CODESYS STORE

Data Sheet CODESYS Application Composer

The CODESYS Application Composer is a development tool for the efficient creation of applications consisting of recurring function blocks.

A 30 day demo license is available for testing. For the full version please contact CODESYS Sales.

Product description

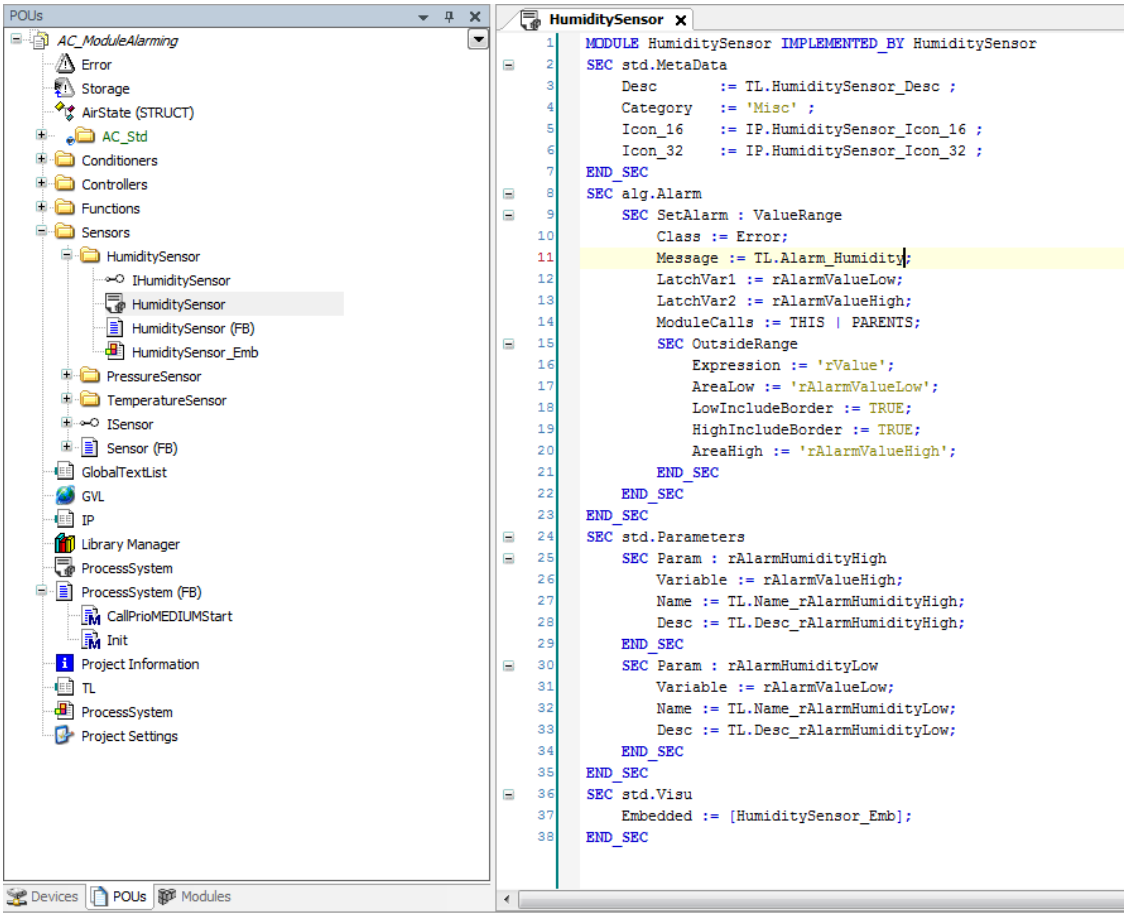
With the CODESYS Application Composer you can customize, i.e. compile and parameterize, complete control applications from previously created software modules.

Modules are functional program units that can correspond both to machine or system parts as well as software functions. Along with the program code, modules typically include components such as visualization elements, parameterization or I/O allocations. With this they cover nearly all programming aspects that CODESYS offers.

The customization of the modules takes place in the module tree. In the process each entry corresponds to a module instance. In the insertion of new elements into the module structure only suitable modules are offered for selection. In the module properties the parameterization, the I/O configuration and the visualization selection are defined for the module instances. Simultaneously the configuration of sequencer modules can take place with the help of an easy-to-use sequence editor. From the module configuration a menu command generates the complete application code including visualization and I/O configuration. Application-specific code can be added in the form of extension modules and remains unchanged if the code is generated again.

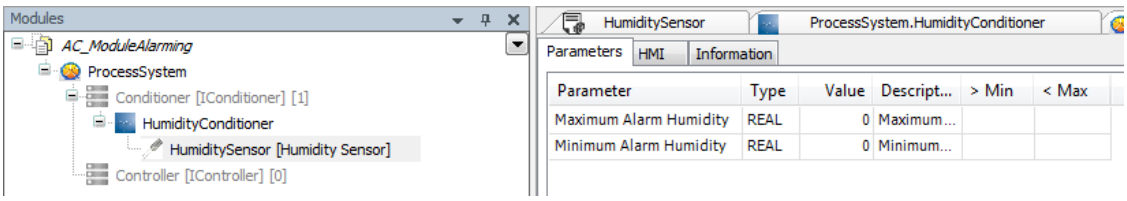
With purchasing a functional license you acquire the possibility to create and use new modules or module declarations within the CODESYS Development System.

The creation of modules takes place in the form of module declarations which can be added as objects in the POU pool (see Picture 1).



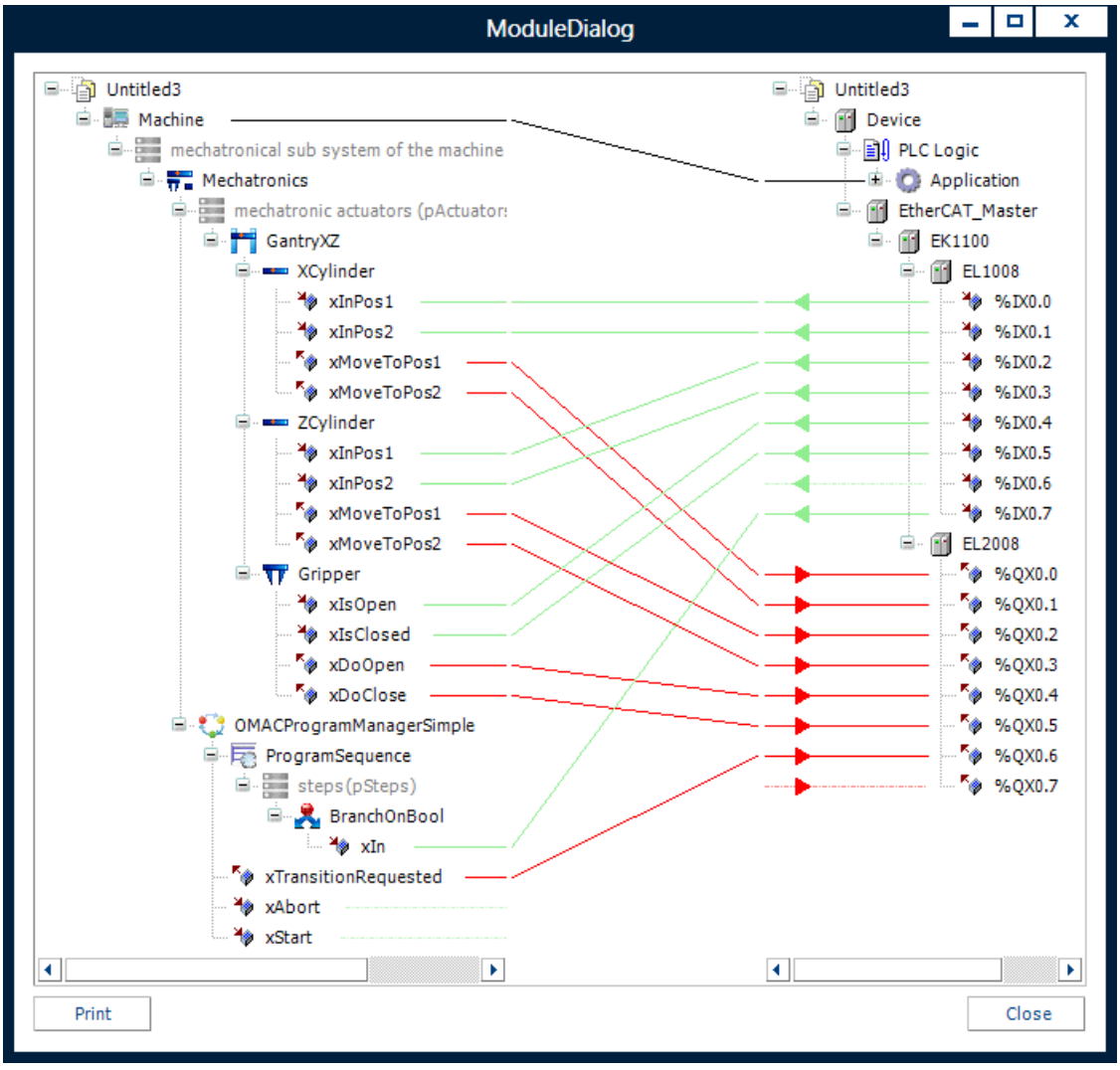
Picture 1: Module declaration

Each module declaration requires the specification of a function module (Modul-FB) as a basis, implementing the program functionalities of the module. Within the module declaration the module FB can be supplemented by additional properties and associated objects: * Parameters: Input variables of the module FB can be designated as parameters. Parameters are then conveniently configured within a parameter module editor (see Picture 2).



Picture 2: Parameter editor

- Inputs/Outputs: Input and output variables of the module FB can be defined as module inputs and outputs. Input variables of the module FB can be designated as parameters. The module inputs/outputs can then be conveniently connected to variables, other module inputs/outputs or device inputs/outputs (see Picture 3).

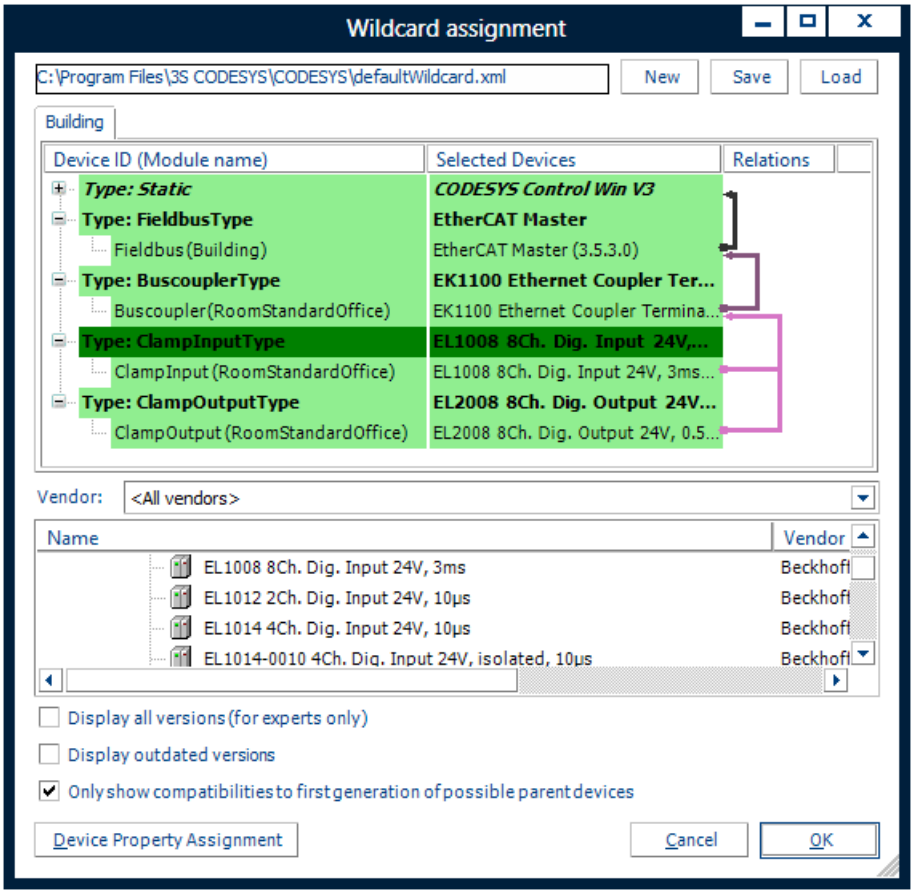


Picture 3: I/O editor

- Slots: Input variables of the module FB can be designated such that they can accept instances of other module FBs. For example, if a module is supposed to be usable as a sub-module beneath another module, a slot is defined in the father module that can accept the corresponding child modules. The associated input variable is then automatically filled with the instances of the child module FBs during the generation.
- Tasks: If it is a top level module (module without a father module), additional tasks can be defined that are generated in the generator run and that automatically call specified methods of the module FB. A top level module for its part then calls methods of its child module FB.
- Visualizations: Each module permits the definition of page and embedded visualizations. This will be

automatically generated with and connected to the module during generation. Page visualizations are displayed explicitly for one module. Embedded visualizations, on the other hand, can be embedded in the page visualizations of father modules.

- Proxy Module FBs: Modules can define so-called proxy representative FBs of their own module FBs. A proxy is used to make possible references on a module FB that go beyond the application and controller boundaries by creating a proxy FB in the respective application as a representative of the referenced module FB. The communication and data exchange between the module FB and its proxy FB below a foreign application is automatically generated by the Application Composer generators.
- Instance-References: With so-called instance references you can define FB instances that will not be identified until configuration time of the module by the module user with actual instances. For example device FBs can be referenced in modules as instance references.
- Devices and inputs/outputs: Modules can define devices (e.g. fieldbuses) that are inserted with the module into the device tree. The inputs and outputs of the devices can then be automatically connected to the inputs and outputs of the module or modules. For the greatest possible flexibility the devices can also be inserted as so-called “wildcards” which do not have to be filled with actual device types until the time of generation (see Picture 4).



Picture 4: Assignment of module channels to devices

- Alarms: Along with visualizations and devices, modules can also use the CODESYS alarm management in order to generate alarms for variables of their module FBs. These alarms can be displayed as well as intercepted and evaluated via a call mechanism.
- Sequence module designation: If modules in the form of sequences are supposed to be editable, this can be defined via the module declaration. Parameters, inputs/outputs or references can be specified which are then displayed directly in the sequence steps.
- Default Submodules: Default assignments and default configurations can be specified for module slots (see above). In the insertion of the module such a default assignment then fills the module slot automatically with the predefined module, which can be correspondingly configured. After the declaration of the modules and the implementation of the associated module FB, these modules can be inserted into the module tree and configured. In order to finally obtain a functional IEC application only a generator run has to be performed. Depending on the selection of generators and configuration, in the generator run the complete IEC code, visualizations, devices etc. are generated beneath an application. The entire generated code, or all objects created in the process can be freely viewed and edited under the device tree.

General information

Vendor:

CODESYS GmbH
Memminger Strasse 151
87439 Kempten
Germany

Support:

https://support.codesys.com

Item:
CODESYS Application Composer
Item number:
2101000006
Sales:

CODESYS Store
https://store.codesys.com

Included in delivery:
- License key

System requirements and restrictions

Programming System	CODESYS Development System 3.5.11.0 or higher
Runtime System	CODESYS Control Version 3.5.0.0
Supported Platforms/ Devices	Note: Use the project “Device Reader” to find out which functions are supported by the controller. “Device Reader” is available for free in the CODESYS Store.
Additional Requirements	-
Restrictions	-
Licensing	• Soft Key (Workplace-bound licensing, free part of all CODESYS products) • Optional: CODESYS Key (Increased security against loss of license keys, transferable licensing to other workstations)
Required Accessories	Optional: CODESYS Key

Note: Not all CODESYS features are available in all territories. For more information on geographic restrictions, please contact sales@codesys.com.
Note: Technical specifications are subject to change. Errors and omissions excepted. The content of the current online version of this document applies.