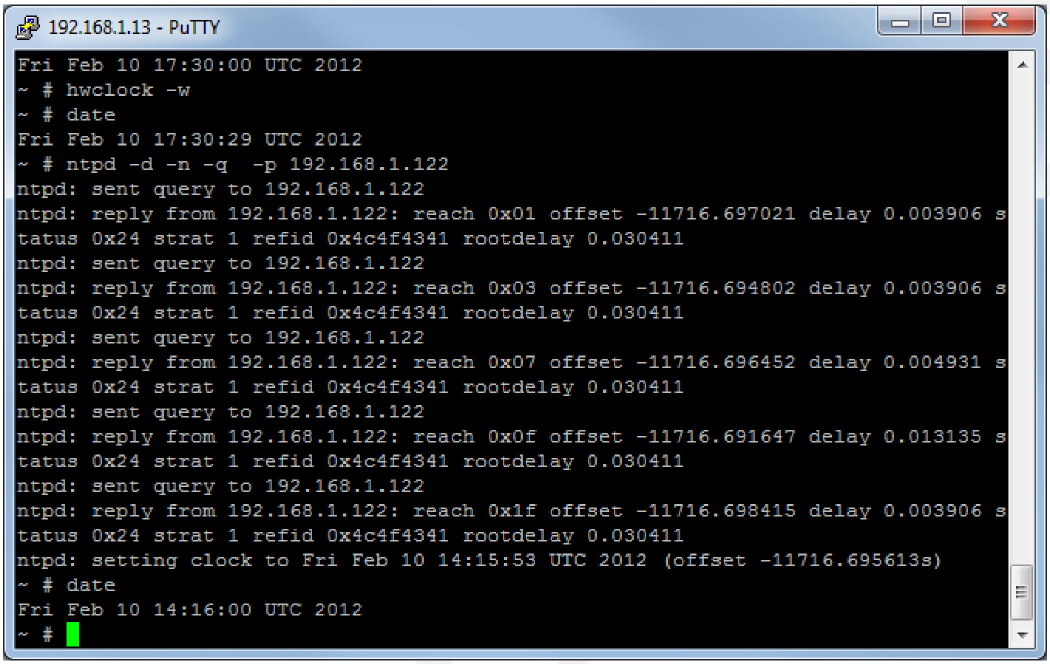
```
ntpd -d -n -q  -p 192.168.1.122
```

(quits after first successful time update)

```
ntpd -d -n -p 192.168.1.122
```

(continues time update)

The command line options present a verbose debug output of the communication with the NTP server.



```
Fri Feb 10 17:30:00 UTC 2012
~ # hwclock -w
~ # date
Fri Feb 10 17:30:29 UTC 2012
~ # ntpd -d -n -q  -p 192.168.1.122
ntpd: sent query to 192.168.1.122
ntpd: reply from 192.168.1.122: reach 0x01 offset -11716.697021 delay 0.003906 s
tatus 0x24 strat 1 refid 0x4c4f4341 rootdelay 0.030411
ntpd: sent query to 192.168.1.122
ntpd: reply from 192.168.1.122: reach 0x03 offset -11716.694802 delay 0.003906 s
tatus 0x24 strat 1 refid 0x4c4f4341 rootdelay 0.030411
ntpd: sent query to 192.168.1.122
ntpd: reply from 192.168.1.122: reach 0x07 offset -11716.696452 delay 0.004931 s
tatus 0x24 strat 1 refid 0x4c4f4341 rootdelay 0.030411
ntpd: sent query to 192.168.1.122
ntpd: reply from 192.168.1.122: reach 0x0f offset -11716.691647 delay 0.013135 s
tatus 0x24 strat 1 refid 0x4c4f4341 rootdelay 0.030411
ntpd: sent query to 192.168.1.122
ntpd: reply from 192.168.1.122: reach 0x1f offset -11716.698415 delay 0.003906 s
tatus 0x24 strat 1 refid 0x4c4f4341 rootdelay 0.030411
ntpd: setting clock to Fri Feb 10 14:15:53 UTC 2012 (offset -11716.695613s)
~ # date
Fri Feb 10 14:16:00 UTC 2012
~ #
```

Figure 3.88: Output from ntpd

To verify that the system time has been updated the date command can be invoked before and after running the ntp client. The above screen shot show how the system time is changed from 17:30 to 14:16 utilising the NTP client.

A typical test could look like this:

1. Set system time(to an incorrect time) using the date command(date [yyyymmddHHMM]):

   ```
   date 201202101730
   ```

2. See the current system time :

   ```
   date
   ```

3. Call NTP Client:

   ```
   ntpd -d -n -q  -p 192.168.1.122
   ```

4. Call date to see that the system time has been reset to the correct time.

   ```
   date
   ```

In case the system time has not been set correctly repeat step 3 and 4.

Another test is to change the system time on the NTP server if possible and the repeat step 2, 3, and 4 to verify that the AWC 500 system time is set accordingly to the NTP server time.

3.11.2 Test NTP server setup (NetTime)

Download the open source NTP server NetTime from Sourceforge
`http://sourceforge.net/projects/nettime/files/latest/download`

Install the NTP server using the default options in the install wizard.

The server is now running as a service.

3.11.3 Configuring NetTime

NetTime will by defualt run as an NTP client and synchronize the host system time with a public NTP server.

To enable the NetTime server, right-click the NetTime tray icon and select Properties.

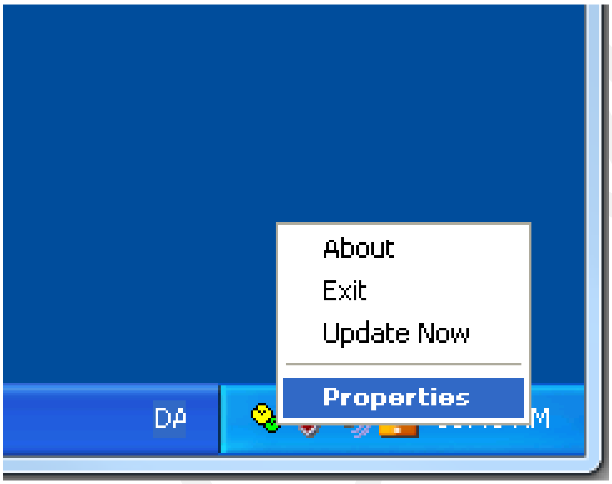
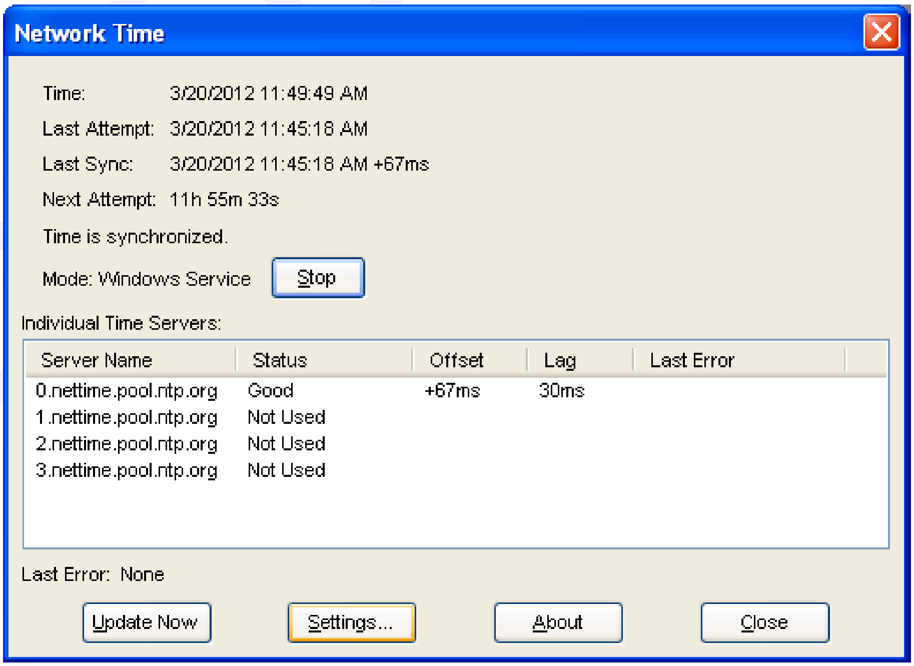


Figure 3.89: Properties for the NetTime server

In the Network Time GUI, see Figure 3.90, click Settings. . .



Time: 3/20/2012 11:49:49 AM
Last Attempt: 3/20/2012 11:45:18 AM
Last Sync: 3/20/2012 11:45:18 AM +67ms
Next Attempt: 11h 55m 33s
Time is synchronized.
Mode: Windows Service

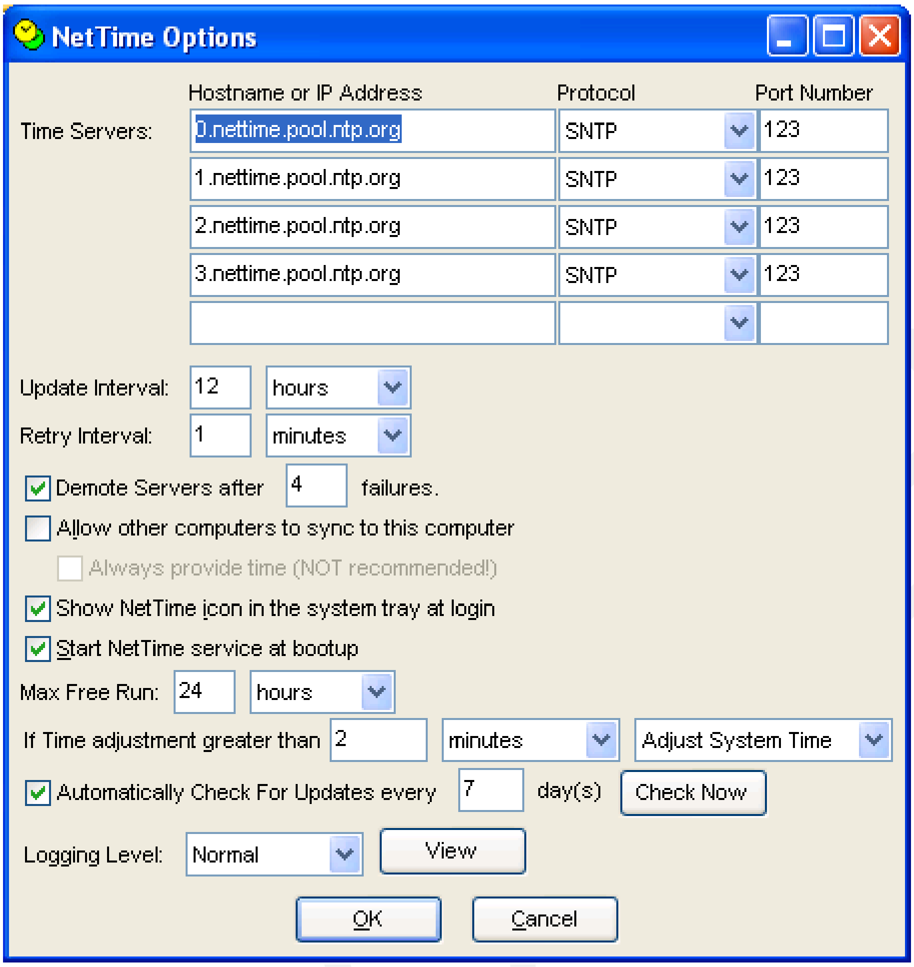
Individual Time Servers:

Server Name	Status	Offset	Lag	Last Error
0.nettime.pool.ntp.org	Good	+67ms	30ms	
1.nettime.pool.ntp.org	Not Used			
2.nettime.pool.ntp.org	Not Used			
3.nettime.pool.ntp.org	Not Used			

Last Error: None

Figure 3.90: Network Time GUI

By Default the checkboxes allow other computers to sync to this computer and always provide times are unchecked.



NetTime Options

Time Servers:

Hostname or IP Address	Protocol	Port Number
0.nettime.pool.ntp.org	SNTP	123
1.nettime.pool.ntp.org	SNTP	123
2.nettime.pool.ntp.org	SNTP	123
3.nettime.pool.ntp.org	SNTP	123

Update Interval: 12 hours

Retry Interval: 1 minutes

☑ Demote Servers after 4 failures.

□ Allow other computers to sync to this computer

□ Always provide time (NOT recommended!)

☑ Show NetTime icon in the system tray at login

☑ Start NetTime service at bootup

Max Free Run: 24 hours

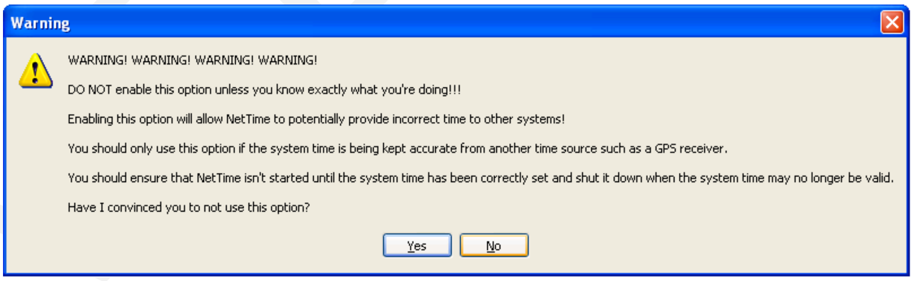
If Time adjustment greater than 2 minutes Adjust System Time

☑ Automatically Check For Updates every 7 day(s)

Logging Level: Normal

Figure 3.91: NetTime Options

Check Both Allow other computers to sync to this computer and always provide time to enable the NTP server. When checking Always provides time the following warning is presented:



Warning

WARNING! WARNING! WARNING! WARNING!

DO NOT enable this option unless you know exactly what you're doing!!!

Enabling this option will allow NetTime to potentially provide incorrect time to other systems!

You should only use this option if the system time is being kept accurate from another time source such as a GPS receiver.

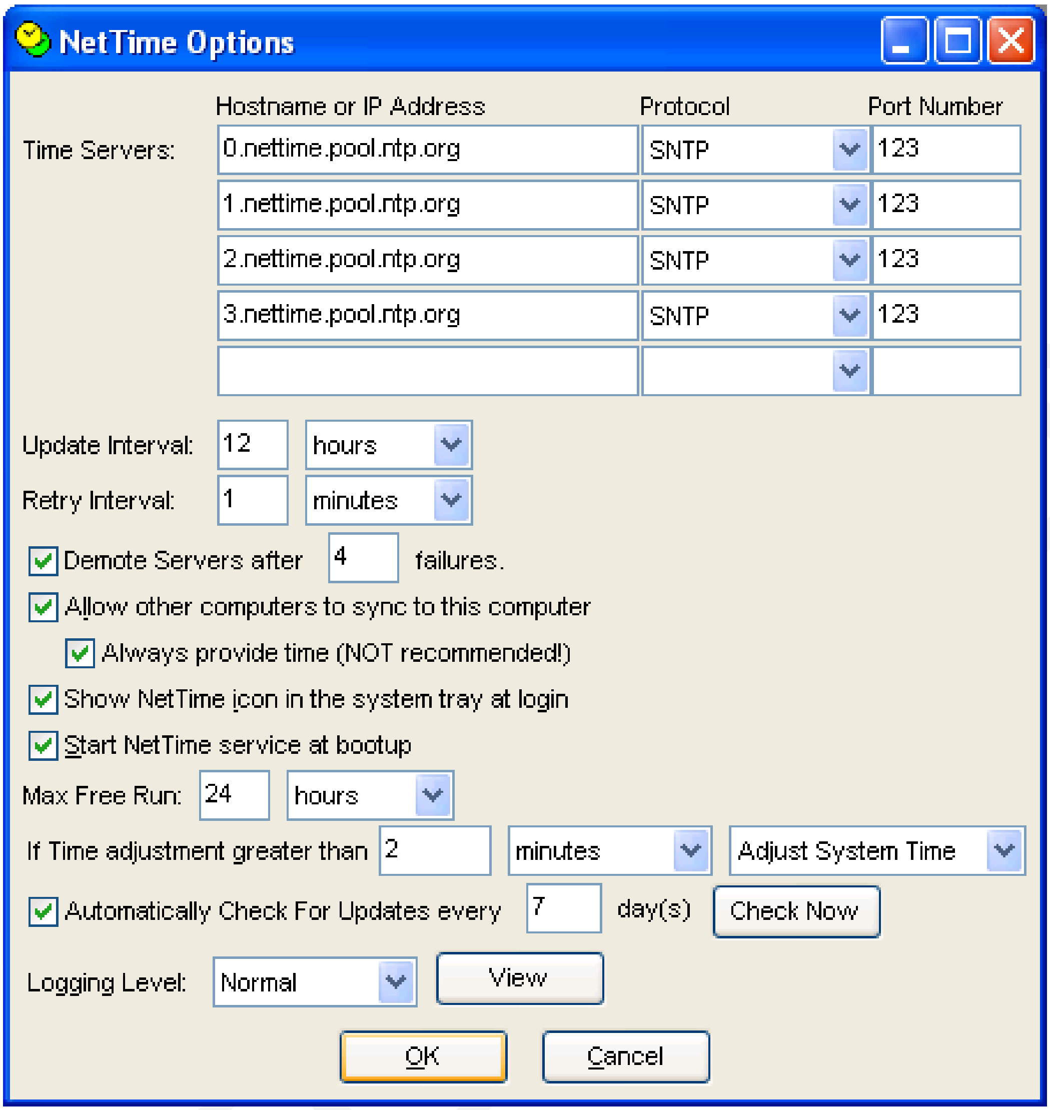
You should ensure that NetTime isn't started until the system time has been correctly set and shut it down when the system time may no longer be valid.

Have I convinced you to not use this option?

Figure 3.92: Warning dialog

Click No, in the Warning dialog which means you are ignoring the risk posed by enabling this option.

Finally the NetTime Options should be as follows.



Time Servers:

Hostname or IP Address	Protocol	Port Number
0.nettime.pool.ntp.org	SNTP	123
1.nettime.pool.ntp.org	SNTP	123
2.nettime.pool.ntp.org	SNTP	123
3.nettime.pool.ntp.org	SNTP	123

Update Interval: 12 hours

Retry Interval: 1 minutes

☑ Demote Servers after 4 failures.

☑ Allow other computers to sync to this computer

☑ Always provide time (NOT recommended!)

☑ Show NetTime icon in the system tray at login

☑ Start NetTime service at bootup

Max Free Run: 24 hours

If Time adjustment greater than 2 minutes Adjust System Time

☑ Automatically Check For Updates every 7 day(s)

Logging Level: Normal

Figure 3.93: Correct set NetTime Options

Click Ok, and the NetTime server is ready to user. In case of connection problems try disabling the host machines firewall.

Then changing the PC time in Windows manually effects the time set on the AWC 500 .

The UTC time is always transferred between the NTP server and NTP client, and time is presented with the local time offsets on PC and AWC 500 .

AWC 500 (NTP Client)				Server (NTP Server)
		0) Current time Client != Server		**Time Zone setting is Beijing...**
03:55:21 (UTC)				12:00:00(UTC+8)
		1) Request time →		
	04:00:01(UTC)	← 2) Response	04:00:01(UTC)	12:00:01
04:00:01 (UTC) 04:00:01 (CoDeSys RTC time)	TZ=UTC	3a) New time set in AWC 500		
12:00:01(Local) 04:00:01 (CoDeSys RTC time)	TZ=Local-8	3b) New time set in AWC 500 Client == Server		

The AWC 500 only operates with UTC time. Full Linux Time zones support is not implemented.
It is a design decision by the DEIF AWC 500 team, that all presentation of the time to the user should be offset from UTC to local time.

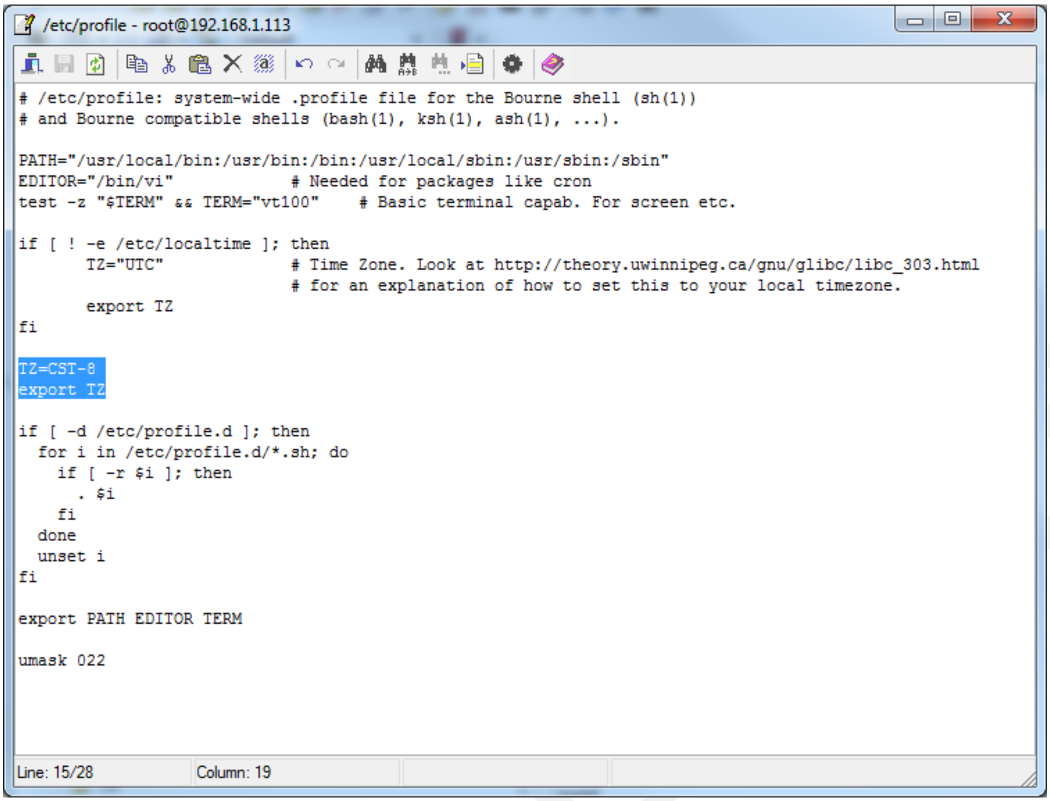
This is to avoid confusion of what time is used in logged files.

The correct localtime when viewing with "date" can still be set up.
And localtime also needs to be handled in the PLC application.

Correct localtime when viewing with "date" can still be set up.

In `/etc/profile` set the Time zone variable, by adding the lines:

```
TZ=CST−8                              (CST (China Standard Time) −8 hours offset)
export TZ
```



```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin"
EDITOR="/bin/vi"            # Needed for packages like cron
test -z "$TERM" && TERM="vt100"    # Basic terminal capab. For screen etc.

if [ ! -e /etc/localtime ]; then
       TZ="UTC"             # Time Zone. Look at http://theory.uwinnipeg.ca/gnu/glibc/libc_303.html
                            # for an explanation of how to set this to your local timezone.
       export TZ
fi

TZ=CST-8
export TZ

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

export PATH EDITOR TERM

umask 022
```

Figure 3.94: Edit profile

To see the effect:

```
~ # date
Mon Feb 20 20:22:04 CST 2012    (Now the China Standard Time (CST) is used)
~ #
~ # unset TZ                    ( unsets timezone )
~ # date
Mon Feb 20 12:22:15 UTC 2012    ( now showing the UTC time again).
```

A reboot will use CST again.

Localtime handling in the PLC application

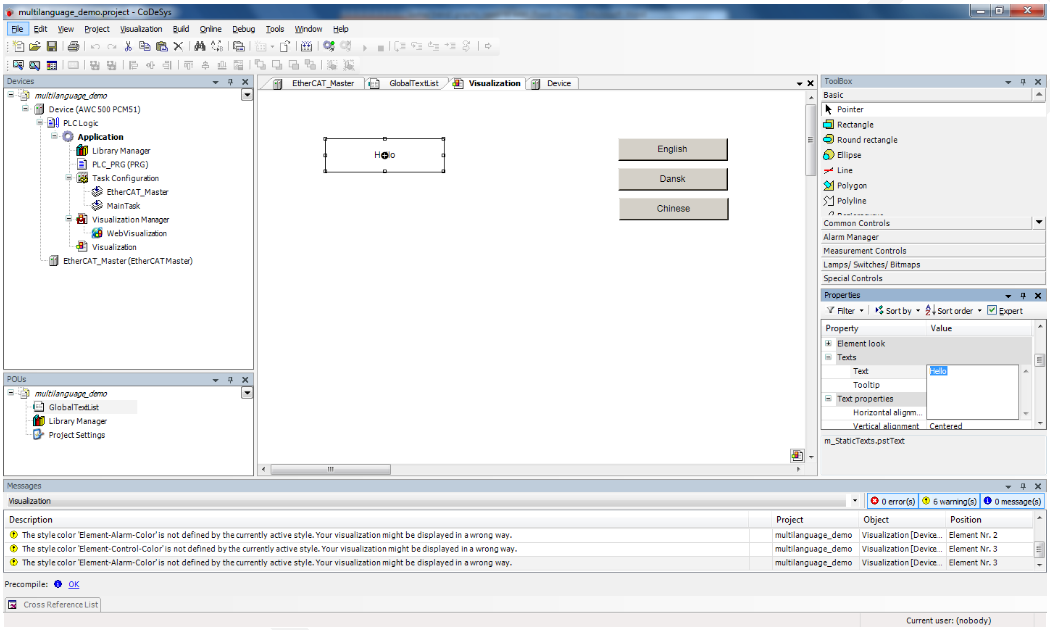
In PLC application we only want the HMI visualization time to show the local time. All logging should still use the UTC time.
A suggestion for handling is simple offset of the UTC time:

```
VAR
    UTCTime_ms  : SysTime;
    LocalTime_ms  : SysTime;
    LocalTime_sys : SYSTIMEDATE;
END_VAR

(* Return the UTC time *)
SysTimeRtcHighResGet(pTimestamp := UTCTime_ms);
(* Offsets the UTC time with 8 hours *)
LocalTime_ms := UTCTime_ms + (8*60*60*1000);
SysTimeRtcConvertHighResToDate(pTimestamp := LocalTime_ms, pDate := LocalTime_sys);
```

The Time synchronization is floating and takes place up to ever 3rd hour automatically, a scheduled function can be made like a script instead.

4 Adding Multilanguage support to visualisations

Make sure to use the font Arial Unicode Ms. in Windows it needs to be enabled in the fonts configuration before it is possible to select in CoDeSys.
Each time a text is added in CoDeSys, it is added to the GlobalTextList, together with a unique ID.



Property	Value
Element look	
Texts	
Text	Hello
Tooltip	
Text properties	
Horizontal alignm...	
Vertical alignment	Centered

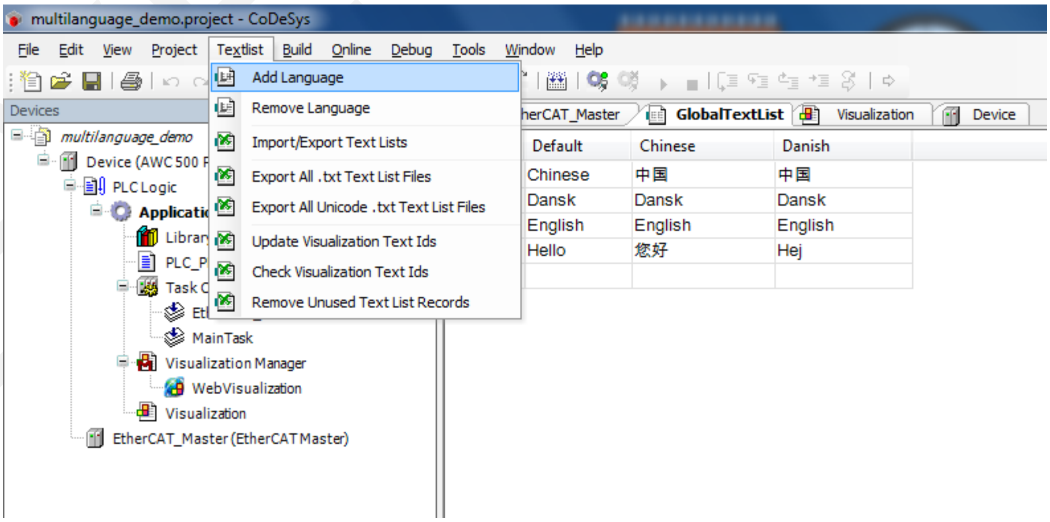
m_StaticTexts.pstText

Visualization — 0 error(s) | 6 warning(s) | 0 message(s)

Description	Project	Object	Position
The style color 'Element-Alarm-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 2
The style color 'Element-Control-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 3
The style color 'Element-Alarm-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 3

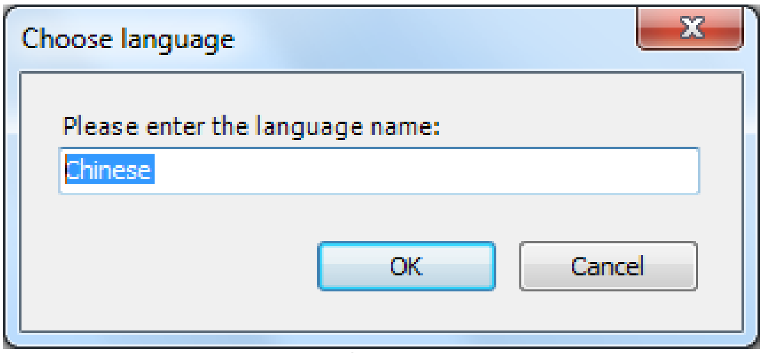
Figure 4.1: Texts are added to the GlobalTextList

In GlobalTextList additional languages can be added different from the design language (default)


multilanguage_demo.project - CoDeSys
File Edit View Project Textlist Build Online Debug Tools Window Help
Devices
multilanguage_demo
Device (AWC 500 P
PLC Logic
Applicatio
Librar
PLC_P
Task C
Et
MainTask
Visualization Manager
WebVisualization
Visualization
EtherCAT_Master (EtherCAT Master)
Add Language
Remove Language
Import/Export Text Lists
Export All .txt Text List Files
Export All Unicode .txt Text List Files
Update Visualization Text Ids
Check Visualization Text Ids
Remove Unused Text List Records
herCAT_Master
GlobalTextList
Visualization
Device

Default	Chinese	Danish
Chinese	中国	中国
Dansk	Dansk	Dansk
English	English	English
Hello	您好	Hej

Figure 4.2: Add Language



Choose language

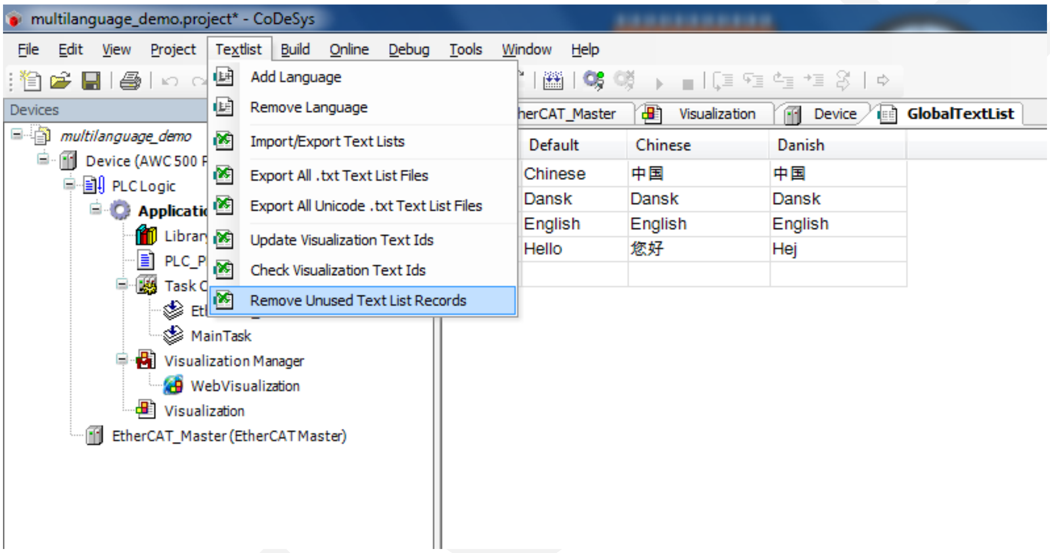
Please enter the language name:

Chinese

OK Cancel

Figure 4.3: Choose Language

Unused Static texts can be removed by:

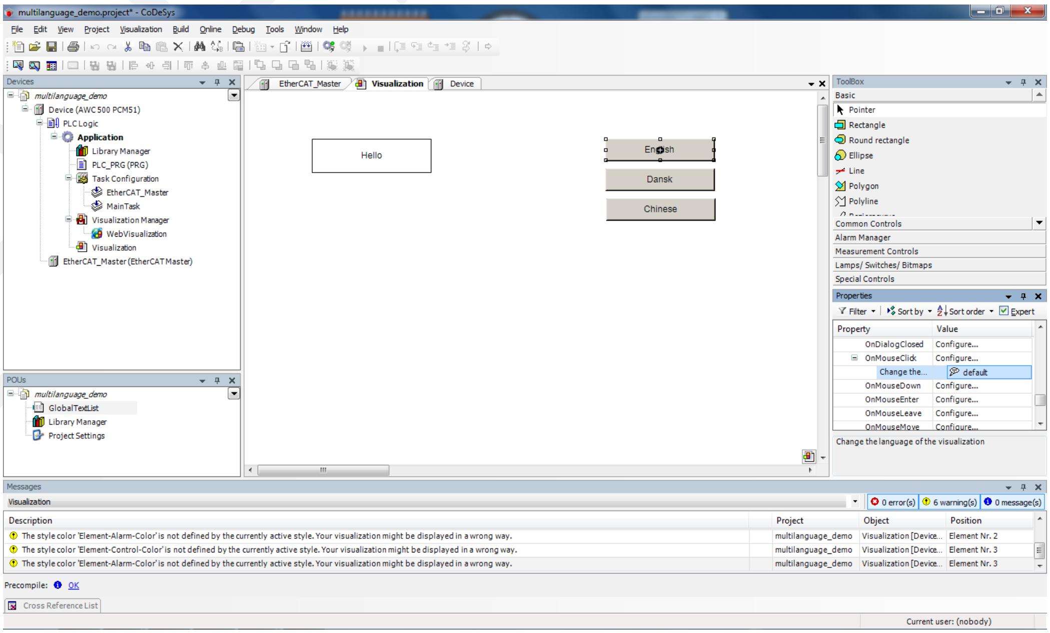


multilanguage_demo.project* - CoDeSys
File Edit View Project Textlist Build Online Debug Tools Window Help
Add Language
Remove Language
Import/Export Text Lists
Export All .txt Text List Files
Export All Unicode .txt Text List Files
Update Visualization Text Ids
Check Visualization Text Ids
Remove Unused Text List Records
Devices
multilanguage_demo
Device (AWC 500 P
PLC Logic
Applicati
Librar
PLC_P
Task C
Et
MainTask
Visualization Manager
WebVisualization
Visualization
EtherCAT_Master (EtherCAT Master)
herCAT_Master
Visualization
Device
GlobalTextList

Default	Chinese	Danish
Chinese	中国	中国
Dansk	Dansk	Dansk
English	English	English
Hello	您好	Hej

Figure 4.4: Remove Unused Text List Records

Insert boxes to change language:



Property	Value
OnDialogClosed	Configure...
OnMouseClick	Configure...
Change the...	default
OnMouseDown	Configure...
OnMouseEnter	Configure...
OnMouseLeave	Configure...
OnMouseMove	Configure...

Change the language of the visualization

Messages

Visualization — 0 error(s) 6 warning(s) 0 message(s)

Description	Project	Object	Position
The style color 'Element-Alarm-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 2
The style color 'Element-Control-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 3
The style color 'Element-Alarm-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 3

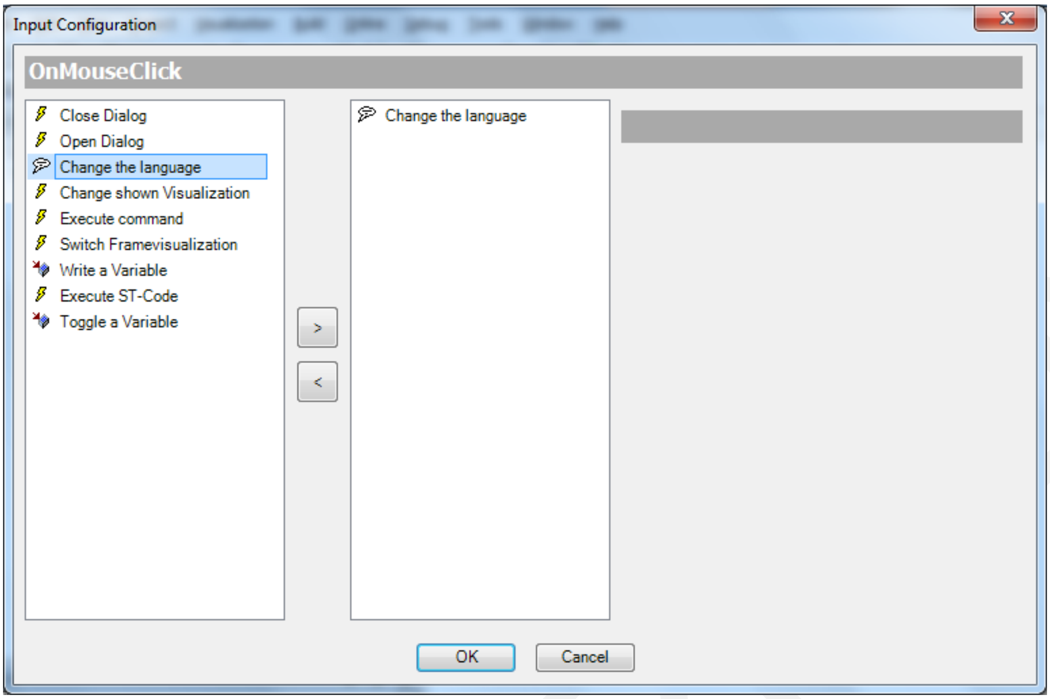
Precompile: OK

Cross Reference List

Current user: (nobody)

Figure 4.5: Boxes to change language

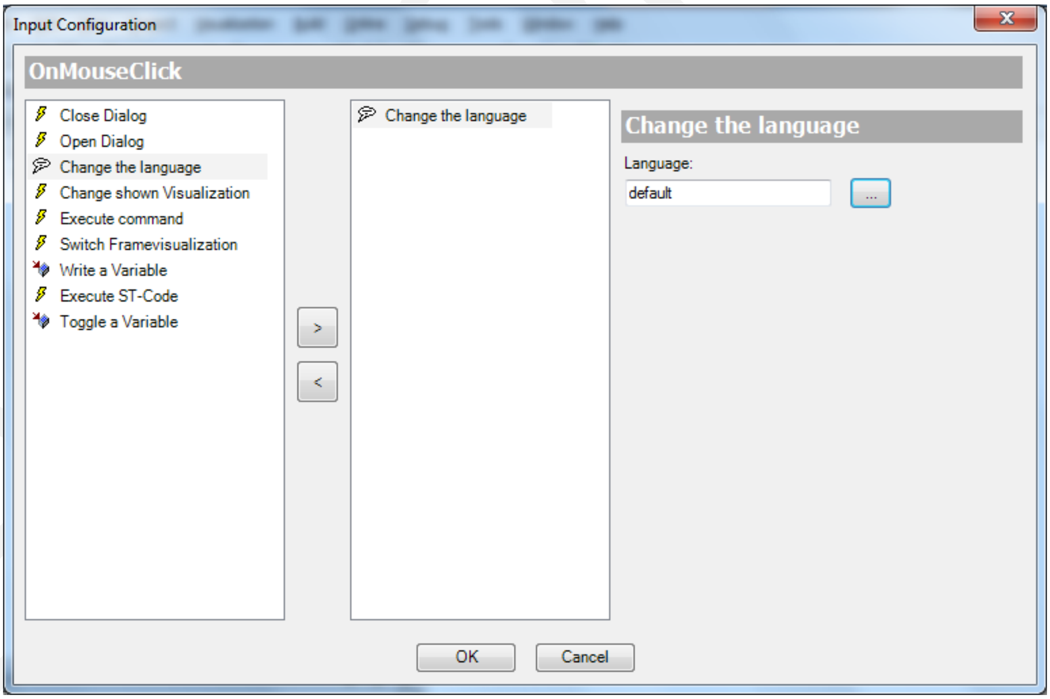
Add the "Change the language":



Change the language

Figure 4.6: Boxes to change language

Write "default" for designer language.



Change the language

Change the language

Language: default

Figure 4.7: Write "default" for designer language

Or select any of the added languages:

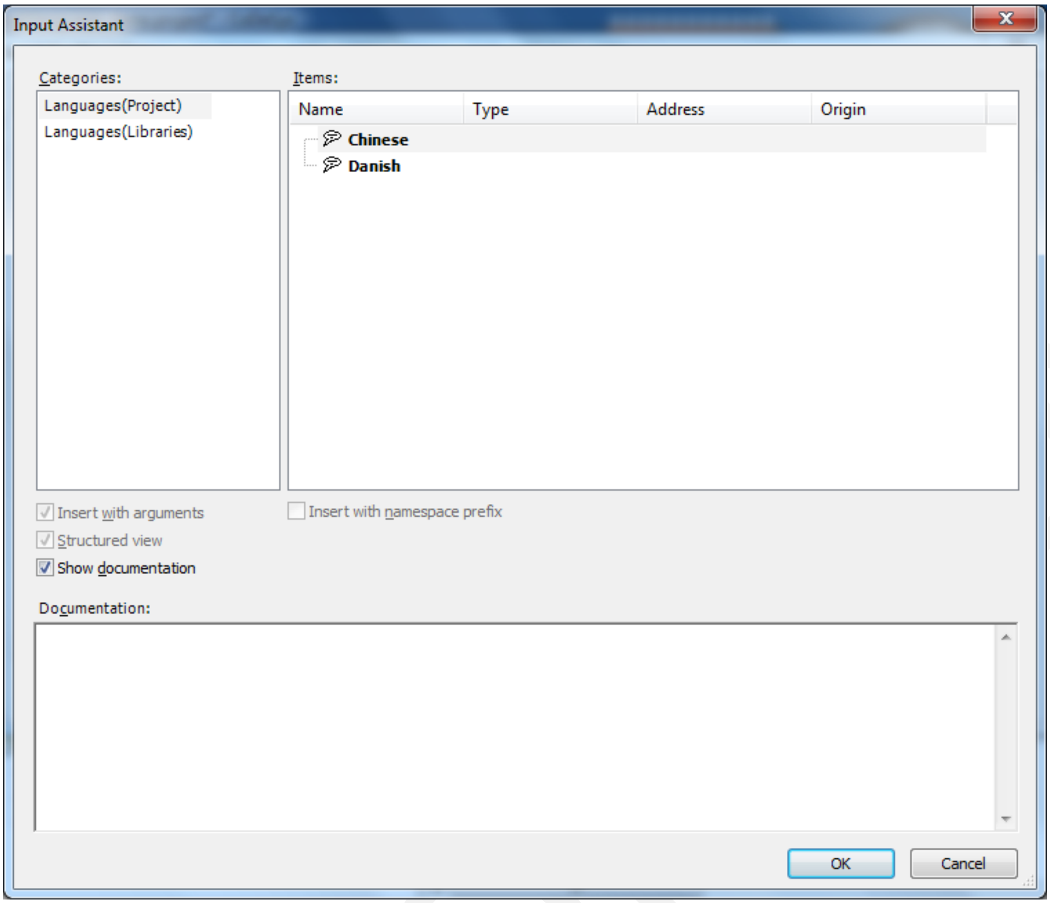
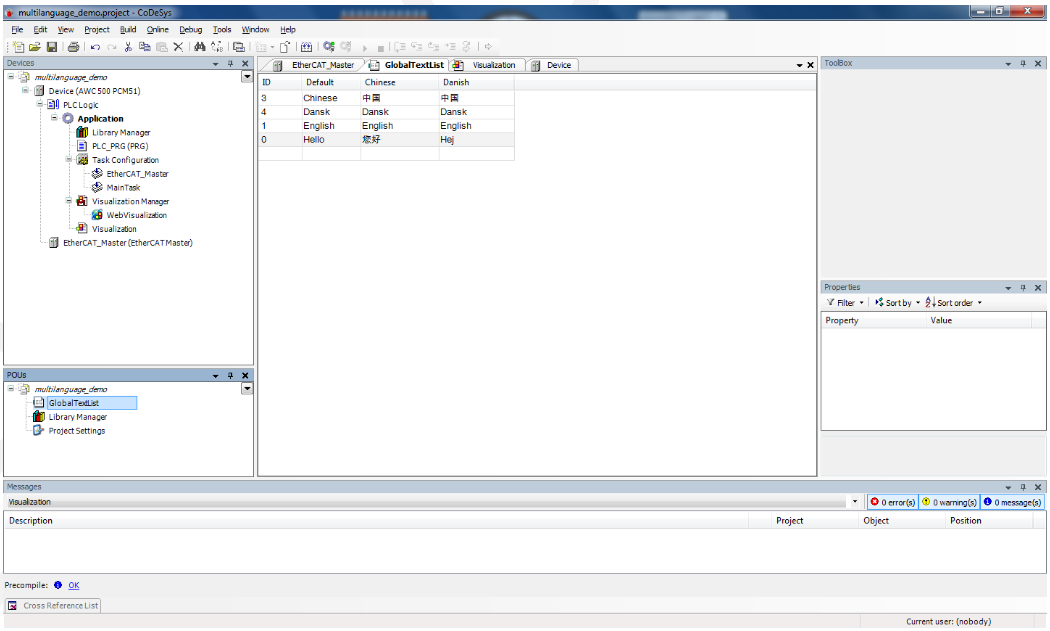
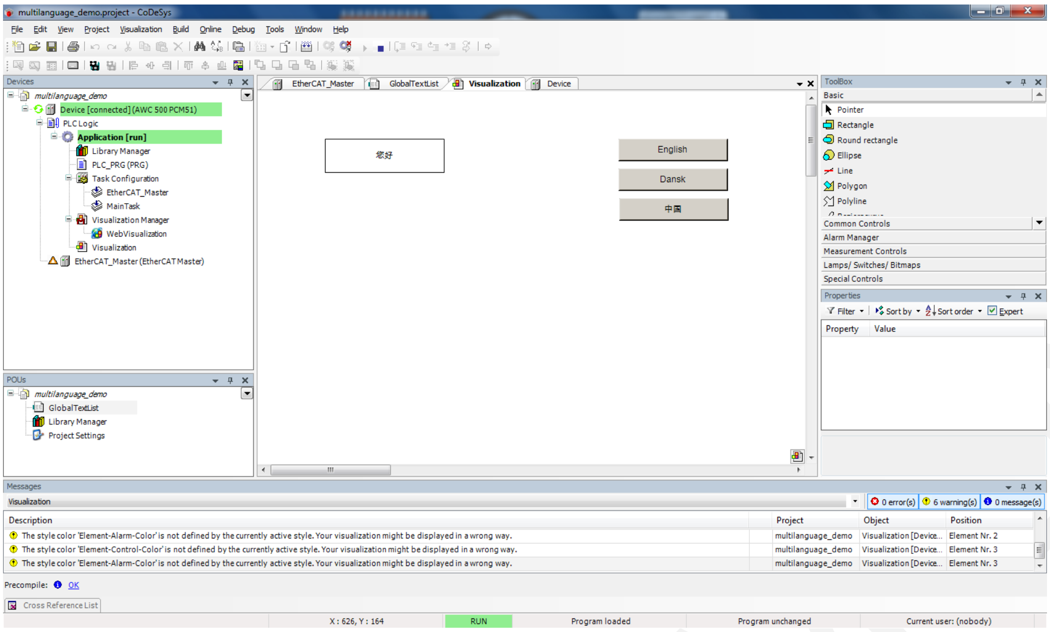


Figure 4.8: Available languages



ID	Default	Chinese	Danish
3	Chinese	中国	中国
4	Dansk	Dansk	Dansk
1	English	English	English
0	Hello	您好	Hej

Figure 4.9: GlobalTextList



Description	Project	Object	Position
The style color 'Element-Alarm-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 2
The style color 'Element-Control-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 3
The style color 'Element-Alarm-Color' is not defined by the currently active style. Your visualization might be displayed in a wrong way.	multilanguage_demo	Visualization [Device...	Element Nr. 3

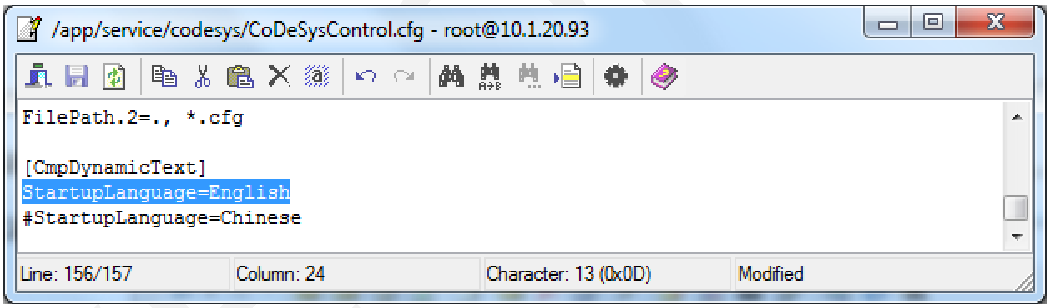
Precompile: OK

X : 626, Y : 164 RUN Program loaded Program unchanged Current user: (nobody)

Figure 4.10: Chinese language example

4.1 Set Chinese as default startup language

Set up Chinese as default startup language in webvisualization(Fixed by CoDeSys 3.4.4.60)
The default startup language by the webvisualization can be changed from "default" to "Chinese" by setting `StartupLanguage = Chinese` (in `/app/service/codesys/CoDeSysControl.cfg`)



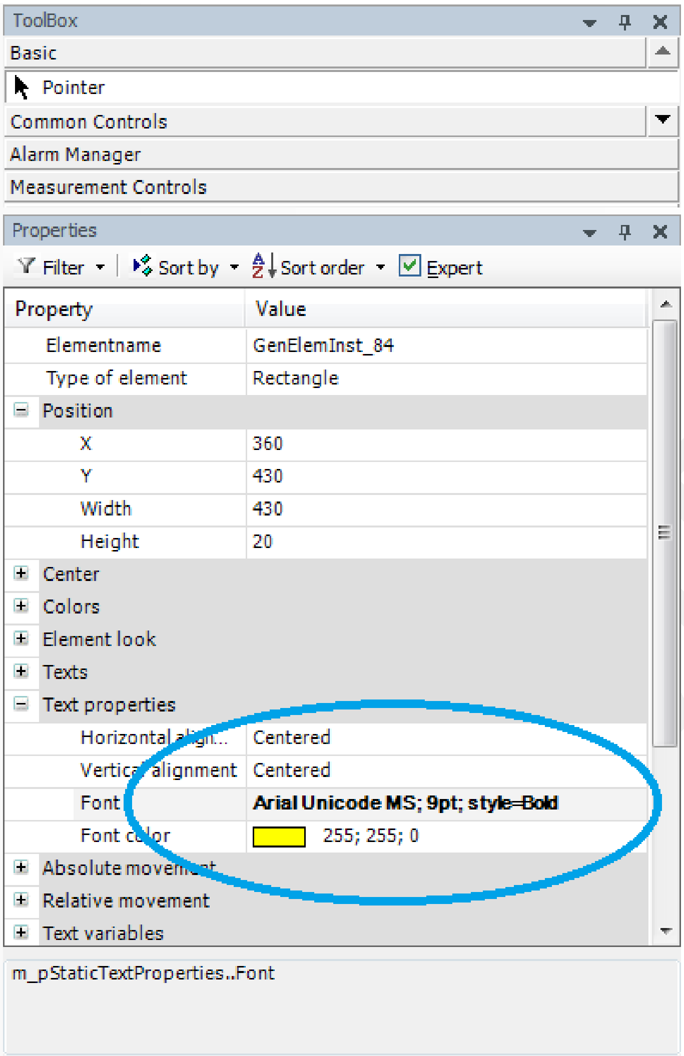
```
FilePath.2=., *.cfg

[CmpDynamicText]
StartupLanguage=English
#StartupLanguage=Chinese
```

Line: 156/157 Column: 24 Character: 13 (0x0D) Modified

Figure 4.11: Edit CoDeSysControl.cfg

Note: In CoDeSys the default language is still used when opened.
DEIF provides an dUpdate script changing this StartupLanguage.

All fonts on items has been changed to Arial Unicode MS on aSYSTEMOVERVIEW page.

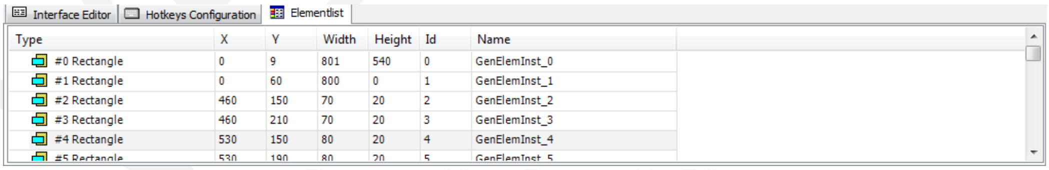


Property	Value
Elementname	GenElemInst_84
Type of element	Rectangle
⊟ Position	
X	360
Y	430
Width	430
Height	20
⊞ Center	
⊞ Colors	
⊞ Element look	
⊞ Texts	
⊟ Text properties	
Horizontal align...	Centered
Vertical alignment	Centered
Font	**Arial Unicode MS; 9pt; style=Bold**
Font color	🟨 255; 255; 0
⊞ Absolute movement	
⊞ Relative movement	
⊞ Text variables	

m_pStaticTextProperties..Font

Figure 4.12: Change font to Arial Unicode MS

Tip : Use the "Visualization→Visual Elements List Editor" to walkthrough all text elements.



Type	X	Y	Width	Height	Id	Name
#0 Rectangle	0	9	801	540	0	GenElemInst_0
#1 Rectangle	0	60	800	0	1	GenElemInst_1
#2 Rectangle	460	150	70	20	2	GenElemInst_2
#3 Rectangle	460	210	70	20	3	GenElemInst_3
#4 Rectangle	530	150	80	20	4	GenElemInst_4
#5 Rectangle	530	190	80	20	5	GenElemInst_5

Figure 4.13: Visual Elements List Editor

4.2 Webvisualisation or Panel PC HMI preparations

For webvisualisation or the Panel PC need to have java installed:

- Online via internet (via `www.java.com/getjava`)
- Or Offline installation (`jre-6u26-windows-i586-s.exe`) via USB stick: (`http://www.java.com/en/download/manual.jsp`)
  - Select Windows 7,XP Offline.
  - Then save the file to USB stick

One installed test the visualization with
`http://[ip]:8080/webvisu.htm` to view the visualisation

Make this webpage the default startup page of explorer.

4.2.1 How to remove scrollbars

As per default the Internet Explorer shows scrollbars. We propose that the generated webvisu.htm is changed manually after generating a boot project in offline mode. Then the changes below to webvisu.htm can be made. Edit `/app/service/codesys/visu/webvisu.htm` on the AWC 500 .

```
<HTML>
        <HEAD>
        <TITLE>CoDeSys WebVisualization</TITLE>
        <style type="text/css">
        body
        {
                margin: 0;
                padding: 0;
                overflow:hidden;  <-Add this
        }
        </style>
        </HEAD>
        <BODY scroll="no" style="overflow:auto" >  <-Add this
        <APPLET CODEBASE=. CODE=_3S/CoDeSys/WebVisu/WebVisu.class name="WebVisu"
                                width="800" height="600" id="webvisuappletV3">
        <param name="archive" value="webvisuclient.jar,visualelements.jar">
        <param name="STARTVISU" value="aSYSTEMOVERVIEW">
        <param name="APPLICATION" value="Application">
        <param name="PLCADDRESS" value="000D">
        <param name="USELOCALHOST" value="TRUE">
        <param name="UPDATERATE" value="200">
        <param name="COMMBUFFERSIZE" value="1000000">
        <param name="BESTFIT" value="False">
        <param name="WEBVISUTYPE" value="Client">
        <param name="ANTIALIASING" value="0">


        </APPLET>
        </BODY>
</HTML>
```

5 Using boot application dUpdate builder

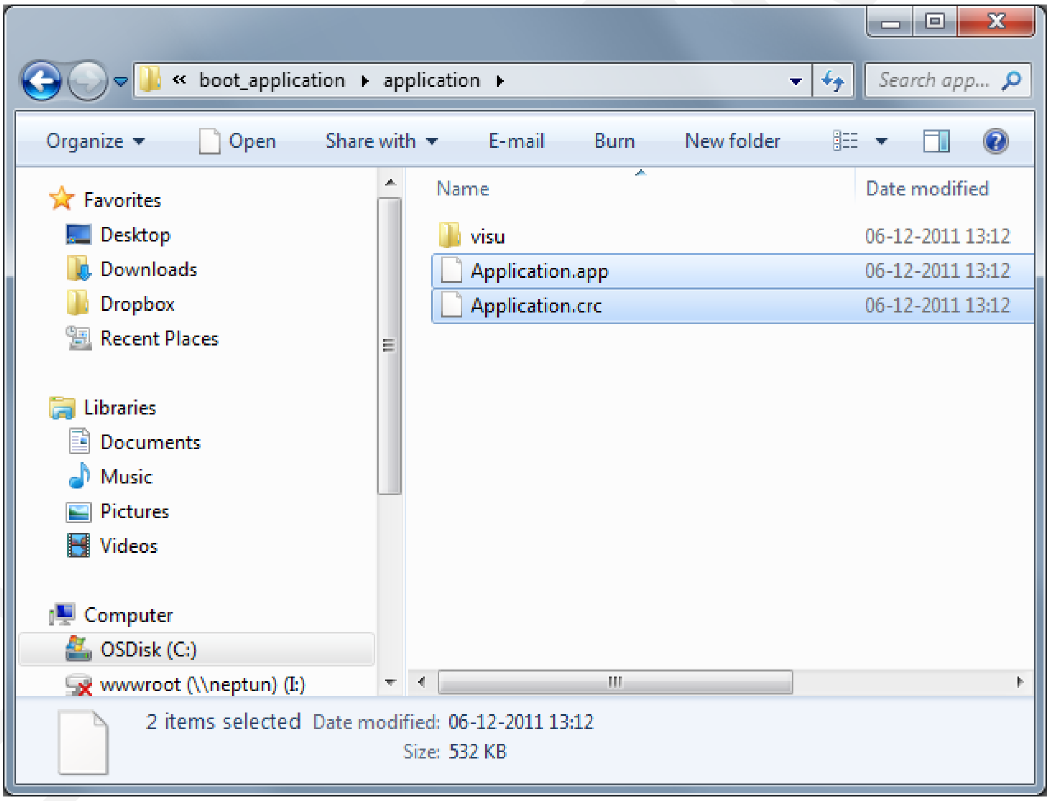
To make it easy to create a CoDeSys boot application package, the boot application .dUpdate builder is used. Then to do remote update of a project simply update copy the file application.dupate to `/tmp/fwupdates` on the AWC 500.

The steps are:

1. Delete existing CoDeSys boot application files.
2. Create new CoDeSys bootproject in offline mode.
3. Run build_dupdate.py script from CoDeSys.
4. Test the build of application.dupdate before distribution.

5.1 Step 1: Delete existing CoDeSys boot application files

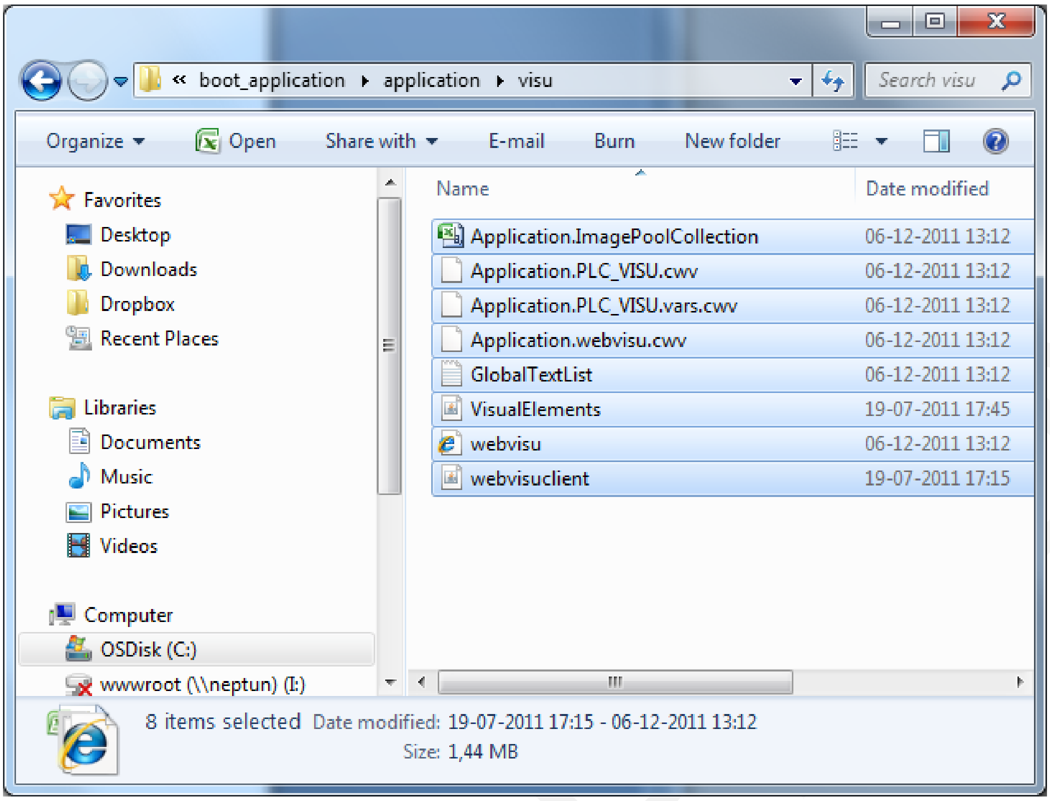
1. Delete existing CoDeSys boot application files from `\boot_application\application`:



Name	Date modified
visu	06-12-2011 13:12
Application.app	06-12-2011 13:12
Application.crc	06-12-2011 13:12

2 items selected Date modified: 06-12-2011 13:12
Size: 532 KB

2. Delete existing CoDeSys boot application visu files from `\boot_application\application\visu`:

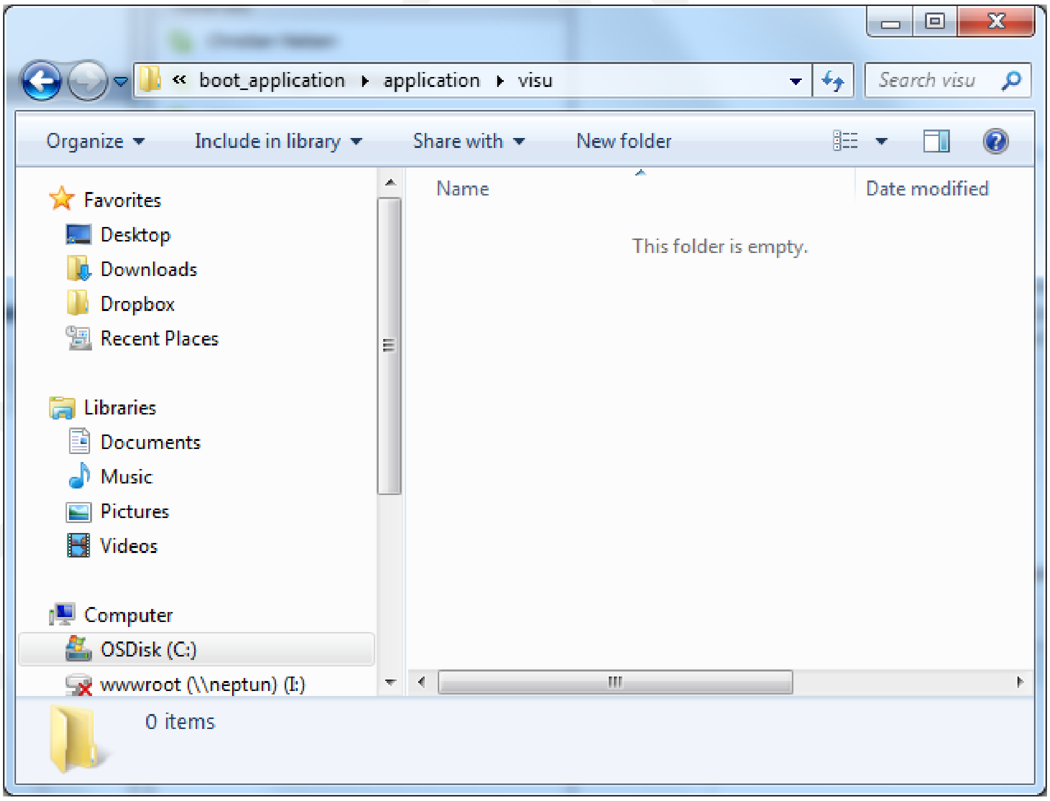


Name	Date modified
Application.ImagePoolCollection	06-12-2011 13:12
Application.PLC_VISU.cwv	06-12-2011 13:12
Application.PLC_VISU.vars.cwv	06-12-2011 13:12
Application.webvisu.cwv	06-12-2011 13:12
GlobalTextList	06-12-2011 13:12
VisualElements	19-07-2011 17:45
webvisu	06-12-2011 13:12
webvisuclient	19-07-2011 17:15

8 items selected Date modified: 19-07-2011 17:15 - 06-12-2011 13:12

Size: 1,44 MB

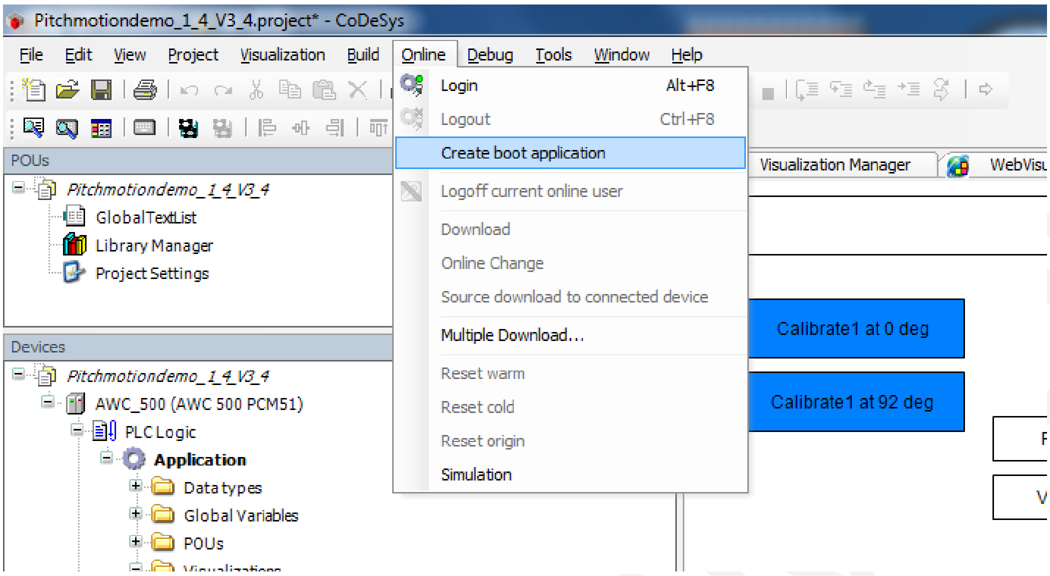
3. Then the folders should be empty.



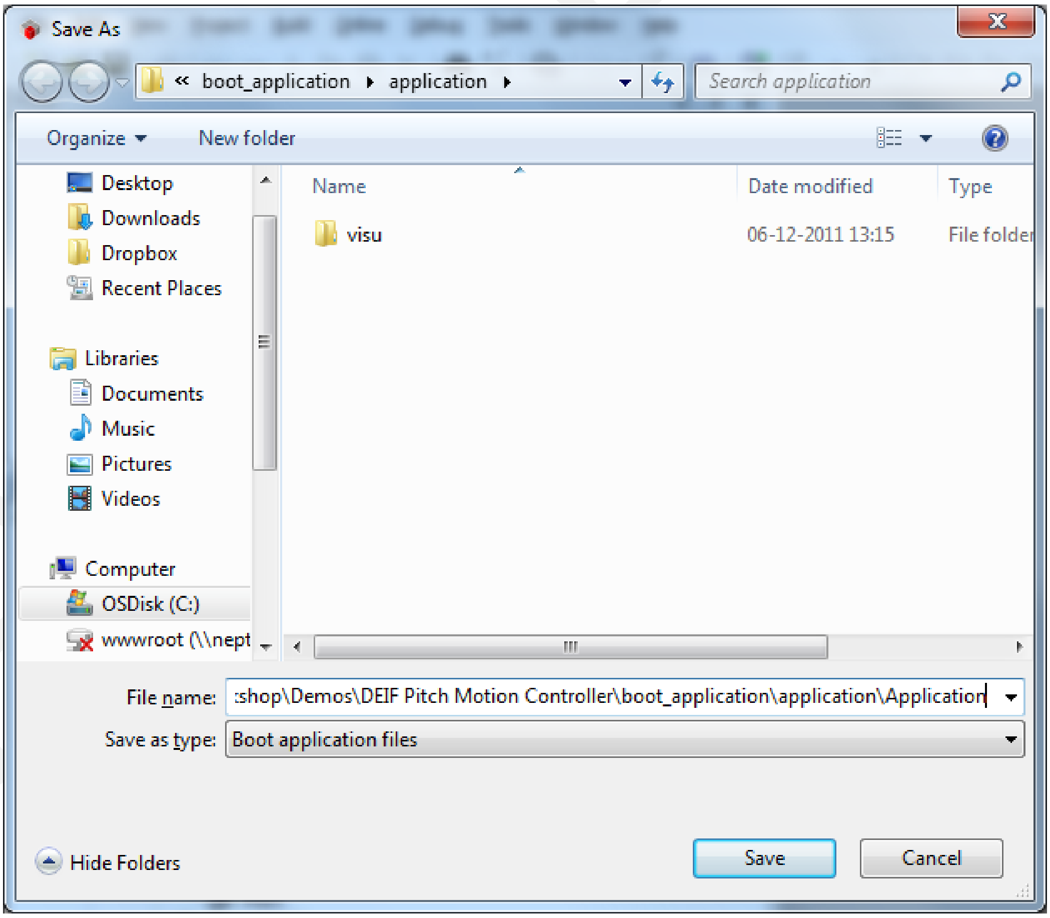
This folder is empty.

0 items

5.2 Step 2: Create new boot application from CoDeSys in offline mode (Online→Create boot application)



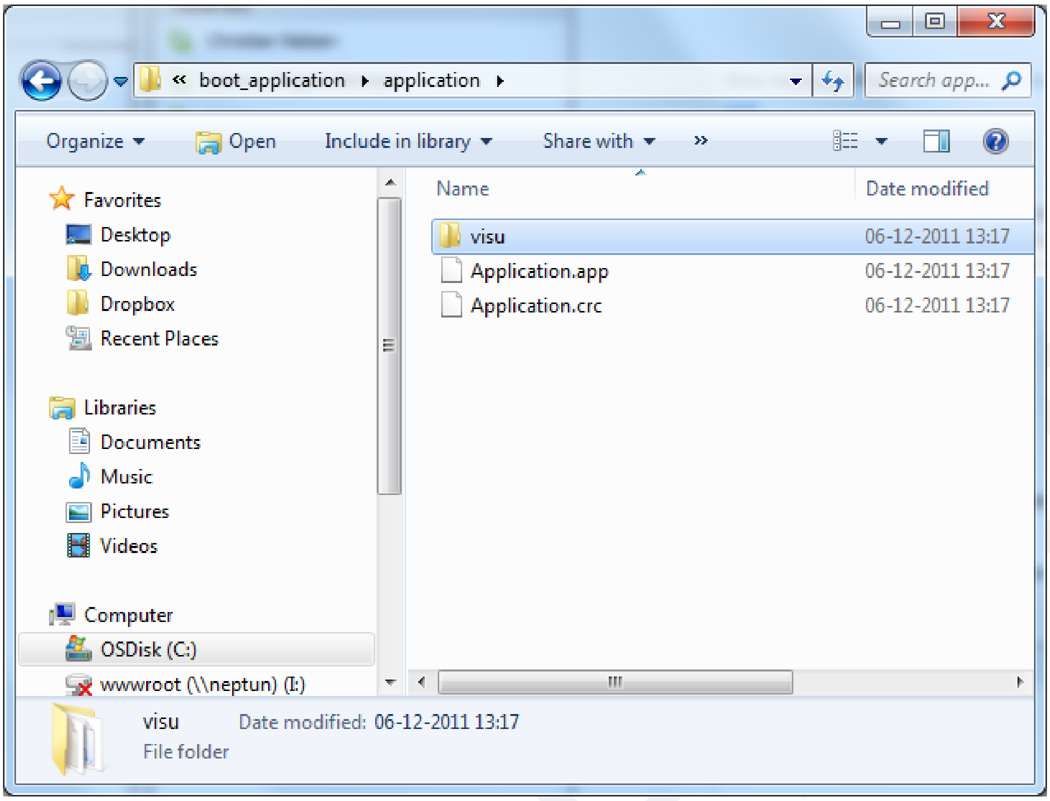
Save the files to the location `\boot_application\application using default names`:



File name: :shop\Demos\DEIF Pitch Motion Controller\boot_application\application\Application

Save as type: Boot application files

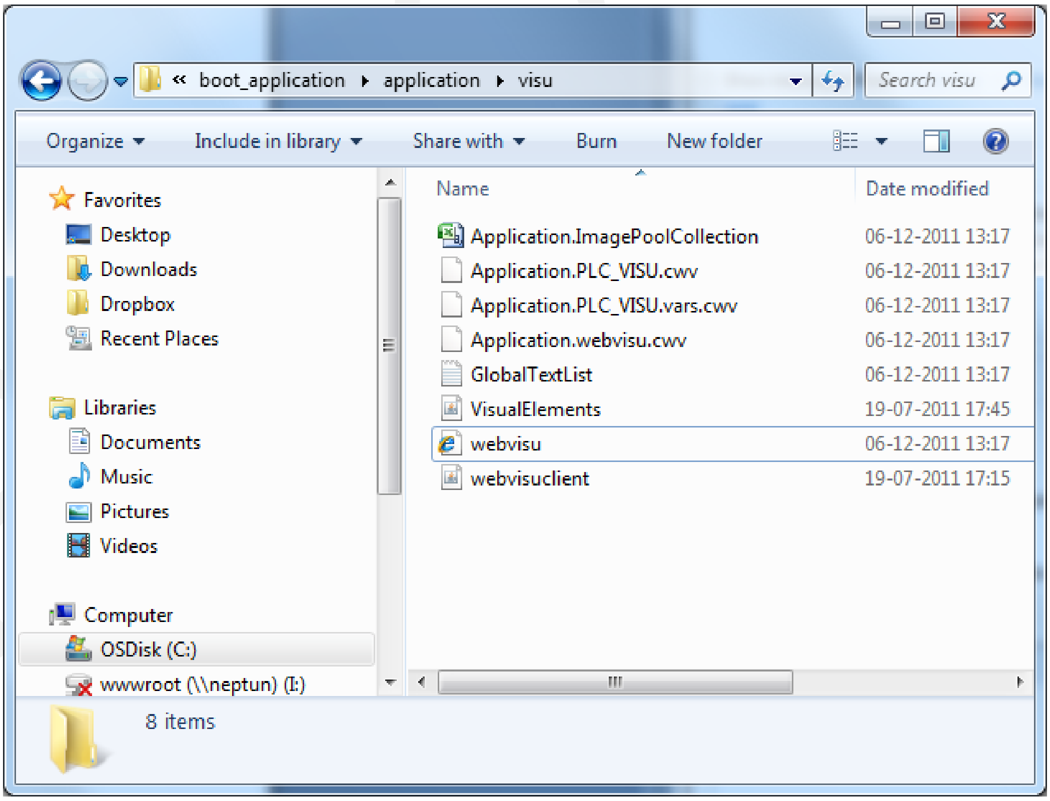
Then the boot application files are created:



Name	Date modified
visu	06-12-2011 13:17
Application.app	06-12-2011 13:17
Application.crc	06-12-2011 13:17

visu Date modified: 06-12-2011 13:17
File folder

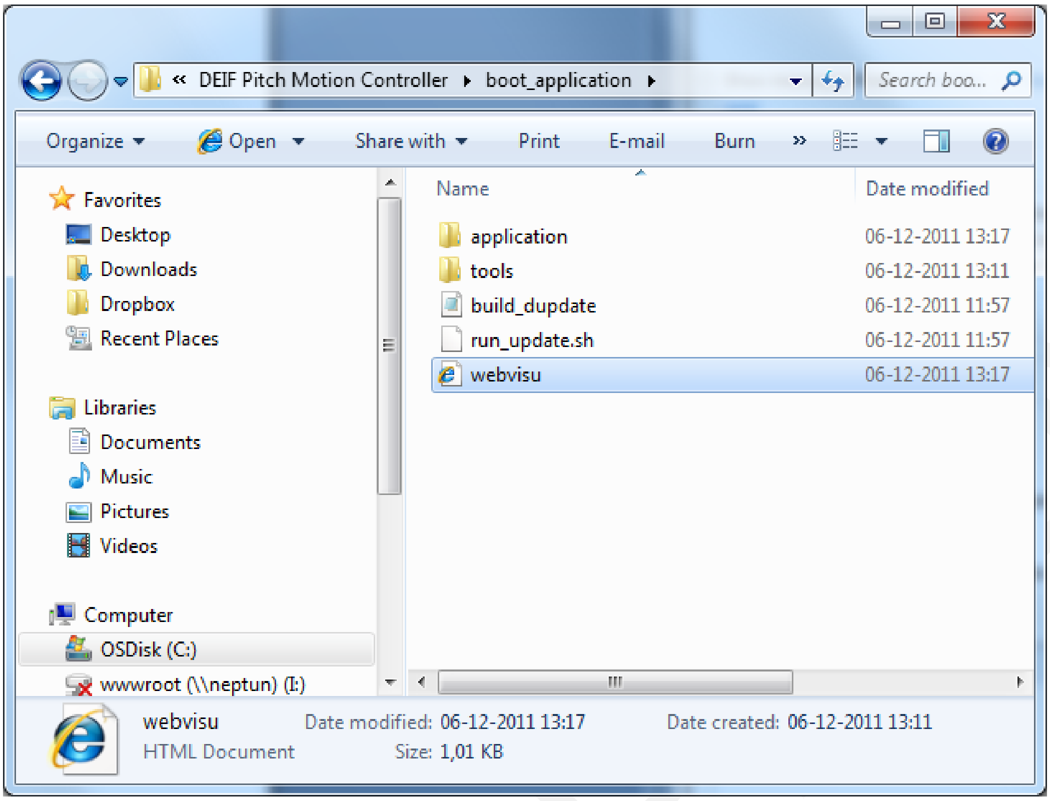
Option: If you want make a customized version of the webvisualisation, then copy the file webvisu.htm from `\boot_application\a` to above path `\boot_application`.



Name	Date modified
Application.ImagePoolCollection	06-12-2011 13:17
Application.PLC_VISU.cwv	06-12-2011 13:17
Application.PLC_VISU.vars.cwv	06-12-2011 13:17
Application.webvisu.cwv	06-12-2011 13:17
GlobalTextList	06-12-2011 13:17
VisualElements	19-07-2011 17:45
webvisu	06-12-2011 13:17
webvisuclient	19-07-2011 17:15

8 items

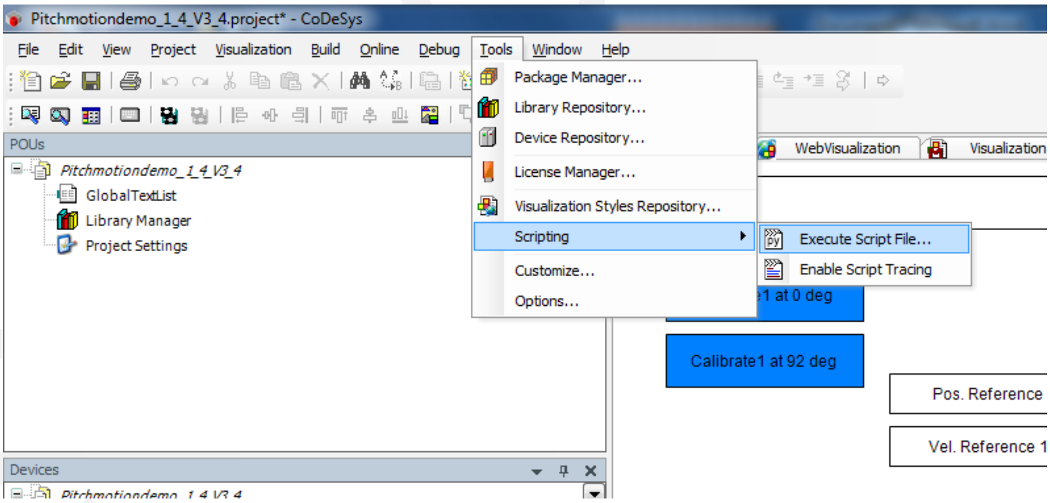
Edit the webvisu.htm if required.



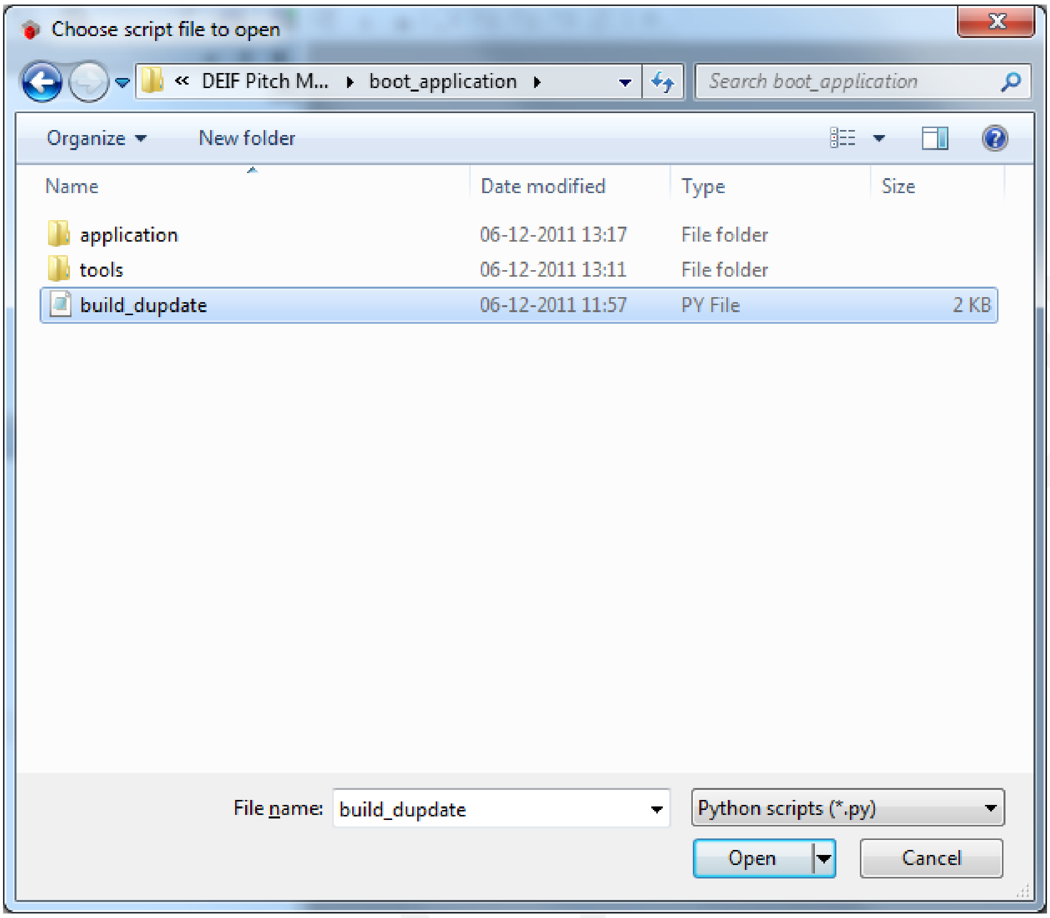
Name	Date modified
application	06-12-2011 13:17
tools	06-12-2011 13:11
build_dupdate	06-12-2011 11:57
run_update.sh	06-12-2011 11:57
webvisu	06-12-2011 13:17

webvisu Date modified: 06-12-2011 13:17 Date created: 06-12-2011 13:11
HTML Document Size: 1,01 KB

5.3 Step 3) Run build_dupdate.py script from CoDeSys (Tools→Scripting→ExecuteScript File. . .)



Select build_dupdate.py

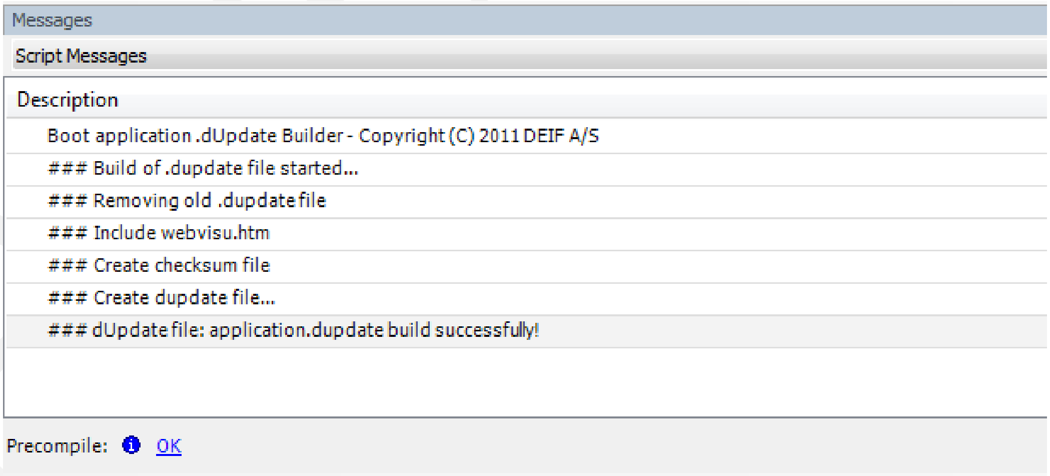

Choose script file to open
« DEIF Pitch M... ▸ boot_application ▸
Search boot_application
Organize ▾
New folder
Open
Cancel

Name	Date modified	Type	Size
application	06-12-2011 13:17	File folder	
tools	06-12-2011 13:11	File folder	
build_dupdate	06-12-2011 11:57	PY File	2 KB

File name: build_dupdate

Python scripts (*.py)

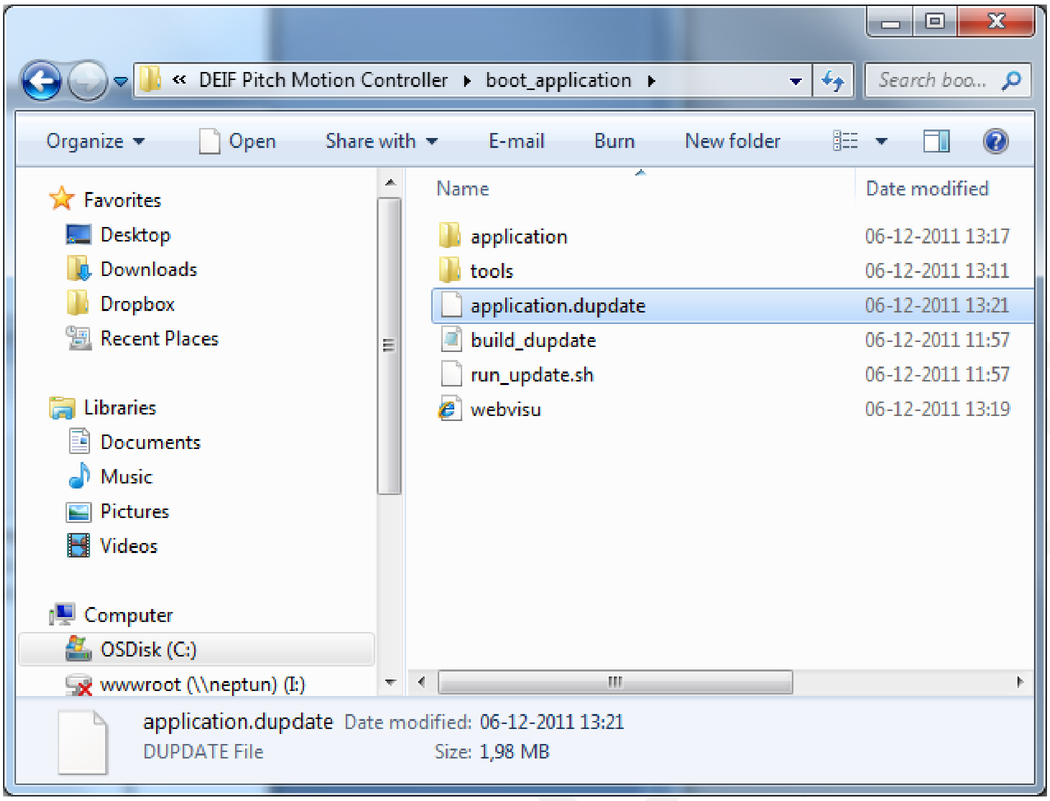
The output message:



```
Description
   Boot application .dUpdate Builder - Copyright (C) 2011 DEIF A/S
   ### Build of .dupdate file started...
   ### Removing old .dupdate file
   ### Include webvisu.htm
   ### Create checksum file
   ### Create dupdate file...
   ### dUpdate file: application.dupdate build successfully!
```

Precompile: ℹ OK

Here after the build application.dupdate file is available in `\boot_application`:

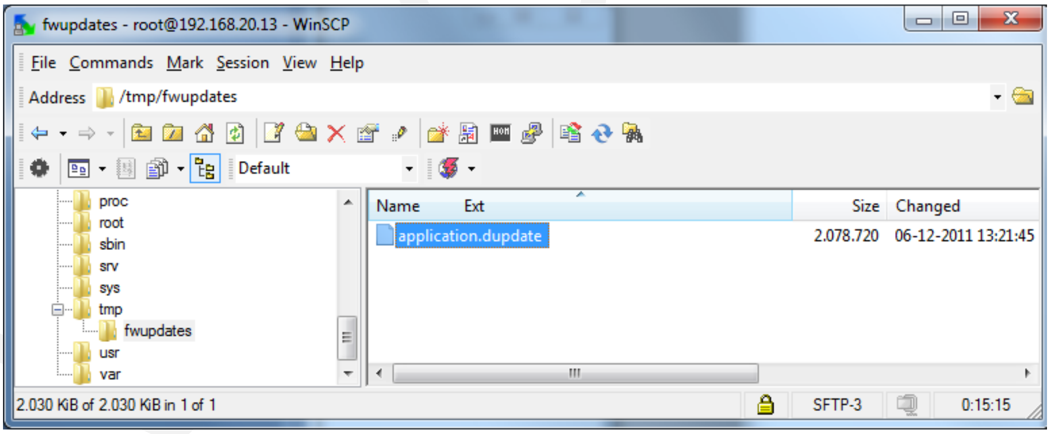


Name	Date modified
application	06-12-2011 13:17
tools	06-12-2011 13:11
application.dupdate	06-12-2011 13:21
build_dupdate	06-12-2011 11:57
run_update.sh	06-12-2011 11:57
webvisu	06-12-2011 13:19

application.dupdate Date modified: 06-12-2011 13:21
DUPDATE File Size: 1,98 MB

5.4 Step 4) Test the build of application.dupdate before distribution.

To update the application on the AWC 500 copy application.dupdate to `/tmp/fwupdate`

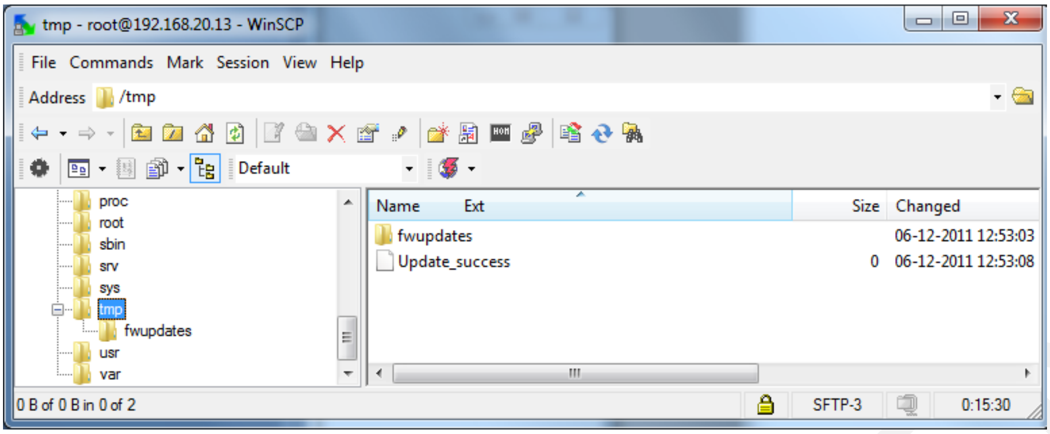

fwupdates - root@192.168.20.13 - WinSCP
File Commands Mark Session View Help
Address /tmp/fwupdates
Default
proc
root
sbin
srv
sys
tmp
fwupdates
usr
var

Name Ext	Size	Changed
application.dupdate	2.078.720	06-12-2011 13:21:45

2.030 KiB of 2.030 KiB in 1 of 1 SFTP-3 0:15:15

The uses the same procedure like updating firmware on the AWC 500:

After successful update the file Update_success is available:


tmp - root@192.168.20.13 - WinSCP
File Commands Mark Session View Help
Default

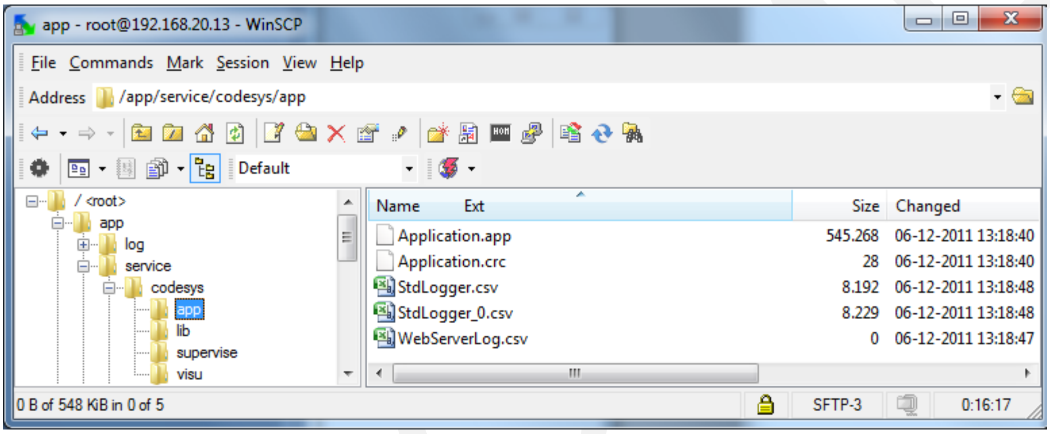
Address /tmp

proc
root
sbin
srv
sys
tmp
fwupdates
usr
var

Name	Ext	Size	Changed
fwupdates			06-12-2011 12:53:03
Update_success		0	06-12-2011 12:53:08

0 B of 0 B in 0 of 2 SFTP-3 0:15:30

The application.dupdate unpacks it self to `/app`:


app - root@192.168.20.13 - WinSCP
File Commands Mark Session View Help
Default

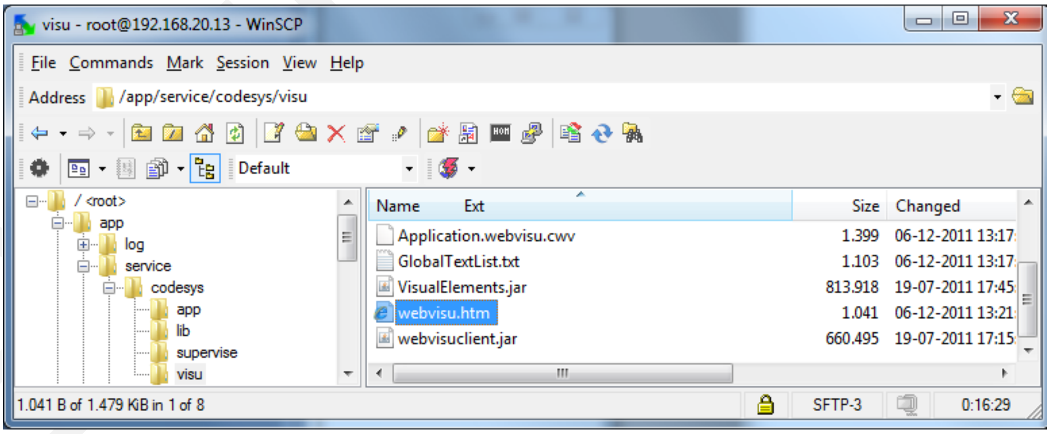
Address /app/service/codesys/app

/ <root>
app
log
service
codesys
app
lib
supervise
visu

Name	Ext	Size	Changed
Application.app		545.268	06-12-2011 13:18:40
Application.crc		28	06-12-2011 13:18:40
StdLogger.csv		8.192	06-12-2011 13:18:48
StdLogger_0.csv		8.229	06-12-2011 13:18:48
WebServerLog.csv		0	06-12-2011 13:18:47

0 B of 548 KiB in 0 of 5 SFTP-3 0:16:17

and to `/visu`:


visu - root@192.168.20.13 - WinSCP
File Commands Mark Session View Help
Default

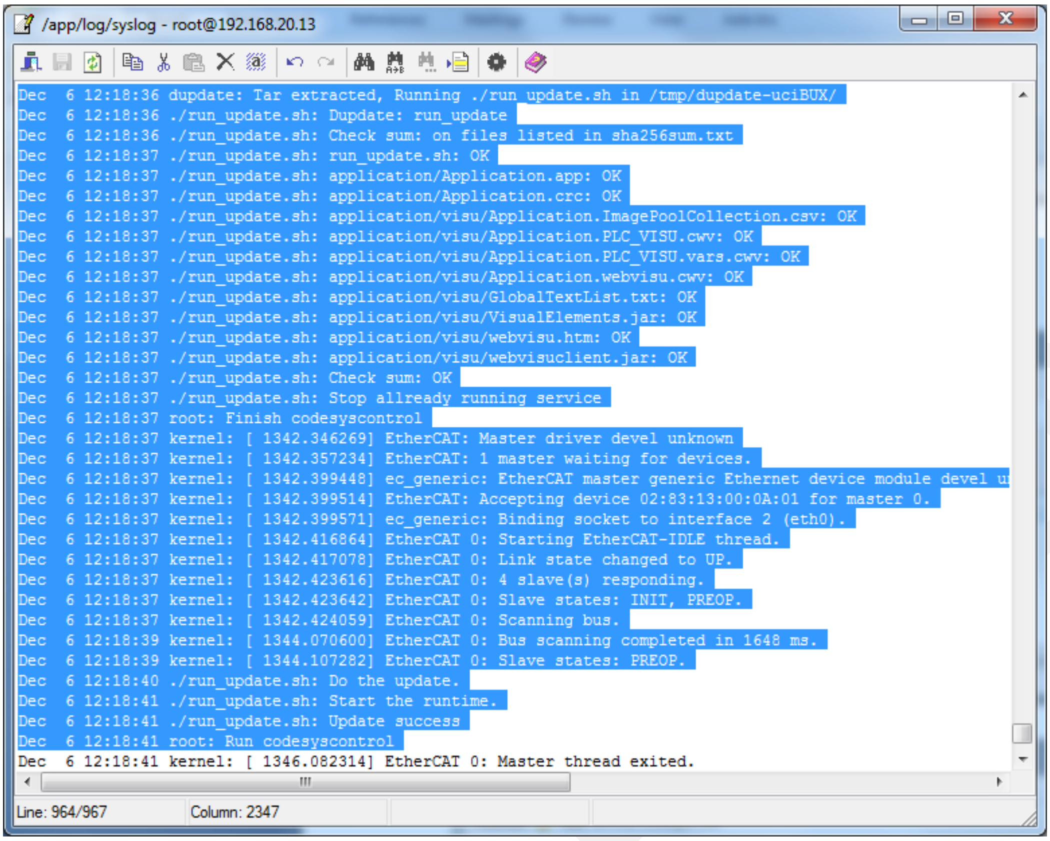
Address /app/service/codesys/visu

/ <root>
app
log
service
codesys
app
lib
supervise
visu

Name	Ext	Size	Changed
Application.webvisu.cwv		1.399	06-12-2011 13:17:
GlobalTextList.txt		1.103	06-12-2011 13:17:
VisualElements.jar		813.918	19-07-2011 17:45:
webvisu.htm		1.041	06-12-2011 13:21:
webvisuclient.jar		660.495	19-07-2011 17:15:

1.041 B of 1.479 KiB in 1 of 8 SFTP-3 0:16:29

Details about the update can be found in `/app/log/syslog`:



```
Dec  6 12:18:36 dupdate: Tar extracted, Running ./run_update.sh in /tmp/dupdate-uciBUX/
Dec  6 12:18:36 ./run_update.sh: Dupdate: run_update
Dec  6 12:18:36 ./run_update.sh: Check sum: on files listed in sha256sum.txt
Dec  6 12:18:37 ./run_update.sh: run_update.sh: OK
Dec  6 12:18:37 ./run_update.sh: application/Application.app: OK
Dec  6 12:18:37 ./run_update.sh: application/Application.crc: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/Application.ImagePoolCollection.csv: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/Application.PLC_VISU.cwv: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/Application.PLC_VISU.vars.cwv: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/Application.webvisu.cwv: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/GlobalTextList.txt: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/VisualElements.jar: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/webvisu.htm: OK
Dec  6 12:18:37 ./run_update.sh: application/visu/webvisuclient.jar: OK
Dec  6 12:18:37 ./run_update.sh: Check sum: OK
Dec  6 12:18:37 ./run_update.sh: Stop allready running service
Dec  6 12:18:37 root: Finish codesyscontrol
Dec  6 12:18:37 kernel: [ 1342.346269] EtherCAT: Master driver devel unknown
Dec  6 12:18:37 kernel: [ 1342.357234] EtherCAT: 1 master waiting for devices.
Dec  6 12:18:37 kernel: [ 1342.399448] ec_generic: EtherCAT master generic Ethernet device module devel u
Dec  6 12:18:37 kernel: [ 1342.399514] EtherCAT: Accepting device 02:83:13:00:0A:01 for master 0.
Dec  6 12:18:37 kernel: [ 1342.399571] ec_generic: Binding socket to interface 2 (eth0).
Dec  6 12:18:37 kernel: [ 1342.416864] EtherCAT 0: Starting EtherCAT-IDLE thread.
Dec  6 12:18:37 kernel: [ 1342.417078] EtherCAT 0: Link state changed to UP.
Dec  6 12:18:37 kernel: [ 1342.423616] EtherCAT 0: 4 slave(s) responding.
Dec  6 12:18:37 kernel: [ 1342.423642] EtherCAT 0: Slave states: INIT, PREOP.
Dec  6 12:18:37 kernel: [ 1342.424059] EtherCAT 0: Scanning bus.
Dec  6 12:18:39 kernel: [ 1344.070600] EtherCAT 0: Bus scanning completed in 1648 ms.
Dec  6 12:18:39 kernel: [ 1344.107282] EtherCAT 0: Slave states: PREOP.
Dec  6 12:18:40 ./run_update.sh: Do the update.
Dec  6 12:18:41 ./run_update.sh: Start the runtime.
Dec  6 12:18:41 ./run_update.sh: Update success
Dec  6 12:18:41 root: Run codesyscontrol
Dec  6 12:18:41 kernel: [ 1346.082314] EtherCAT 0: Master thread exited.
```

6 Using DEIF VisuLoader

6.1 Purpose of the VisuLoader

The DEIF VisuLoader application is an alternative to using a web browser to display the CoDeSys Web Visualization. It has two main purposes:

1. Enabling auto-reconnecting (in many cases) to the controller in case of connection loss due to either power reset of the AWC 500 or disconnection of the Ethernet.

6.2 Installing the VisuLoader

6.2.1 Java installation

If not already installed, install Java 7+ JRE on all panel PCs

6.2.2 Copy files

- Copy VisuLoader.jar to panel PCs `C:\visu\`
- Copy runVisuTower.bat to tower panel PC Windows startup folder
- Copy runVisuNacelle.bat to nacelle panel PC desktop

6.2.3 Edit files

- Edit runVisuTower.bat and change:
  - Controller IP address: e.g. `IP=192.168.20.13`
  - PLC address (hex): e.g. `PLCADDRESS=000D`
  - Visualization start page: e.g. `STARTVISU=aSYSTEMOVERVIEW`
- Edit runVisuNacelle.bat and change:
  - Tower panel PC IP address: e.g. `towerip=192.168.20.14`
  - Controller IP address: e.g. `IP=192.168.20.13`
  - PLC address (hex): e.g. `PLCADDRESS=000D`
  - Visualization start page: e.g. `STARTVISU=aSYSTEMOVERVIEW`

6.2.4 Configuring

Tower panel PC or single panel setup

It is recommended that the tower panel PC will have the runVisuTower.bat file located in both the windows startup folder (for automatic startup on panel PC power on) as well as on the desktop for easy manual startup.

Nacelle panel PC

This panel PC should have the runVisuNacelle.bat located only on the desktop for manual starting, since the visualization should only be active when servicing the Nacelle and people are physically present.

6.3 VisuLoader features

6.3.1 Automatic reconnection

The VisuLoader will try to reestablish the connection to the PCM5·1 in case of connection issues. In some special cases, reestablishing the connection is not possible and the VisuLoader will need to be manually restarted. These cases include if connection is lost during the initialization phase.

6.3.2 Tower Panel Lockout

In case the VisuLoader is opened on the Nacelle panel PC (if exists), the Visualization on the Tower Panel PC will be locked from interference by any user. This situation continues until the VisuLoader is closed on the Nacelle Panel PC.

6.4 Using the VisuLoader

6.4.1 Running the VisuLoader

On the Tower Panel PC the VisuLoader will automatically open when the panel PC is turned on.
On the Nacelle Panel PC the VisuLoader should be manually started by double clicking the "runVisNacelle.bat" on the desktop.

If needed the VisuLoader can be manually restarted by closing it using the button in the upper right corner and then started again by double clicking the "runVisu. . . bat" located on the desktop or in the windows startup folder.

6.4.2 General info

Note that it is not possible to do any action on the Tower Panel PC (including exiting the VisuLoader) as long as the VisuLoader is active in the Nacelle Panel PC.

DEIF A/S reserves the right to change any of the above.