TM830

APROL Project Engineering

PERFECTION IN AUTOMATION
www.br-automation.com

Requirements	
Training modules	TM800 – APROL System Concept
Software	APROL LINUX
Hardware	1 control computer, 1 controller

Table of Contents

Table of Contents

1 Introduction

Project engineering represents the majority of the work needed to make a system operational. When doing this, it is important to list the amount of data involved in the project and the logic that must be implemented right from the start.

This is generally done according to a functional specification, which then becomes the software design specification after clarifying all of the options in APROL. The customer must take care of this with the project manager.

A successful engineering phase can then begin once this has been done.

Figure 1: Project Planning

1.1 Objectives

After completing this seminar, you will be able to implement a process control system project using the standard components provided by **APROL**. This applies not only to the system logics, but for the visualization applications, alarm systems and trend recordings as well.

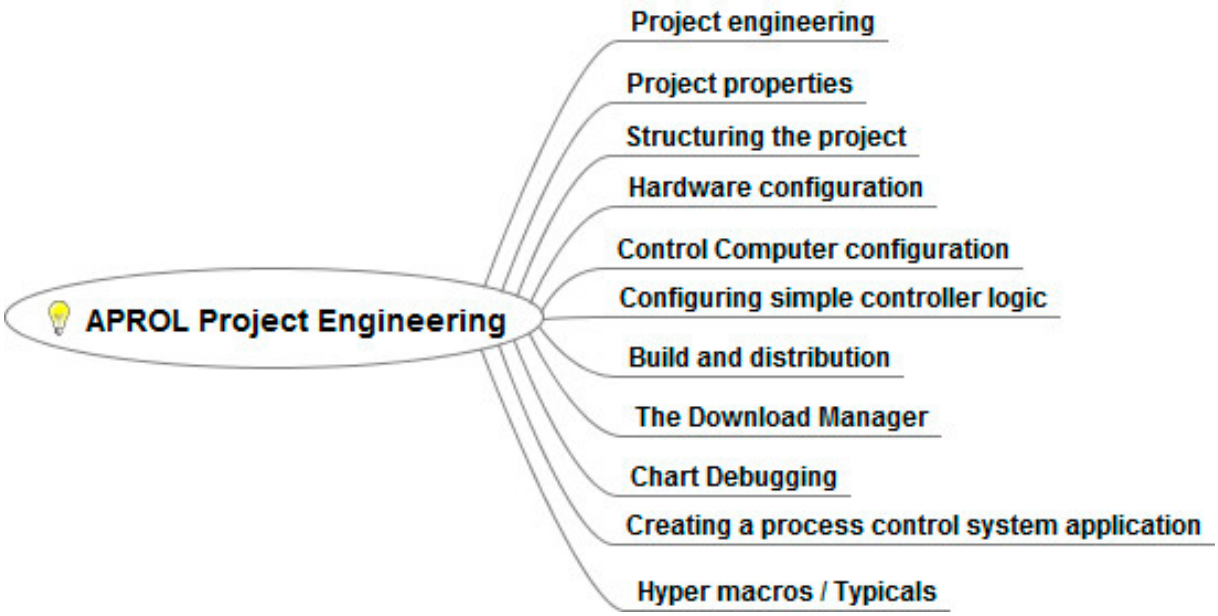


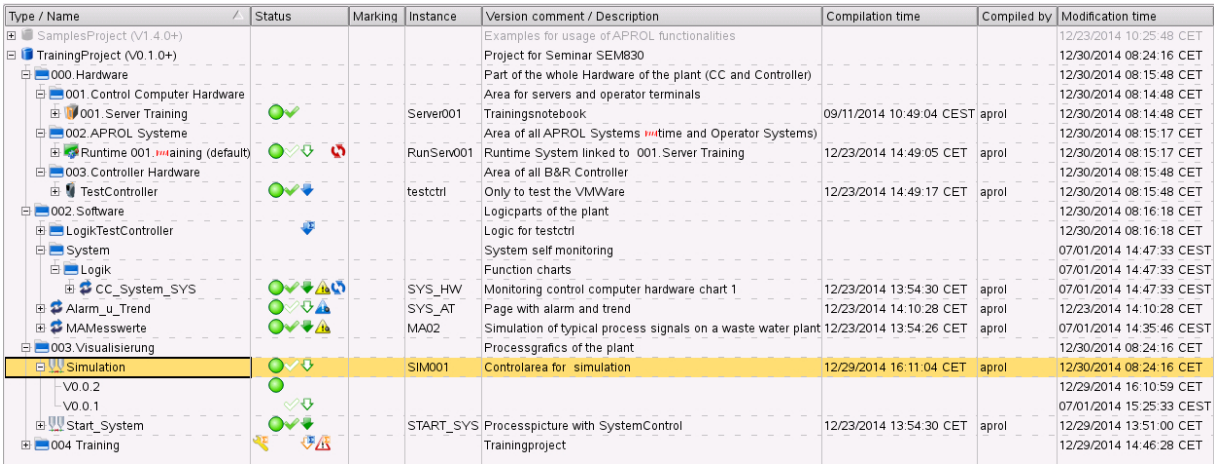
Figure 2: Overview

2 Project Engineering

Before beginning the project creation process, the project requirements need to be determined. This has to do with the project's size in relation to the number of operator stations, the number of I/O points and how they are structured. The operating philosophy with regard to the number of access rights, colors, and templates should have already been specified.

Once this framework has been established, it is possible to select the most suitable process control system components such as servers, operator stations, and controller hardware. The process bus, its topology, the number of controllers, and the structure of the I/O level can all be defined.

If the project configuration isn't known in advance, however, it is not really a problem since the system can always be expanded.



Type / Name	Status	Marking	Instance	Version comment / Description	Compilation time	Compiled by	Modification time
SamplesProject (V1.4.0+)				Examples for usage of APROL functionalities			12/23/2014 10:25:48 CET
TrainingProject (V0.1.0+)				Project for Seminar SEM830			12/30/2014 08:24:16 CET
000 Hardware				Part of the whole Hardware of the plant (CC and Controller)			12/30/2014 08:15:48 CET
001 Control Computer Hardware				Area for servers and operator terminals			12/30/2014 08:14:48 CET
001 Server Training			Server001	Trainingsnotebook	09/11/2014 10:49:04 CEST	aprol	12/30/2014 08:14:48 CET
002 APROL Systeme				Area of all APROL Systems (Runtime and Operator Systems)			12/30/2014 08:15:17 CET
Runtime 001 Training (default)			RunServ001	Runtime System linked to 001 Server Training	12/23/2014 14:49:05 CET	aprol	12/30/2014 08:15:17 CET
003 Controller Hardware				Area of all B&R Controller			12/30/2014 08:15:48 CET
TestController			testctrl	Only to test the VMWare	12/23/2014 14:49:17 CET	aprol	12/30/2014 08:15:48 CET
002 Software				Logicparts of the plant			12/30/2014 08:16:18 CET
LogikTestController				Logic for testctrl			12/30/2014 08:16:18 CET
System				System self monitoring			07/01/2014 14:47:33 CEST
Logik				Function charts			07/01/2014 14:47:33 CEST
CC_System_SYS			SYS_HW	Monitoring control computer hardware chart 1	12/23/2014 13:54:30 CET	aprol	07/01/2014 14:47:33 CEST
Alarm_u_Trend			SYS_AT	Page with alarm and trend	12/23/2014 14:10:28 CET	aprol	12/23/2014 14:10:28 CET
MAMesswerte			MA02	Simulation of typical process signals on a waste water plant	12/23/2014 13:54:26 CET	aprol	07/01/2014 14:35:46 CEST
003 Visualisierung				Processgrafics of the plant			12/30/2014 08:24:16 CET
Simulation			SIM001	Controlarea for simulation	12/29/2014 16:11:04 CET	aprol	12/30/2014 08:24:16 CET
V0.0.2							12/29/2014 16:10:59 CET
V0.0.1							07/01/2014 15:25:33 CEST
Start_System			START_SYS	Processpicture with SystemControl	12/23/2014 13:54:30 CET	aprol	12/29/2014 13:51:00 CET
004 Training				Trainingproject			12/29/2014 14:46:28 CET

Figure 3: Example extract from a project

3 Project Properties

The **'Project properties'** dialog opens when you create a new project with the **'File / New'** menu item. There are further subgroups such as 'Project master data, Engineering partner, Libraries, Engineering options, Download options, Communication Parameters, Runtime options, Print options and Documentation'.

One opens the project properties by clicking the right mouse button on the project directory and selecting the **'Properties'** menu entry.

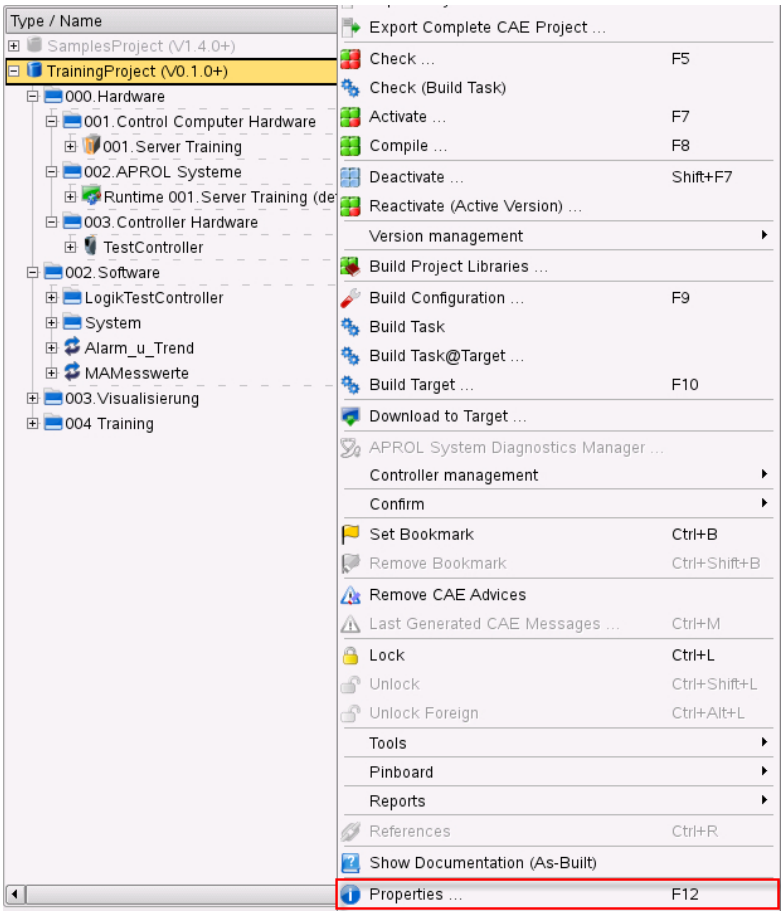


Figure 4: Project properties

Project master data

The first tab is generally used to enter and manage your project's master data. The majority of entries are simply used for documentation purposes. This makes it possible to store data such as your company's information and logo. This information appears later when printing out function charts. This data is also used in the 'User management' (ENGIN) and 'OperatorManager' (RUNTIME). This information is also used here to sign own certificates.

This data provides information about the creator of the project or his customer. In turn, this provides the creator with documentation that has the look and feel of the respective company.

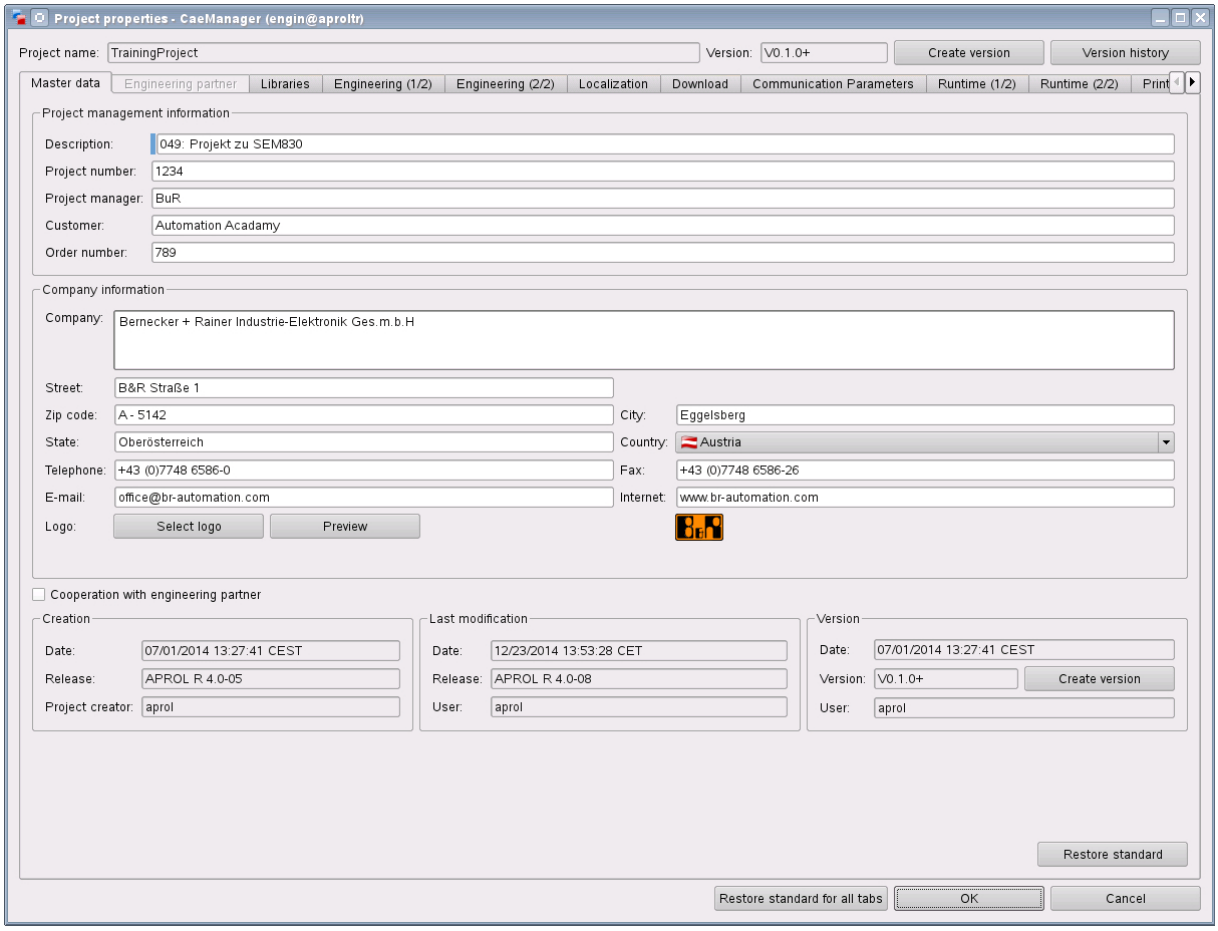


Figure 5: Company master data

One can also set the **'Co-operation with the engineering partner'** flag. This opens a new window (Engineering Partner) where data concerning this partnership can be added as needed.

Libraries:

This tab contains the libraries that are part of the project. Dependencies between libraries are also displayed. When new libraries are created (or when existing libraries are imported), they also have to be linked here in order to use them in the project.

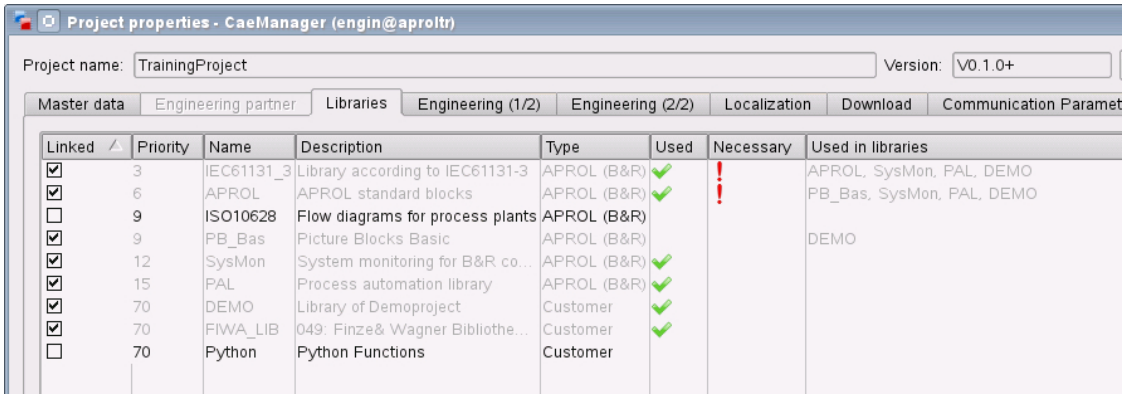


Figure 6: Project properties: Library dependencies

The following libraries are delivered with **APROL**.

- **APROL** contains helpful blocks, which differ to the IEC library, to configure the **alarms**, **trend system,** etc. in **APROL**.
- **IEC61131_3** contains all **IEC 61131-3 function blocks**.
- **SysMon** System monitoring for controllers and control computer
- **PB_Bas** contains graphic blocks with basic functions for visualization elements
- **ISO10628** contains flow sheets for process control systems
- **PAL** Process Automation Library (contains **hyper macros** and **typicals** such as valves, motors, controllers, signal conditioning ……)

Note: Libraries

In **APROL**, it is possible to create a number of additional project or branch-specific libraries. The project manager can then compel all other project users to only use blocks and picture macros from project-specific or customer-specific libraries. A high level of security is provided because the selected libraries are also protected from manipulation using access rights and passwords. Among other things, it is these features that support the system developer to validate a system.

Note: Operation

An operator template or faceplate is a window connected to the process graphic that enables the operation of individual units or equipment. The term originates from the early days of process control technology when attempts were made to imitate the faceplates of hardware controllers. The step up from the discretely built operator levels (mosaic circuit diagrams) to a control system was meant to be simplified for the system operators by reproducing as much previous knowledge as possible.

Engineering options:

Engineering options are used to manage the configuration process. The justification possibilities are explained in their respective **tool tips**.

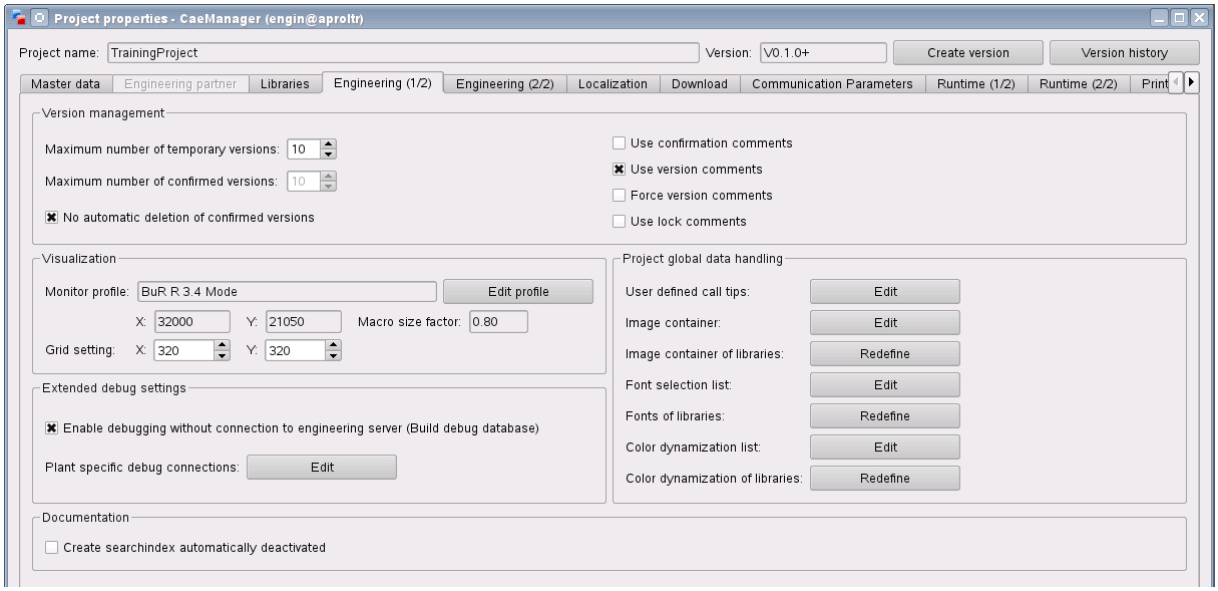


Figure 7: Project properties / Engineering Options 1

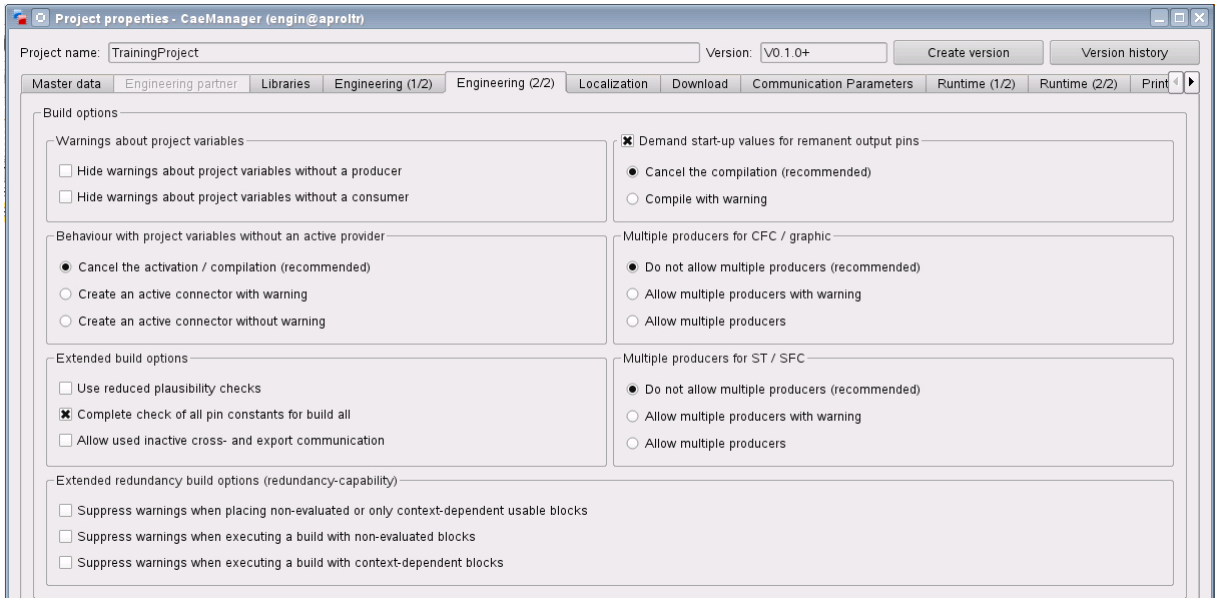


Figure 8: Project properties / Engineering Options 2

They allow the properties and settings of the following subgroups to be configured:

- **Version management**
  Any change to an object requires a new version for that object.

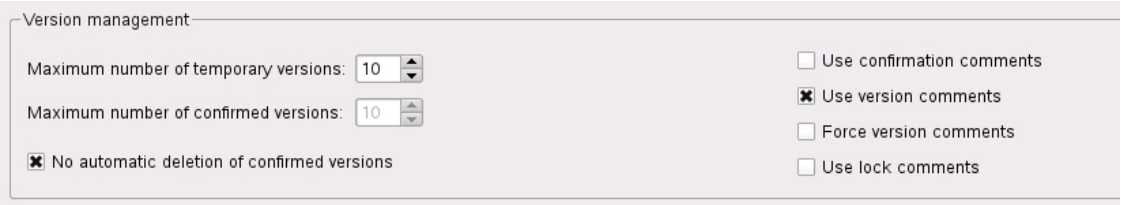


Figure 9: Version management

Note: Confirmation of versions

"Confirmation" is not automatically handled by the system; the user must do this manually. This allows a user with the appropriate authorization to identify a version as "correctly functioning". Versions, which are marked as such and (even better) are also labeled as such, should not be deleted.

- **Build options**
  Build options make it possible to control what happens when the code is compiled. In addition, it is possible to specify which warnings are or are not shown in the project tree.

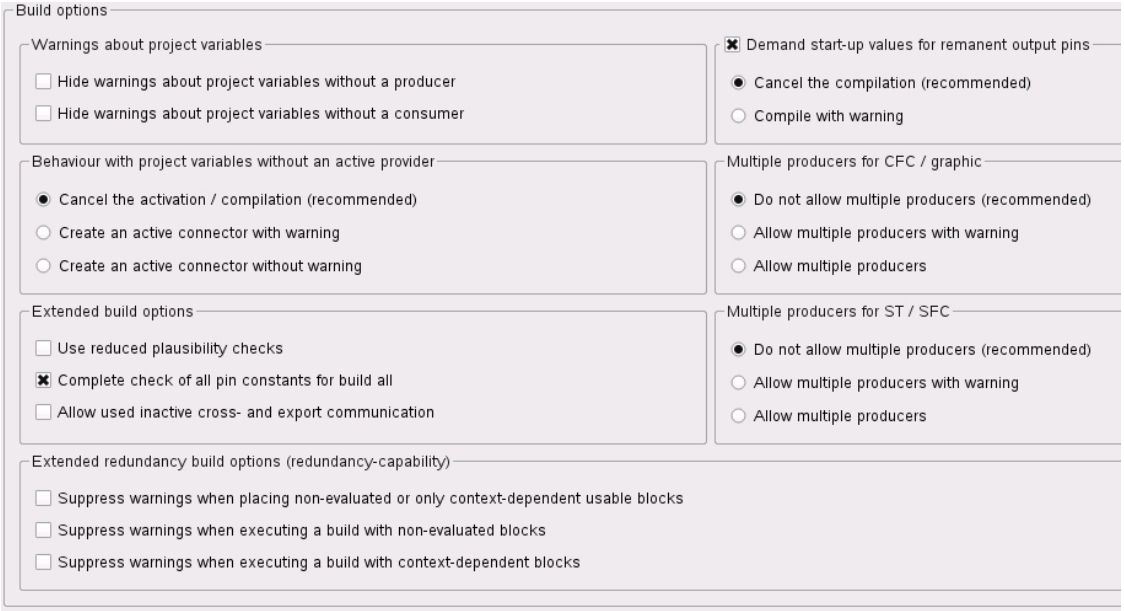


Figure 10: Engineering properties: Build options

If the flag "Create task-local variables for debugging new function charts" is enabled, then the online debugging system will also display variables that would not normally be transferred via the process bus. Although this function helps the designer locate errors, it adds a little more load to the process bus and should therefore only be used if absolutely needed.

Note: Debugging

Debugging represents one of the greatest differences between a programming device which has access to all memory locations on the controller via a programming device interface and a process control system which only uses its process bus to transfer the data required by other control system components via its process bus. "Function chart debugging" can be used in APROL to display all values used in a function chart.

- **Visualization**
  The visualization area allows you to define the resolution for displaying process graphics. (Topic 4:3 or 16:10 …..) One can create and manage own profiles.

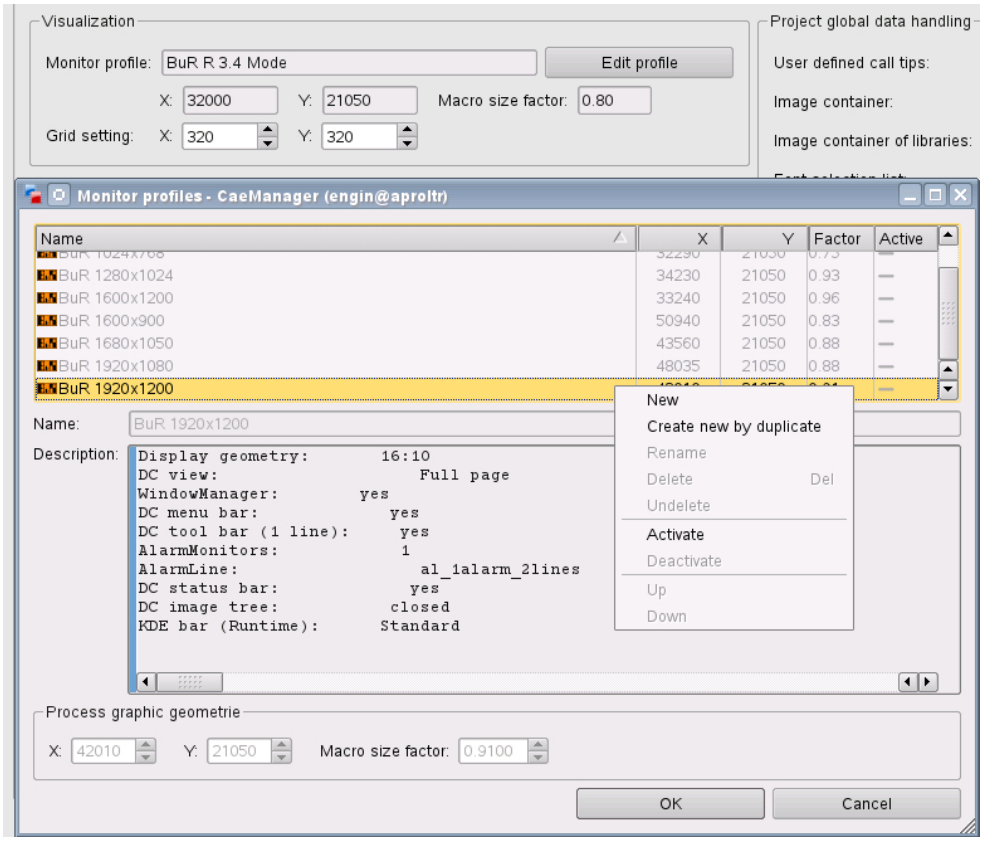


Figure 11: Profile monitor

- **Project-global data handling**
  This is where texts and images can be manipulated and managed.

- **Multiple producers for CFC / Graphic / ST / SFC**
  This allows variables to be written to several times. This naturally needs to be taken into consideration in the application itself. (who wins?)

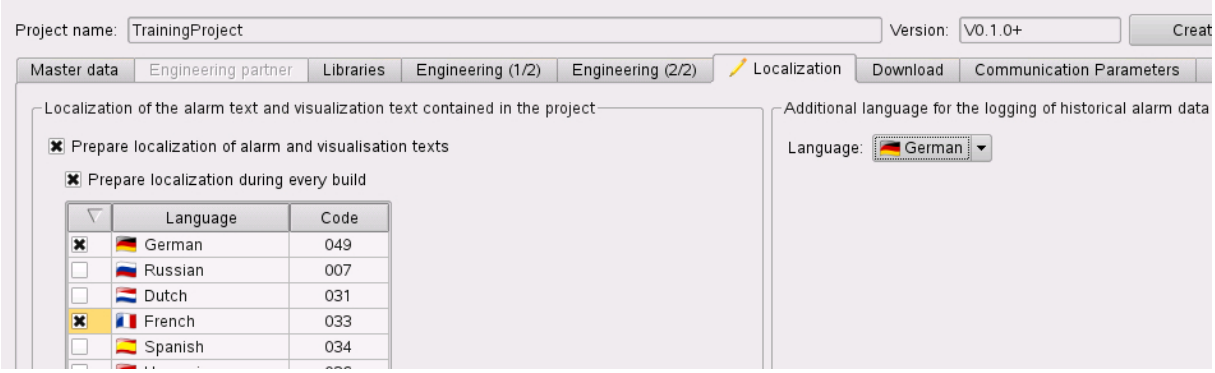
- **Parameter management**

Checks the parameters with regard to IEC types.

- **Documentation**
  Controls the creation of the "search index".

Localization:

APROL provides multilingual support for projects (Unicode-capable). It is therefore necessary to include translations for the texts used in the visualization as well as for the alarm system. The translations themselves can then be taken care of in the TranslationManager.



All setting for the multilingual support of the project are made in this tab. If the project is created afterwards, the visualization and alarm system texts will be extracted.

Download options:

Download options make it possible to define interactions, auto-starts, recovery points, and much more in relation to the DownloadManager.

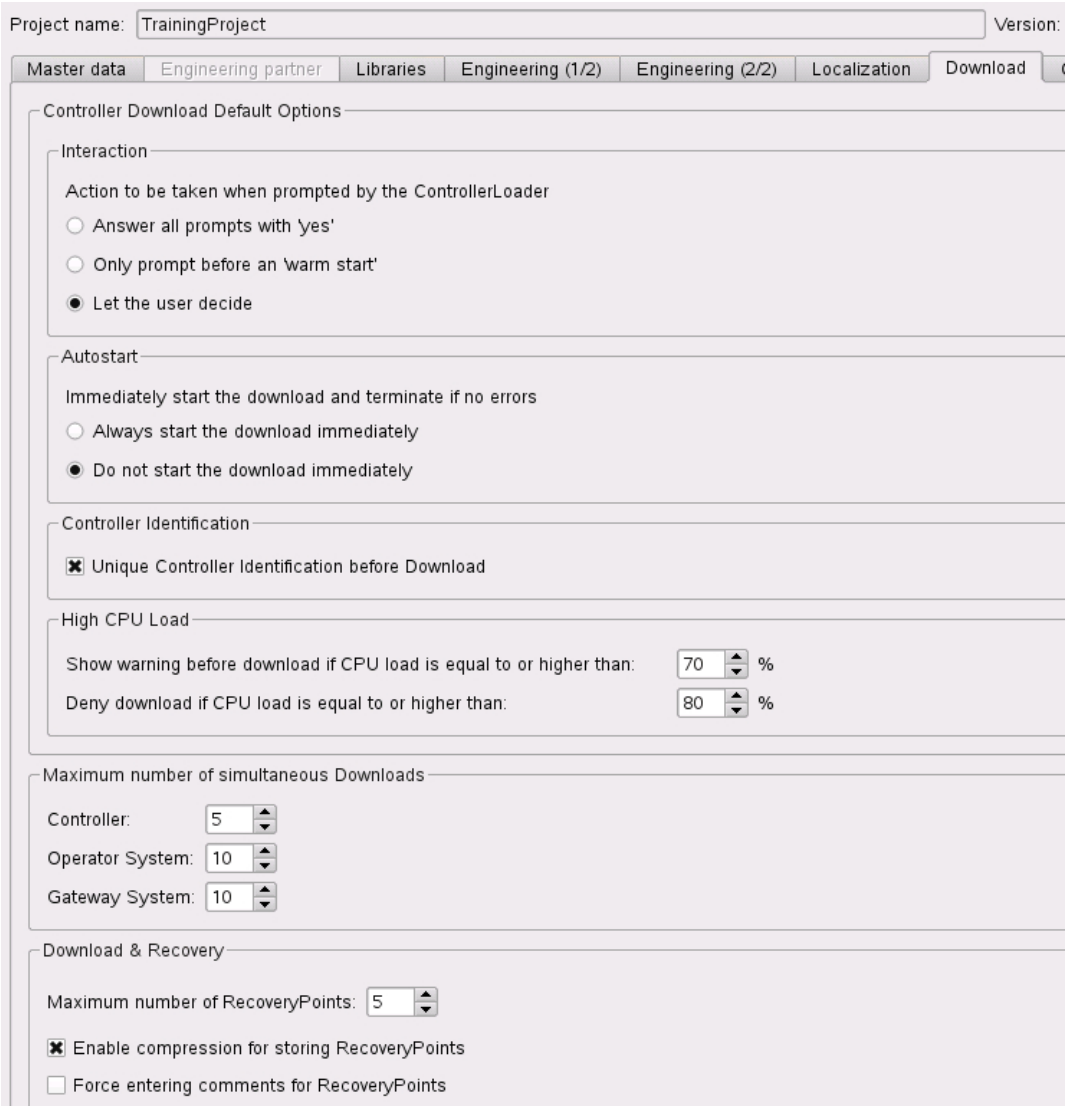


Figure 12: Download options

Communication parameters:

These settings are present because of compatibility reasons. The current project used by B&R is ANSL (Automation Net Service Link). This is a TCP/IP-based communication.

The communication was realized via INA (Industrial Network Architecture) previously.

Each controller on the network is assigned an INA node number (rotary switch on the controller) that is used for communication. Every INA service (InaDriver, ControllerManager, ControllerLoader, etc.) receives a virtual INA node number. These INA node numbers must not collide whenever a connection is established. The communication parameters make it possible to manage and reserve these node numbers.

An ANSL management is therefore not mandatory.

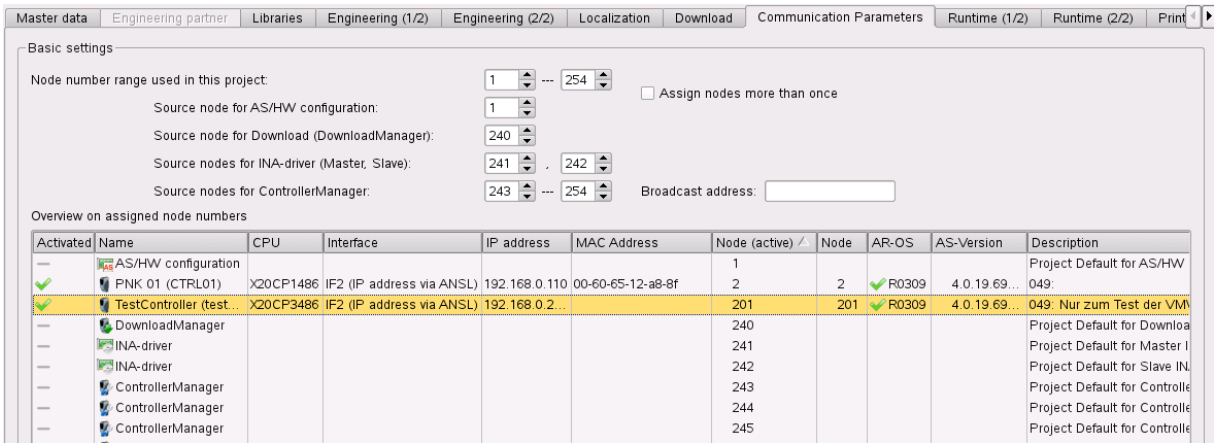


Figure 13: Communication parameters

Runtime options:

Many runtime options allow you to tailor the system as desired.

- **Authentication:**
  The user management can be used to for the reports integrated in **APROL** to specify whether working with these applications is allowed or not. This verification requirement can be enabled or disabled globally in the project options. This is of relevance for remote access.

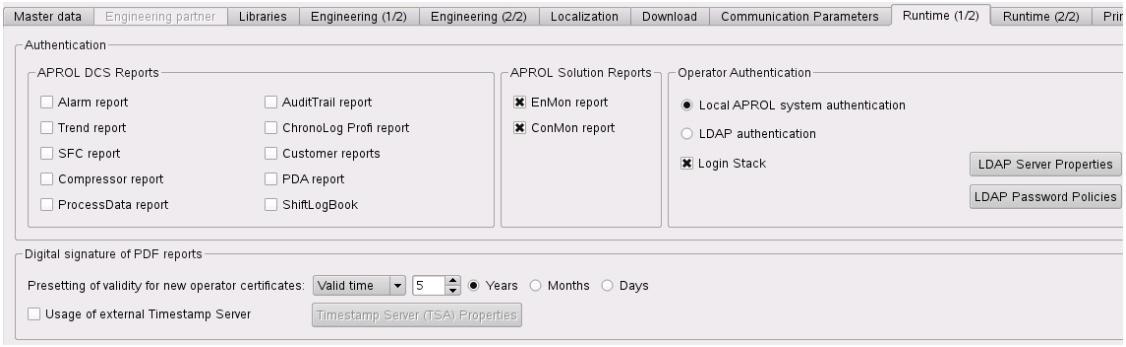


Figure 14: Runtime options: Authentication

- **Global settings for visualization**
  This is where options concerning switching actions, tooltips, the overall color environment and highlighting can be managed.

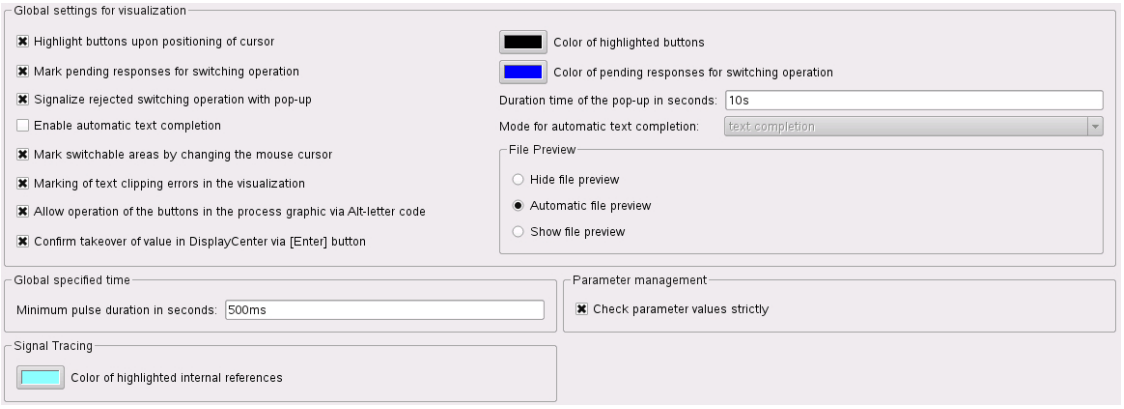


Figure 15: Visualization settings

- **Parameter management**

Figure 16: Strict check of parameter values

- **Global settings for SFC**
  Configuration settings of the background and object colors in the SFCViewer.

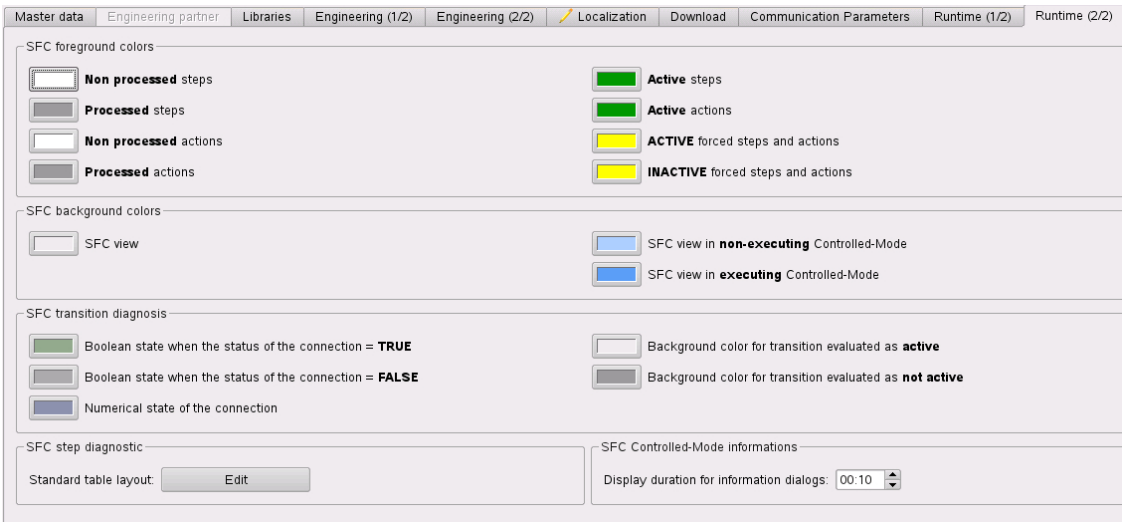


Figure 17: SFC settings

Print options:

Contains the default settings for printing function charts as well as configuring multiple print jobs.

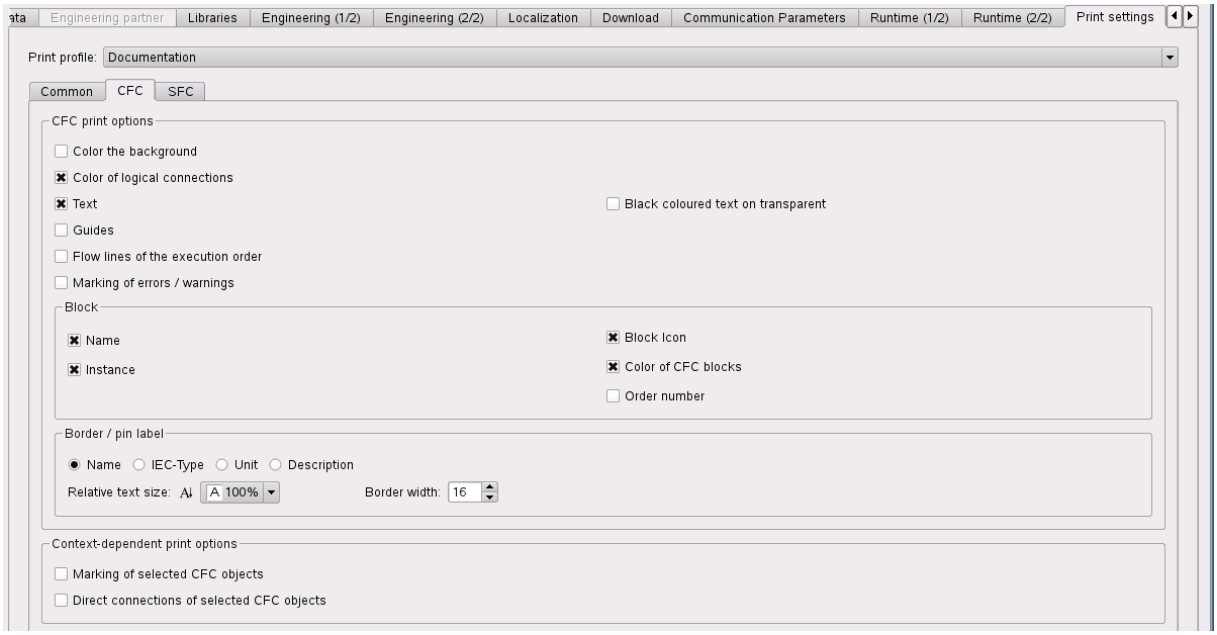


Figure 18: Printing settings for function charts, etc.

Documentation:

A project must always be documented appropriately. To aid with this, the project's documentation header can be created here. Afterwards, a description can be given for each project part being worked on in the CaeManager's project engineering. All of these entries put together then comprise the automatically generated project documentation.

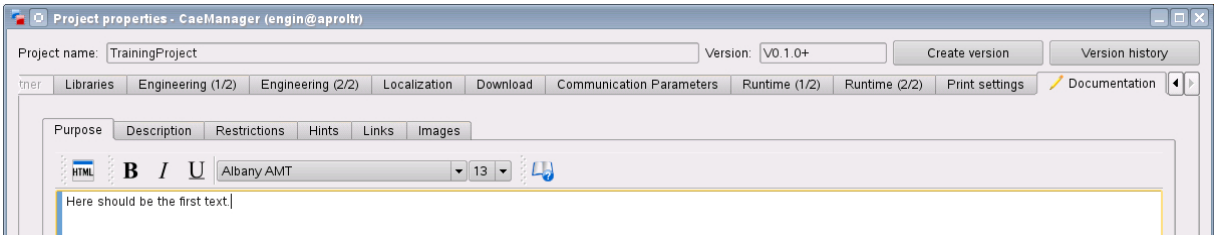


Figure 19: Documentation header for the project

Basic project settings for users

Basic project settings made by members of the project are done with the **"Extras / User Options"** menu item.

These basic project settings have to do with configuration options for the user who is currently logged in. These are the settings for handling the CaeManager (queries, menu guidance, etc.). These definitions are always loaded when a user logs in. This allows confirmation messages used by the system to be adapted to whoever is logged in.

Several different configuration possibilities are the result. Detailed information can be obtained from the tool tips or the **APROL** online help..

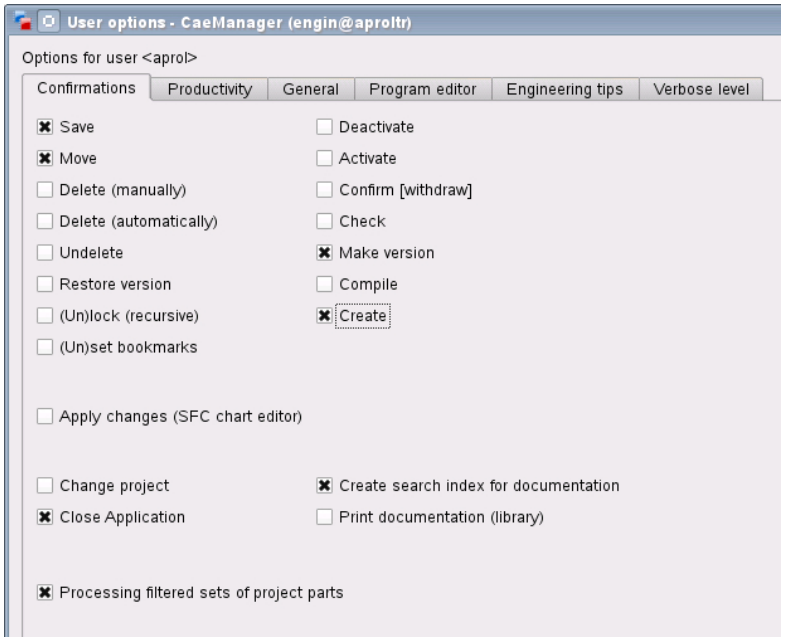


Figure 20: Options for the user

Depending on the settings in the **"Productivity"** frame, you can accelerate your work on the project sections and save the effort of having to execute extra commands in the CaeManager. This should be taken into account at the beginning.

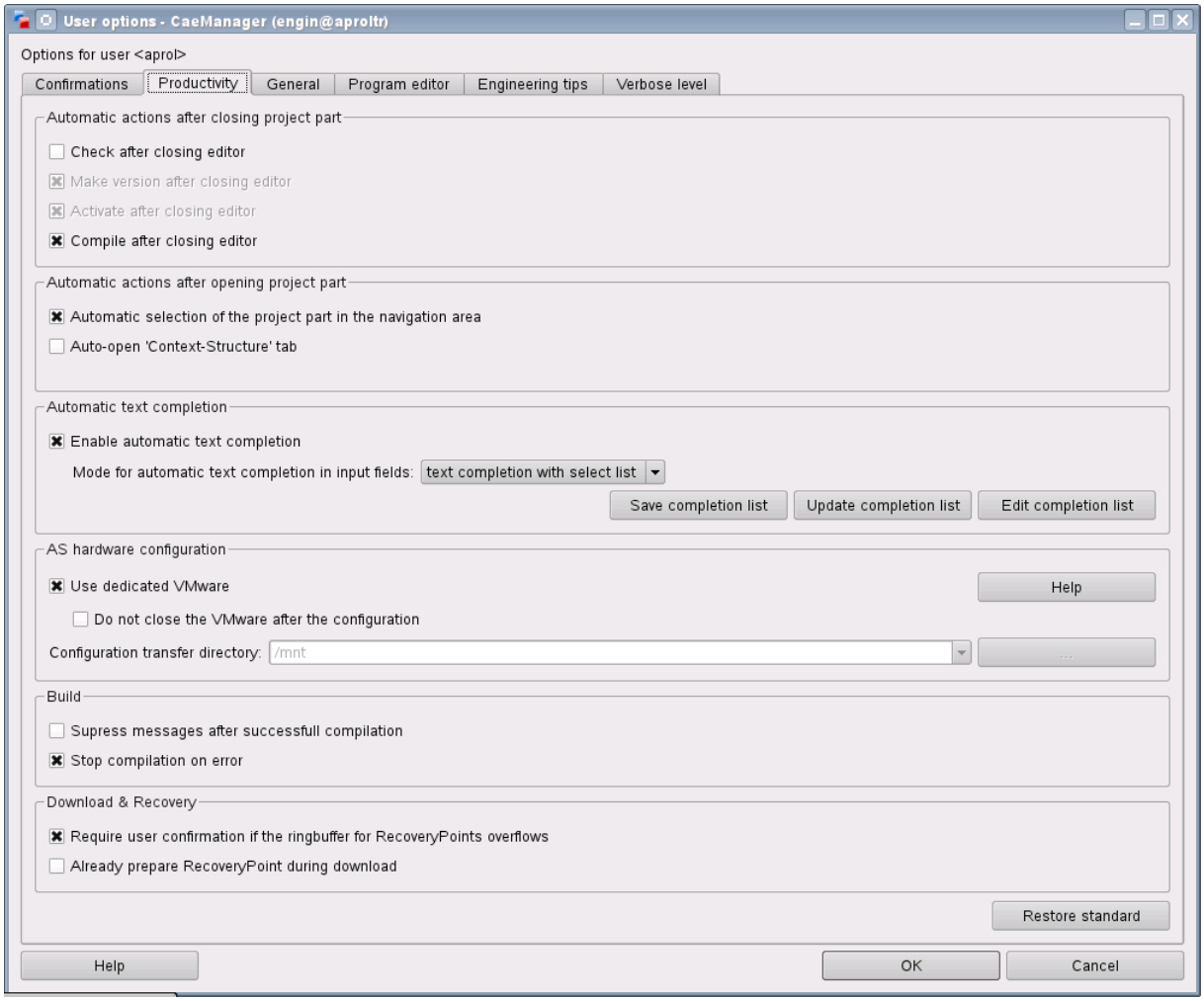


Figure 21: Productivity

- The behavior of the CaeManager can be controlled by activating the individual checkboxes in the section "Automatic actions after closing project part".
  If either "Make version after closing editor", "Activate after closing editor" or "Compile after closing editor" is selected, then the items which precede that respective option are grayed out since they will be carried out automatically anyway.

- Automatic text completion during input
  This activates the automatic text completion. In addition, the current list of texts for auto-completion can be saved or merged with another saved list.

- AS hardware configuration
  This setting makes it easier to deal with the AS if it is used regularly.
- Build
  Settings for code creation (e.g. let everything run and check afterwards)

- Download & Recovery
  Options for organizing recovery points and downloads.

General CaeManager settings can be changed in the **"General"** section. Detailed information about the individual options can be found in the respective tool tip and in the **APROL** online help.

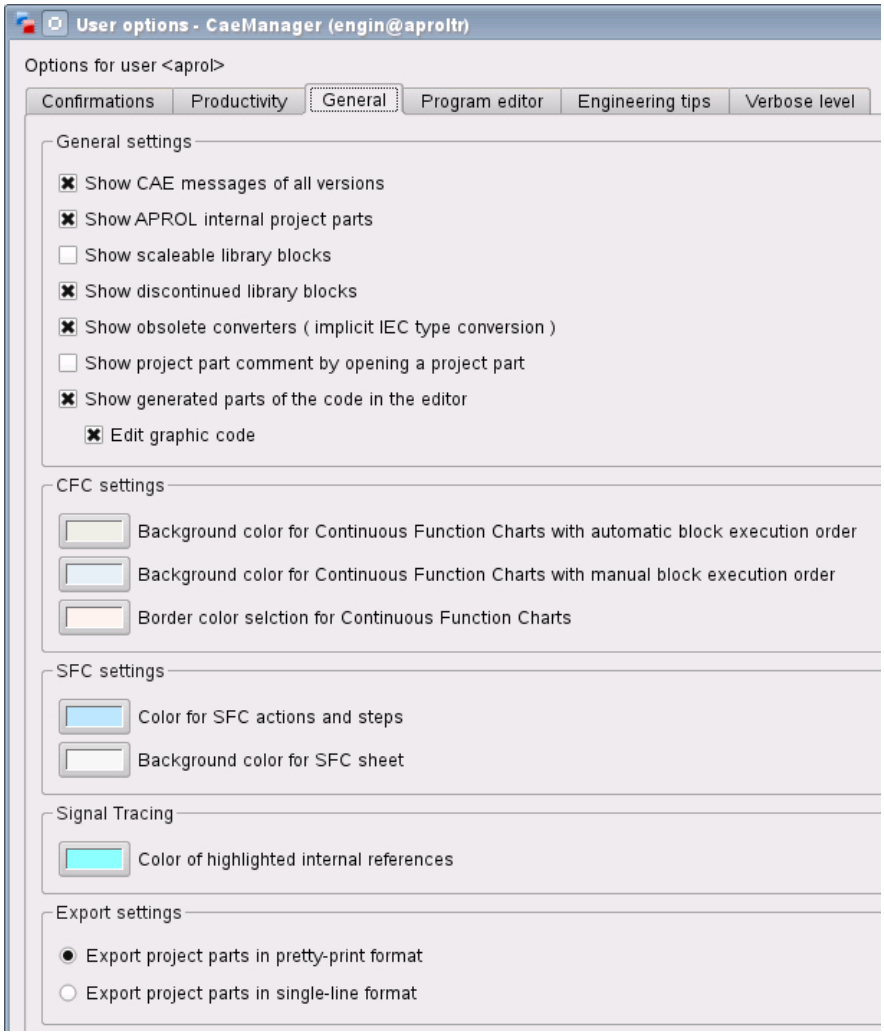


Figure 22: General information

APROL contains many **"Engineering tips"**. These can be enabled and disabled. It is recommended to allow these tips in the initial phase of project engineering, because they provide some very useful information. All tips can also be viewed here.

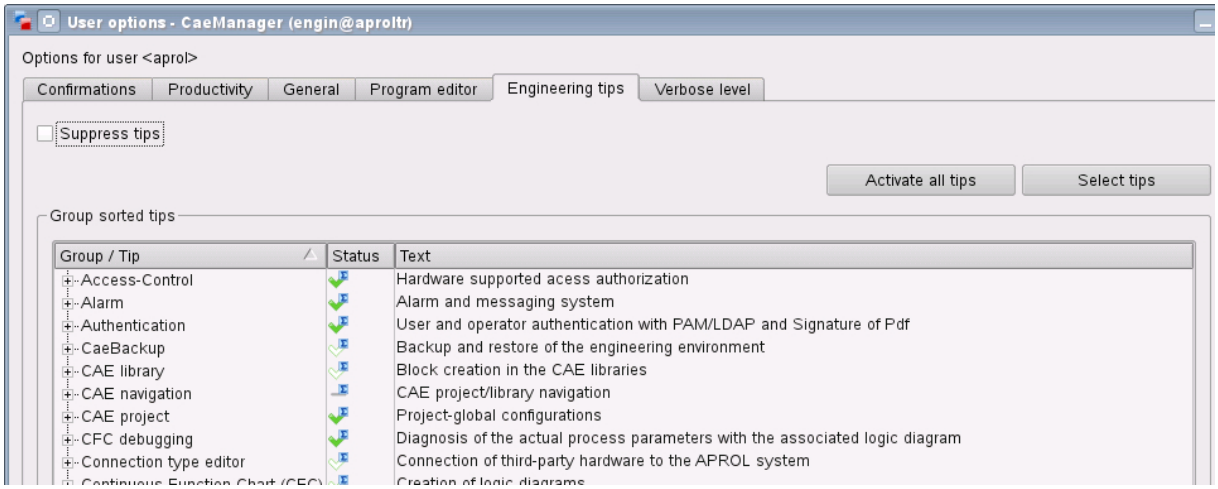


Figure 23: Engineering tips

The **Verbose Level** frame affects the output when compiling or executing a build. The higher the level selected, the more detailed the output becomes (verbose = wordy, long-winded, talkative).

Note: Error output

Error messages are **not** suppressed by the level of verbosity!

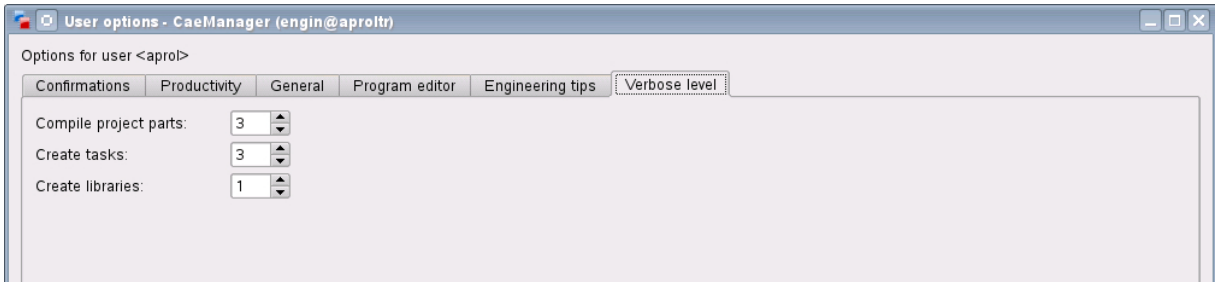


Figure 24: Verbose level

4 Structuring the Project

Projects should be carefully structured so that they can be read and understood by all users. **APROL** allows directories to be "nested" inside of others for a clear structuring of projects.

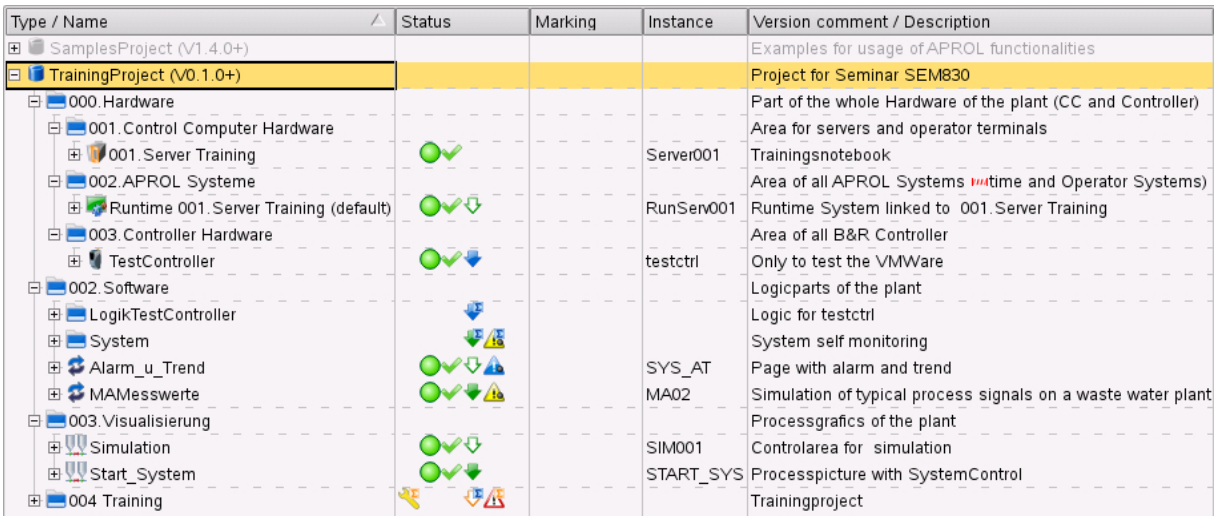


Figure 25: Project structure

4.1 Inserting Directories

Select the area where a new directory should be inserted. The dialog opens by selecting the **"File/New/ Directory"** menu item or with a right click on the desired position in the CaeManager.

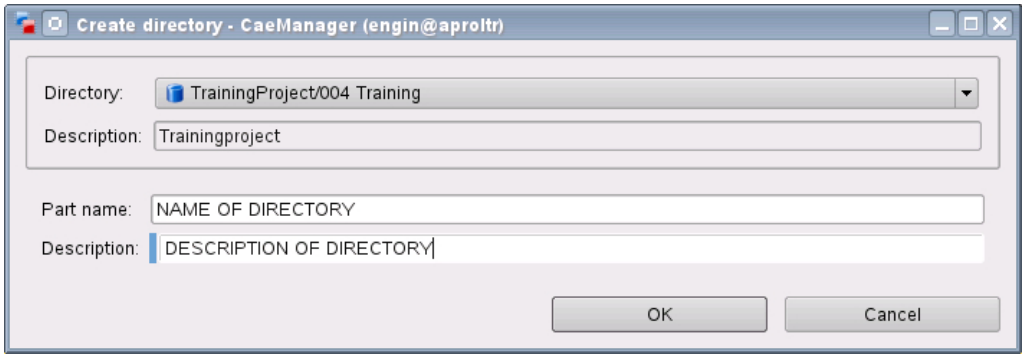


Figure 26: Creating a directory

Normally, this is a structure where the respective programming logic for the controller hardware is stored. Tasks that have a name identical or similar to the directories are created in the hardware engineering area.

Of course, the respective engineering partner ultimately decides how the project will be structured. The "SamplesProject" and "Demoproject" can serve as examples. They are supplied and can be found in the path "/opt/aprol/ENGIN/DEMOPROJECTS/".

5 Hardware Configuration

Now it is time to configure the controller. This will only be a representation of the framework in the CaeManager since this framework was already defined at the start. Use the "Import options" to make this easier.

5.1 Inserting a Controller

A new controller can be added to the selected directory by right-clicking with the mouse. An identical dialog is also shown with the **„File/New/Controller“** menu.

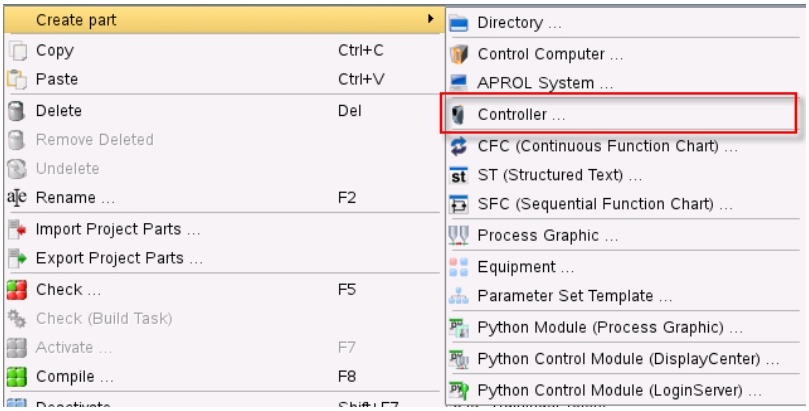


Figure 27: Creating a controller

The controller must be given a unique name, a description, and a unique instance in the following dialog box. Validators prohibit syntax errors.

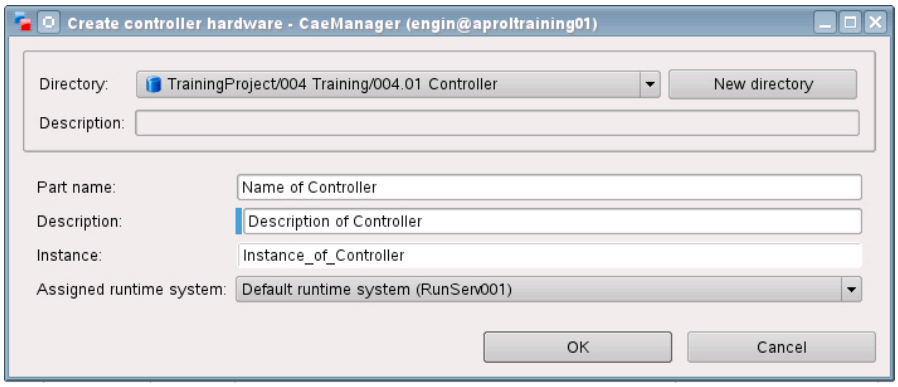


Figure 28: Creating a new controller

5.2 Configuring the Controller

A controller can be selected in the tree once it has been added. Double-clicking on the controller icon in the engineering area of the CaeManager (configuration area) opens up the hardware engineering area.

5.2.1 Starting the Hardware Configuration in Automation Studio

The hardware configuration can be started with the **„AS/HW configuration“** function. An Automation Studio now starts in a VMware.

Note: Leaving VMware

Pressing **<Alt>+<Ctrl>** followed by **<Alt>+<Tab>** allows you to temporarily leave VMware without having to exit it.

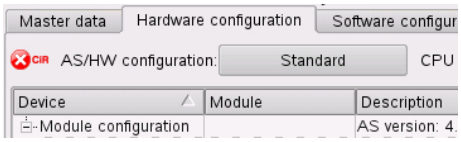


Figure 29: Starting the hardware configuration

5.2.2 Selecting the CPU Type

After VMware starts, a dialog box appears where the appropriate CPU type can be selected.

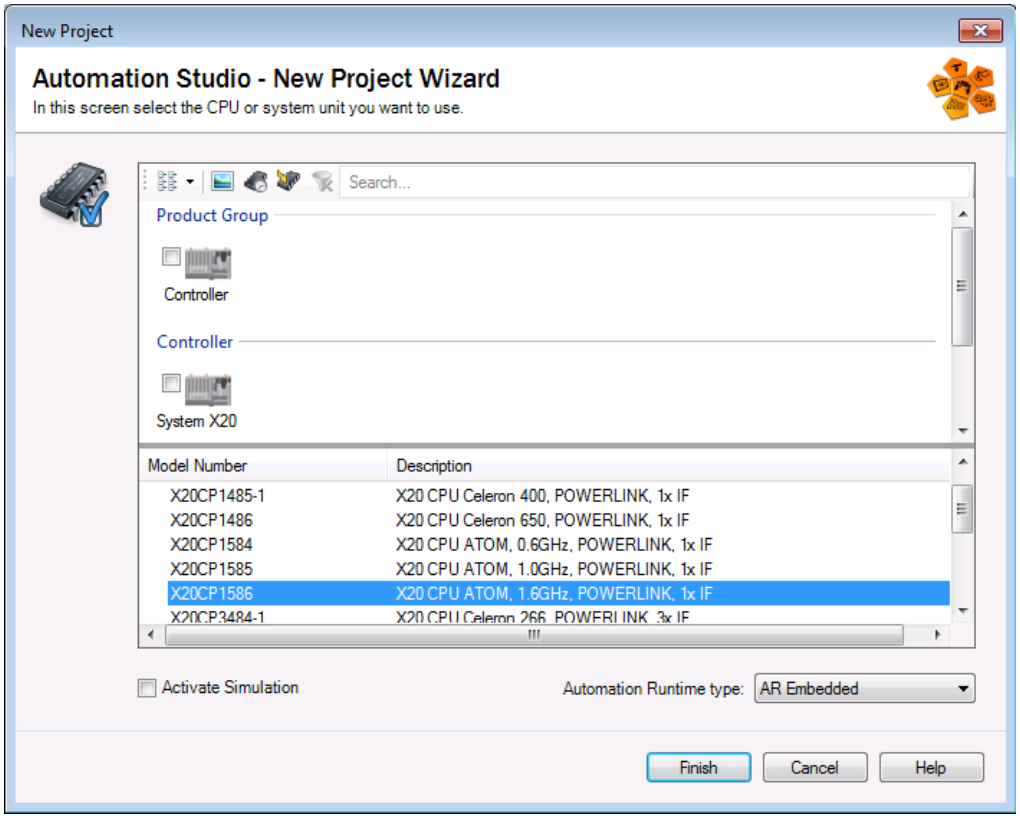


Figure 30: Selecting the CPU type

At this point it's also necessary to choose the Automation Runtime version to be used. Once the CPU has been selected, you will see the "Physical View" in Automation Studio.

Note: Changing the Automation Runtime version at a later time

The AR version can be changed at a later point in time with **Project / Change runtime version**. This results in a download of the new AR and the complete controller project.

5.2.3 CPU Properties

The shortcut menu for the CPU (right-click) is a jumping off point for managing several different configuration options:

- **Software:** Parameters such as memory, boot behavior, time synchronization, timing, etc. can be set here
- **Ethernet:** Ethernet slaves can be inserted here (SimDevice, Modbus TCP Slave)
- **POWERLINK:** Several POWERLINK and Ethernet modules can be inserted here
- **X2X Link:** Modules can be placed on the X2X bus (backplane bus) here
- **I/O Mapping:** The CPU link nodes are visible here
- **Configuration:** Host name, gateway, and DNS parameters can be specified here
- **IF1 Serial Configuration:** Serial interface parameters can be set here
- **IF2 Ethernet I/O Mapping:** The IF2 interface link nodes are visible here
- **IF2 Ethernet Configuration:** The IF2 interface parameters can be set here
- **IF3 POWERLINK I/O Mapping:** The IF3 interface link nodes are visible here
- **IF3 POWERLINK Configuration:** The IF3 interface parameters can be set here
- **IF6 X2X I/O Mapping:** The IF6 X2X interface link nodes (backplane bus) are visible here
- **IF3 X2X Configuration:** The IF6 X2X interface parameters (backplane bus) can be set here

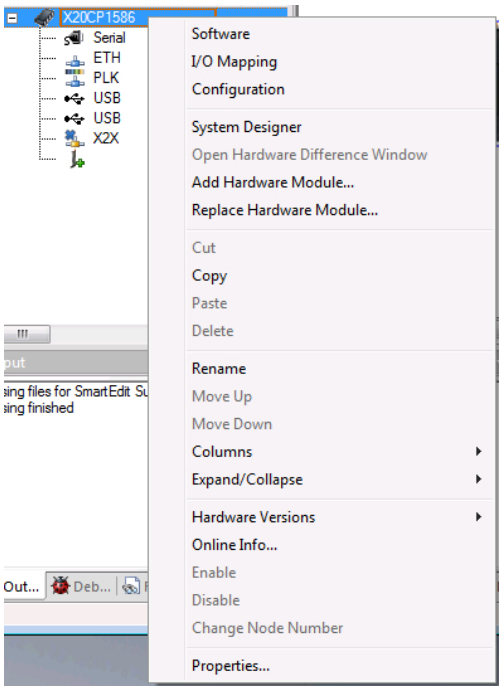


Figure 31: The CPU's shortcut menu

5.2.4 Placing Modules

Further configuration of the hardware (I/O module, POWERLINK network, etc.) takes place with the System Designer. This is an integrated part of Automation Studio and is opened with the **"Open/SystemDesigner"** menu. The graphical view of the hardware now appears. All supported hardware components can be selected in the right part of AS and placed in the graphical depiction via drag-and-drop. The network assignment is made simply by drawing lines.

The I/O mapping is configured later in the APROL engineering (CaeManager).

Details can be found in the Automation Studio help.

This is also explained in an Automation Studio seminar.

5.2.5 Module Documentation

Documentation pertaining to individual modules can be called up at any time to find out more about their technical details. This help documentation corresponds to information from the B&R hardware user manuals and can be started by selecting the desired module and pressing "F1".

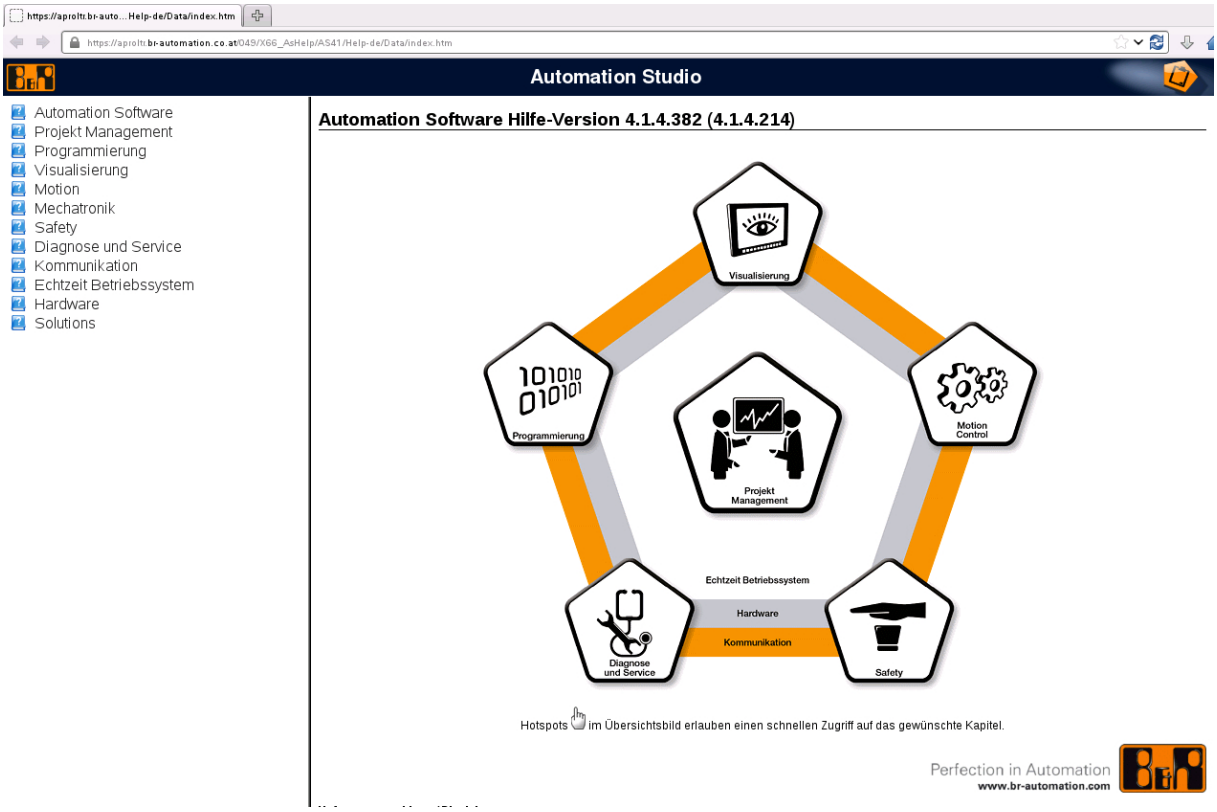


Figure 32: Hardware documentation

5.2.6 Module Properties

Once all modules have been placed, they can be configured. Module properties can be configured in the module context menu (right-clicking on the module) and selecting **"Configuration"**.

Of course, the data is different for each module. The technical data can be looked up in the corresponding module help files or in the **APROL** help system.

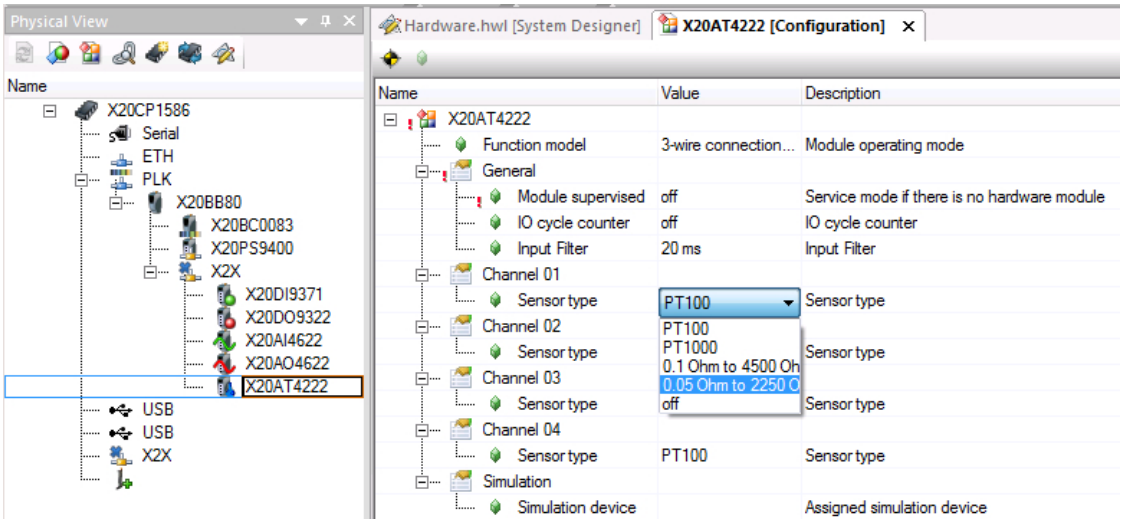


Figure 33: Example: AT6402 module configuration

The screenshot above shows an example of this. The configuration options are mostly self-explanatory.

5.2.7 Adding an Ethernet POWERLINK Rack

In most cases, the controller consists of more than just the master rack. POWERLINK racks are also usually connected. These are remote I/Os channels connected using Ethernet (Cat. 5e crossover cables). It is possible to use either a star (hub) or bus topology as well as a combination of both. Fiber optic cable should be used in conjunction with media converters for distances greater than 100 m.

On X20 CPUs, the IF3 interface can be used as a POWERLINK master. The respective parameters can be set in the PLK entry context menu with "Configuration". There are also POWERLINK interface modules available for use in subslots.

The configuration of POWERLINK is done with System Designer (drag-and-drop) and drawing the respective lines.

The I/O mapping is configured later in the **APROL** engineering in the CaeManager.

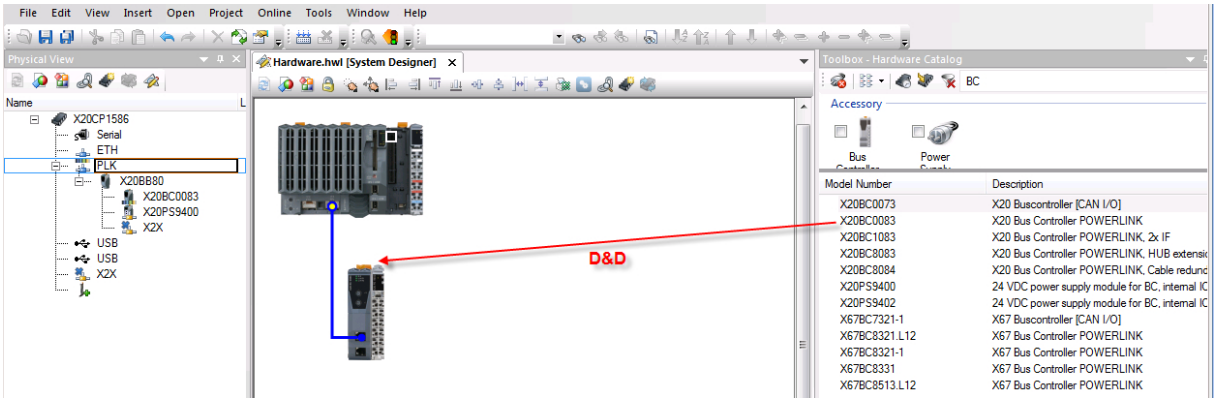


Figure 34: Creating a POWERLINK extension

5.2.8 Build of a Project

If all desired modules have been inserted and their required parameters configured, hardware engineering can be seen as completed at this point. A **„Build Configuration“** or **"Rebuild Configuration"** of the project must now be carried out. Once this has been done, Automation Studio can be closed.

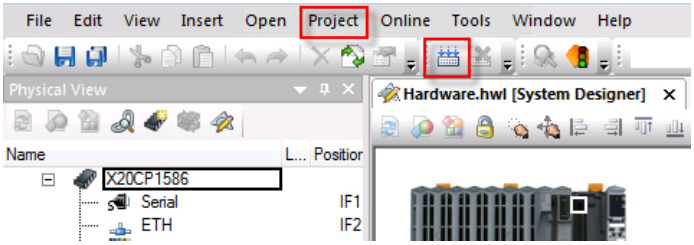


Figure 35: Build configuration

5.2.9 Configuring Module I/O Mappings

Once you are back in the CaeManager, you can begin configuring the I/O mappings for the modules. The dialog for **"I/O mapping of I/O module"** appears with a double-click on the desired module. All hardware I/O channels offered to project users for selection are configured in these dialog boxes.

Note: I/O

Incorrect entries are prevented later on in the project development because the I/Os are entered just one time and are then selected from lists.

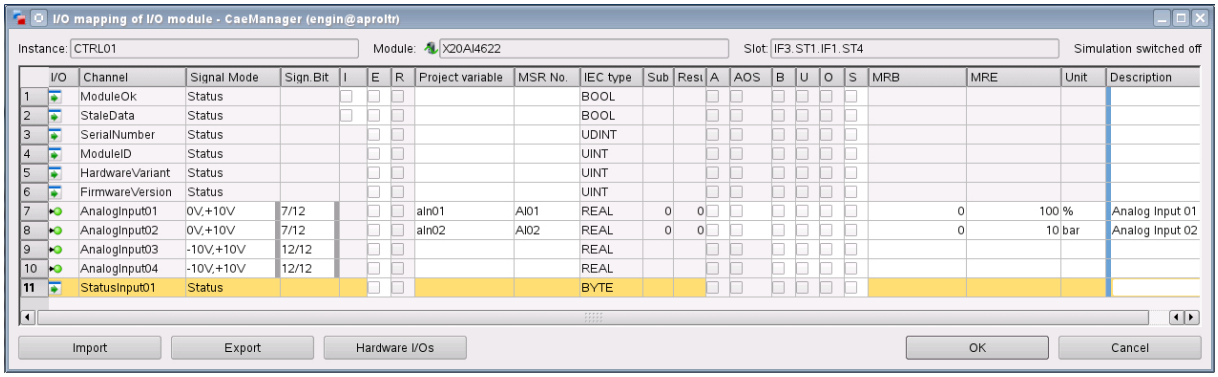


Figure 36: I/O configuration

All information which affects the I/Os can be configured (if allowed). This includes type, signal type, signal bit, event variable, name, tag number, data type, measuring range start, measuring range end, unit, and description.

Most of this information is provided for function chart engineering and should be used. The advantage of this method is that the data can be managed at exactly one location.

Note: Scaling channel

The measurement range limits are defined centrally and the scaled value is made available to the entire project. If, during commissioning, it is discovered that an encoder does not have the previously determined measurement range, the correct measurement range can be quickly applied without having to search for a scaling function block in function charts. If the signal is not linear, it is normal to scale from 0 - 100 (%) and then act upon the corresponding non-linear calculation in the logic.

I/O assignment can be handled for each module, as well as the entire controller, using an import file from the planning systems.

5.2.10 Download Parameters

The task is downloaded using another application, called the DownloadManager. In order to perform the download, it requires the data that is taken automatically from the controller's configuration (Ethernet). The following screens can be used to make any necessary changes.

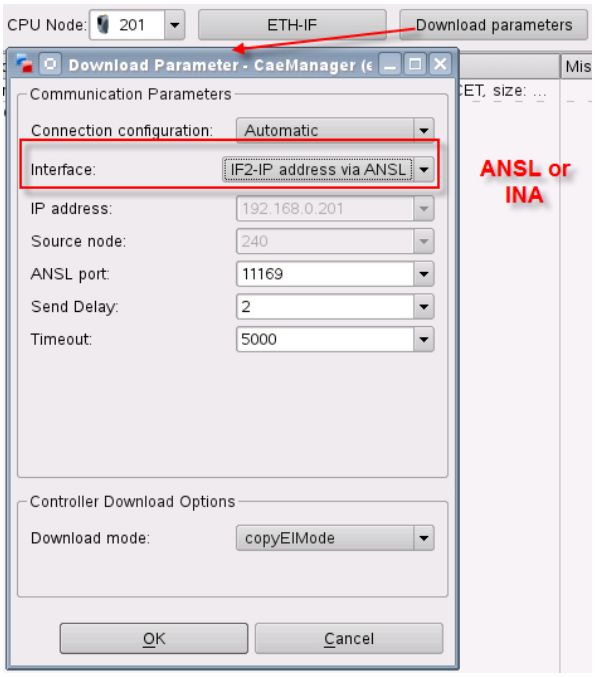


Figure 37: Download settings

The **"Download mode"** is one of the most important settings. A bumpless download is only possible using **"copyElMode"**. In this regard, bumpless means that I/O does not "drop out" thereby making it possible to expand the system "online" anytime.

5.2.11 Creating a Task

A new task can be created by right-clicking on the CPU or a task class in the **"Software configuration"** tab. Its name should fit the project's structure, and the description should be clear as to the task's function. Tasks can be created in all available task classes, and the total number of tasks is only limited by the amount of memory available on the CPU and its configuration.

A newly created task can be assigned to other function charts during the engineering process. These function charts are then executed in this particular task.

Note: task

The number of tasks should be kept to the necessary minimum since each task takes up space on the controller with its file information storage space and computing time (task scheduler). It would not make sense to start a separate task for each new function. Functions that have a clear relationship to one another should be compiled into one task and processed together.

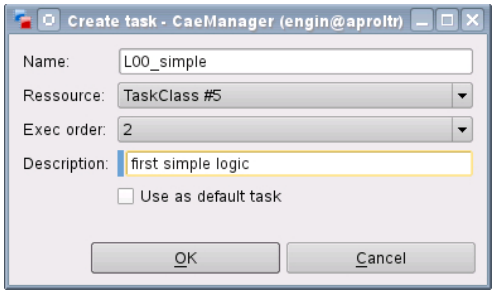


Figure 38: Creating a task

5.2.12 Inserting Modules

Modules with the extension **"*.br"** can be inserted using **"Software configuration"**. They can be broken down into system modules and project modules.

System modules are libraries and functions which are provided directly by **APROL**. (These are often Automation Studio libraries.)

Project modules are ".br" files developed explicitly for a project. In most cases, Automation Studio is the developing environment used with these modules.

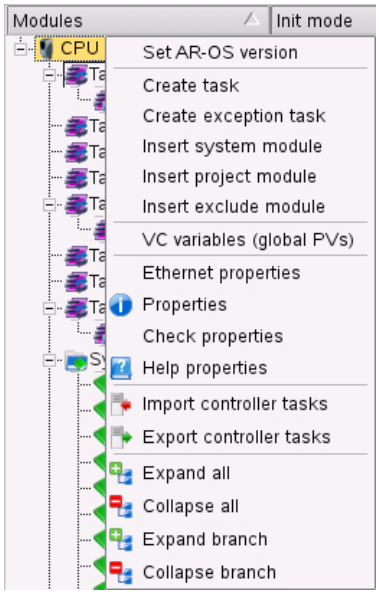


Figure 39: Inserting a ".br" module

5.2.13 Exclude Module

This function is needed to exclude modules from being deleted during a download. The B&R controller dynamically creates a few modules (online on the CPU) that are not in the download list. This means that these modules are deleted during a download by default. In most cases, this is not desired. Therefore, this function is used to specify the modules that should not be deleted during download.

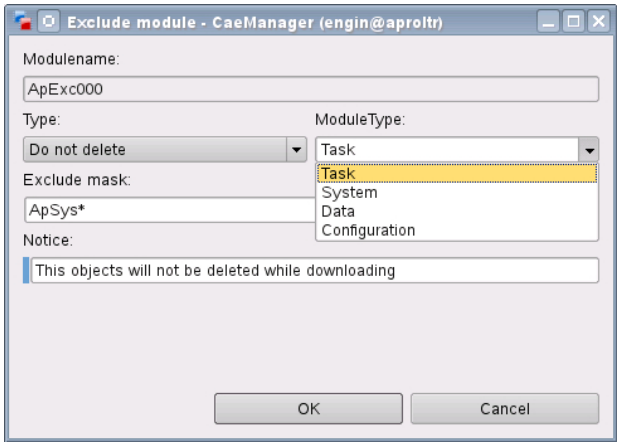


Figure 40: Exclude module

The **"Do not change"** function is also provided. The specified module will never be modified if this setting is used.

The **"ModuleType"** option determines the type of module.

In the **"Exclude mask"**, shortcuts (e.g.: Ap*) can be used or entire file names can be entered directly.

5.2.14 Ethernet Properties

A CPU's Ethernet properties are actually configured in Automation Studio. This small interface was created so that they can be read and checked quickly. However, changes cannot be made here.

5.2.15 Properties (Sysconf)

The configuration of the system settings **(Sysconf)** takes place in Automation Studio. The CPU system settings **(Sysconf)** can be shown again using the menu option **"Properties"**. However, changes cannot be made here.

5.2.16 Checking Properties

The **"Check properties"** menu item checks the current Sysconf settings with the **APROL** recommendations and reports deviations.

5.2.17 Help Properties

The **„Help properties“** menu item delivers a help page with the Sysconf recommendations. This page can also be opened by pressing F1 if "CPU" is selected.

5.2.18 Data Modules

It is often the case that a data module is needed to implement some sort of functionality or simply to continue working.

A data module is created in the same way as in Automation Studio, in the **„Data modules"** section.

5.2.19 Communication

Standard **APROL** definitions can be found in the **„Communication“** section. The most important configuration is the "controller-to-controller" connection.

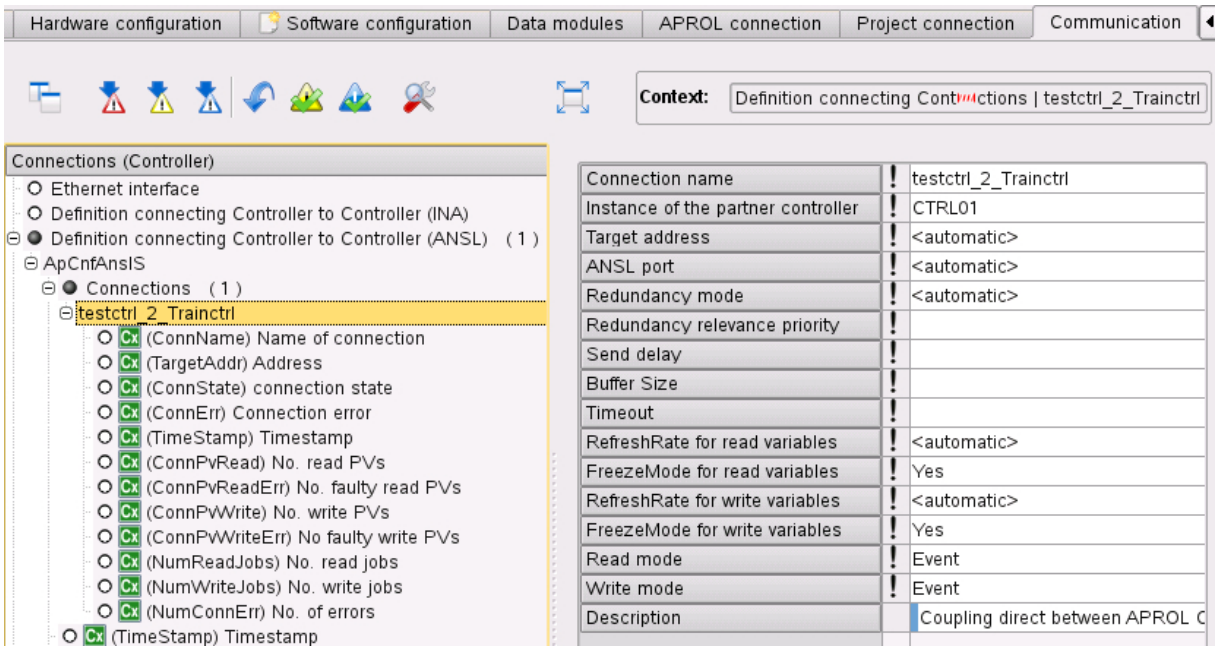


Figure 41: Controller-controller coupling

This is a simple way to exchange the data directly between the controllers (the variable must not go over the control computer (AnslDriver)). Essentially, the user specifies the communication path (routing path). The **APROL** system recognizes automatically if a variables needs to be "shoveled", because it is a variable with a physical view transition.

Therefore, if a connector is written in a function chart of controller A and read in a function chart of controller B, then it is automatically a variable for the controller-to-controller connection. Furthermore, the user can also define if communication is possible in both directions (read, write) and whether it should occur cyclically or be event-triggered when write variables are involved. Often, only "EVENT" is configured for "Write" (controller A), and vice versa on the partner controller (controller B).

Note: Controller-controller coupling

Direct communication between controllers is highly recommended because this increases reliability. A control computer is never as reliable as a controller.

Further details about this topic and the other communications can be found in the **APROL** documentation and in the expert training (module TM850).

5.2.20 APROL Connection

Standard connections for 3rd-party devices can be found in the **APROL** connections section. Amongst others, the MODBUS RTU/ASCII connection is realized here. Each connection requires a different configuration. The individual connections can be learnt in workshops or realized with the help of the **APROL** online help.

5.2.21 Project Connection

A custom connection can be made in this area. To do this, a corresponding rule must first be defined using the project rule editor. Details about this can be found in the **APROL** help.

5.2.22 Saving Changes

After completing the configuration, changes are saved using **"File / Save"**. This and any other change generates a new "modified" version when saved. This is only a temporary backup of the project part and is not subject to the version management. This "modified" version can now be versioned. The system displays the complete version information in the dialog box. In addition, the user has the option of adding a **comment** to this information. The **"New Release"** button allows you to increment the release number.

Note: Version management

Version management cannot be manipulated. A version number for an object can only exist once.

5.2.23 Activating and Compiling

Several versions of an object can be saved in the **APROL** engineering database.

A particular version is **activated** to let the system know which of the saved versions should be used for a build and for later usage in a project. This particular version must be activated in the version view of an object in the CaeManager engineering area. You can reach the version view by selecting **Versions** from an object's shortcut menu (right-mouse click). If the most recently saved version should be activated, you can select **Compile** from the object's shortcut menu.

Compiling checks the object to see if it can be **generated**. Any configurations made for an object are checked for their plausibility and correct syntax.

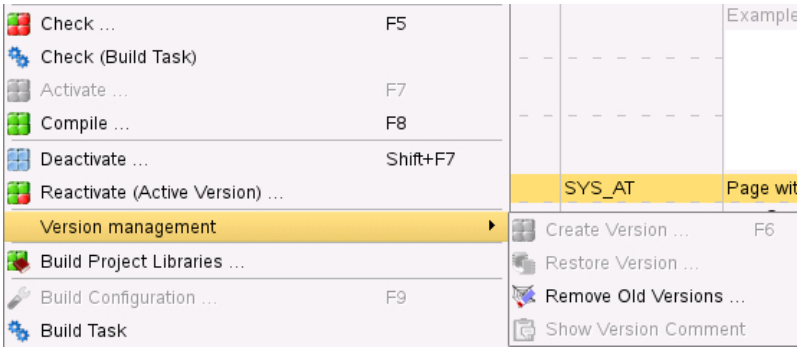


Figure 42: Menu for the selected object

Note: Productivity

In most cases, the menu **"Extras/Options..."** is used to configure the behavior so that after saving, the version is automatically activated and compiled.

Exercise 1:

Replace the B&R controller on the workspace with a POWERLINK rack. Mandatory entries are: IP address, host name, ... Create PVs for it in the I/O mapping area. These will be needed later on. In doing so, structure this part of the project logically with directories and tasks.

Space for additions:

6 Control Computer Configuration

In addition to the controller which we have just configured, the control system components also belong to the **APROL** process control system.

The control system components is the hardware configuration and the configuration of the higher level **APROL** system. As already described in the system introduction, the engineering, runtime, and operator systems can be combined in almost any way.

The **APROL** system can be extended at any time by adding other servers, operator stations, or gateway servers. It is thus possible to scale from the smallest application (a server with integrated operator station) to an extremely large project (many servers (MRS)).

6.1 Create control computer

A new hardware component can be inserted in the selected directory using the **"Create Part / Control Computer"** context menu.

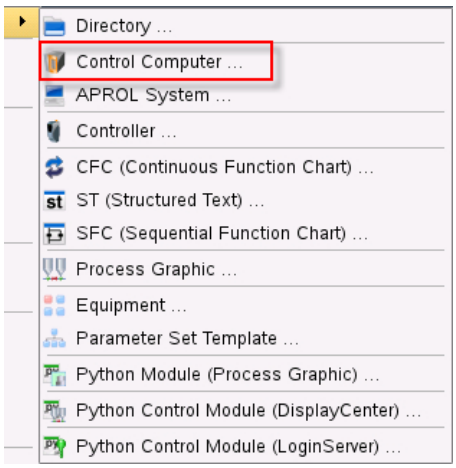


Figure 43: Create Control Computer

A unique name, description and instance must be entered into the following dialog box.

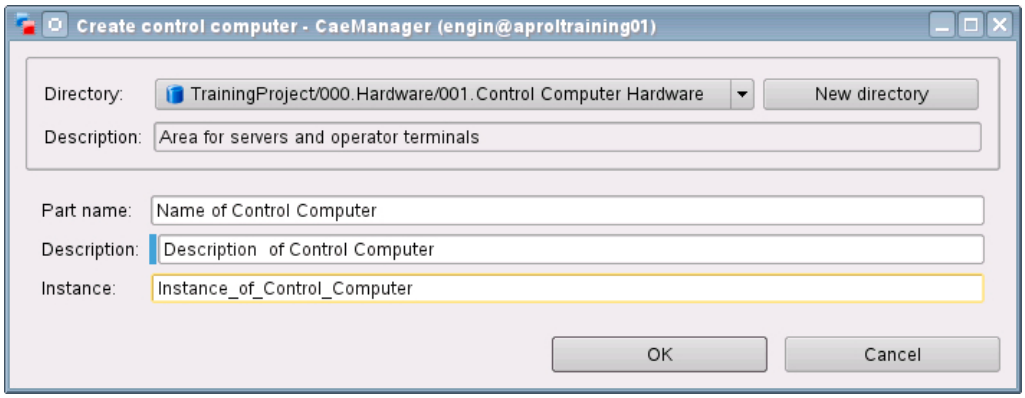


Figure 44: Creating a control computer

6.1.1 Configure Control Computer

A control computer can be selected in the tree once it has been inserted. Double-clicking on the control computer in the engineering area of the CaeManager opens the hardware engineering area.

The following dialog box allows you to make all of the settings needed for the control computer:

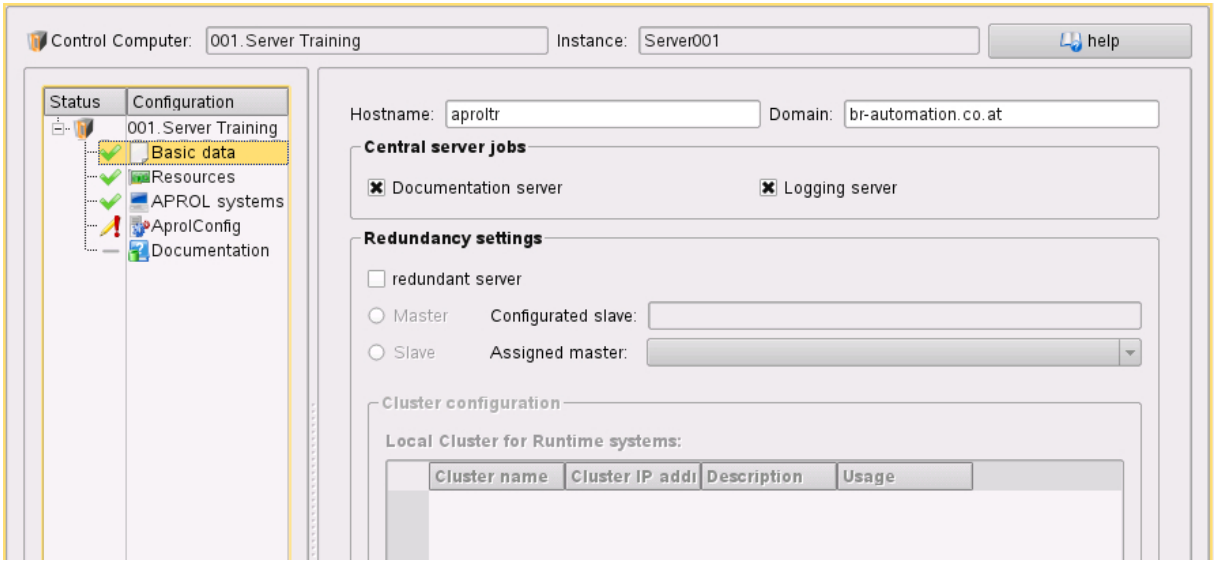


Figure 45: Control computer configuration

You can navigate in the leftmost area of the configuration section. The data created from the server installation by AprolConfig can also be imported here. Validators prohibit syntax errors in all input fields.

6.2 Creating the APROL System

A new **APROL** component can be inserted in the selected directory using the **"Create Part / APROL system"** context menu.

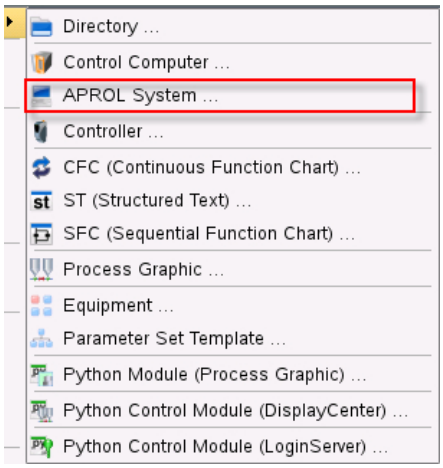


Figure 46: Creating the APROL system

A unique name, description and instance must be entered into the following dialog box.

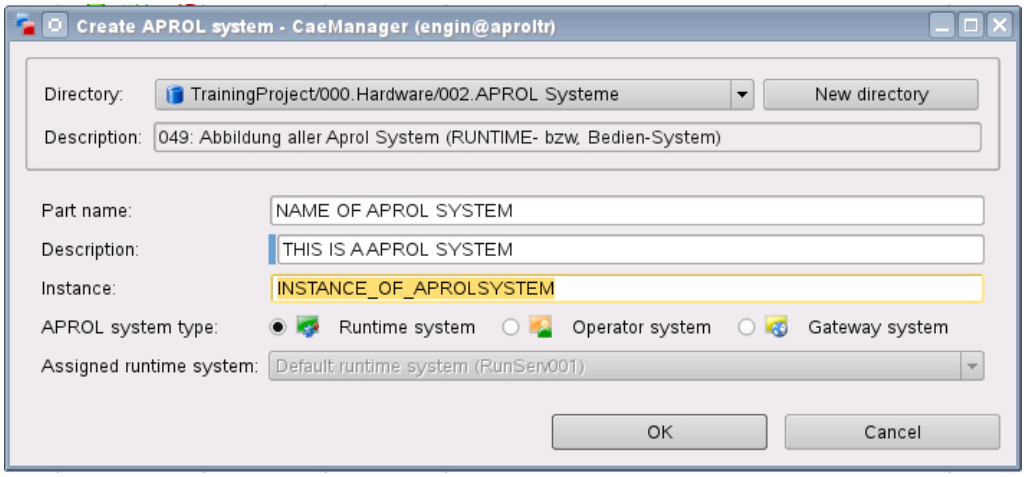


Figure 47: Creating an APROL system

6.2.1 Configuring the APROL System

An **APROL** system can be selected in the explorer once it has been inserted. Double-clicking on the **APROL** system in the engineering area of the CaeManager opens up the configuration.

The following dialog box allows you to make all of the settings needed for the control computer:

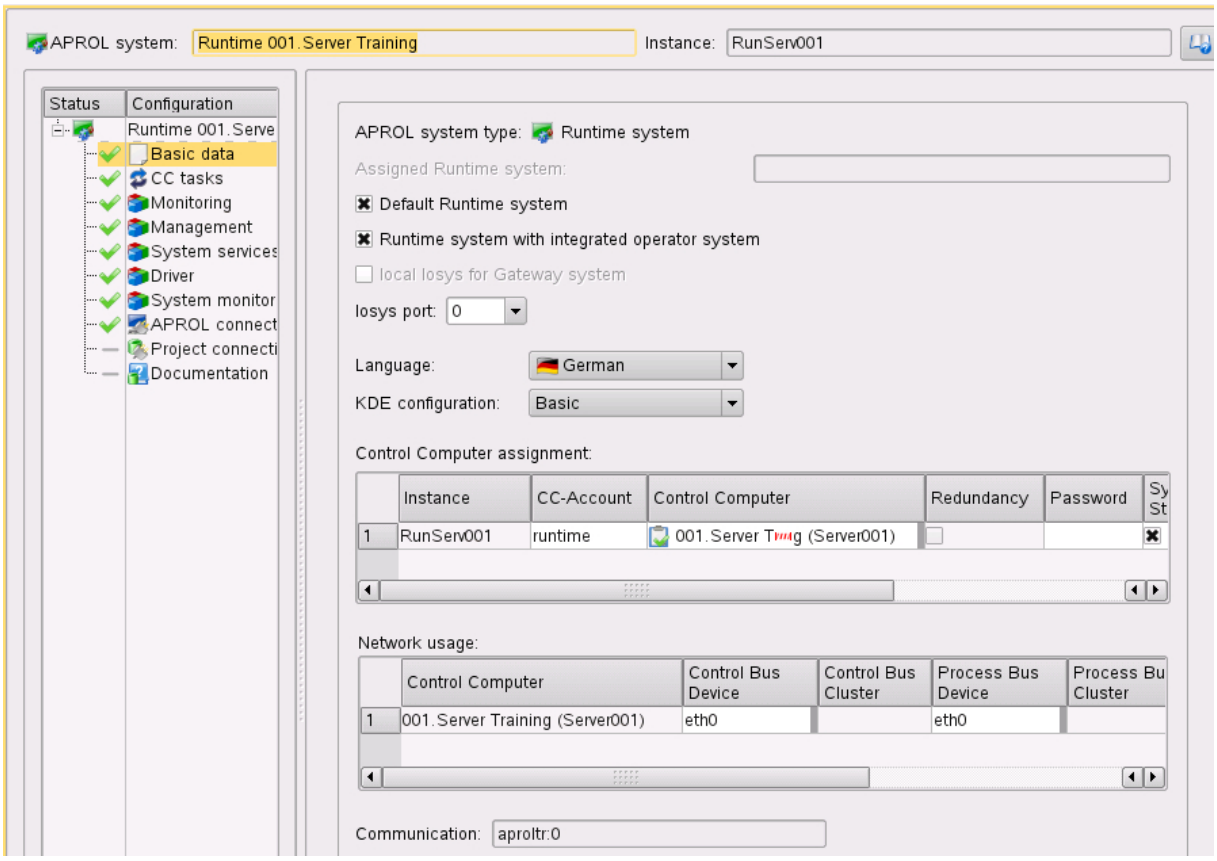


Figure 48: APROL system configuration

- **Master data**
  This is where you select what this system should be used for (runtime, operator or gateway system). In addition, you can set the Iosys port (default "0") as well as assign the respective control computer (hardware) where the system will be running. The Iosys is the process data base All relevant variables, their source, and target information is stored here.

- **CC tasks**
  Similarly to a controller, the duties that a control computer must perform are mostly processed in several tasks.

  One (or more) control computer tasks can only be implemented on control computers where a runtime system has already been set up.

- **Operating and monitoring**
  Applications for operating and monitoring the plant are configured here, e.g. DisplayCenter, TrendViewer, etc.

- **Management**
  The applications in this area handle the **APROL** system. The OperatorManager and StartManager are a part of this.

- **System services**

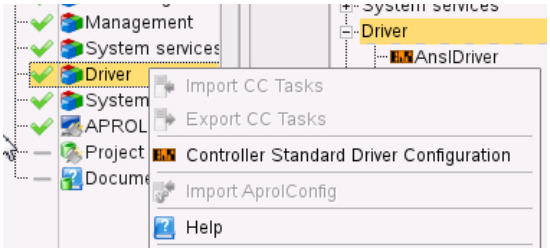
The system services included here are already configured by default but can be modified if needed. This involves background programs which are started in the Linux runlevel 3. (Iosys, AlarmServer, TrendServer, SysInfo, etc.)

- **Driver**
  Drivers establish the communication between the control computer and the controller or 3rd-party device.

Note: Integrating a newly configured controller

If a new controller is created in **APROL**, the respective AnslDriver must of course be configured.

This can be done by selecting **Controller standard driver configuration** from the Driver shortcut menu.



The dialog box that appears then lets you use the recommended standard configuration for the driver.

- **System monitoring**
  The services and processes included here are used for system self-monitoring tasks and for the analysis of a wide range of components. There is therefore always a debugging of all processes including signal tracing.

- **APROL connections**
  Connections to 3rd-party devices and other systems can be implemented right on the RUNTIME server or via a gateway server in addition to being set up via the controller. The settings of these connections must be entered in the **"APROL connections"** section. The drivers that are eventually needed for this are to be configured in the **"Driver"** area. Details can be found in the APROL help.

- **Project connections**
  Project-specific connections – which are not listed with the default connections under "APROL connections" – can be created by the user here.

- **Documentation**
  The data stored here is written automatically to the as-built documentation.

6.3 Saving Changes, Activation, Compilation

Saving, activating, and compiling control computers and **APROL** systems is done in exactly the same way as with the controller.

Exercise 2:

Configure the required AnslDrivers for the previously created controllers with the "Standard controller driver configuration" in the existing runtime system.

Optionally, you can create a new control computer and **APROL** system. Once again, pay attention to the "readability" of the project.

Space for additions:

7 Configuring Simple Controller Logic

7.1 General Information about Function Charts

APROL is programmed with a link-oriented, graphic function block language according to IEC 1131-3. The following block types are available when creating function charts.

• **Functions:**	Logic created in ANSI C which is processed within a cycle and whose return value is named the same as the function itself. Because of this, a function can only have one return value (output).
• **Function blocks**	Logic created in ANSI C which can have more than one return value (output).
• **Display driver function blocks**	Interface between logic and a process graphic.
• **Control system function blocks**	Blocks provided by B&R which depict control system functions such as the alarms, trends, PDA, etc.
• **Macros (hyper macros)**	Function charts which can be placed as a function block. The pins jutting out of the I/O bars on the block can be connected with the surrounding logic.
• **UCB blocks**	User Configuration Block: Mostly programmed using Python or Shell script.
• **ST/SFC blocks**	Blocks which are created with the ST or SFC programming language.

The types of subroutines placed in a function chart can be separated by their color.

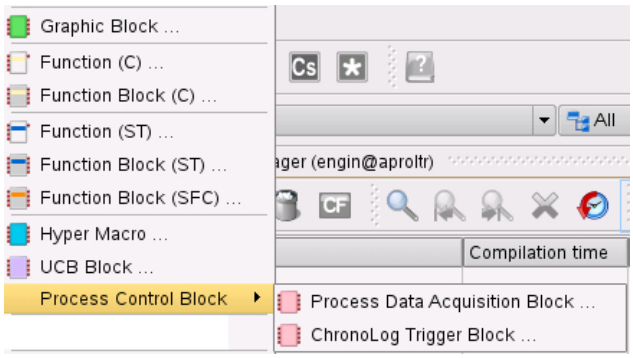


Figure 49: Block types

These blocks can then be placed in function charts with **drag-and-drop** and then connected.

7.2 Creating a Function Chart

A function chart can be created in the selected directory with the **"Create part / Function chart"** context menu.

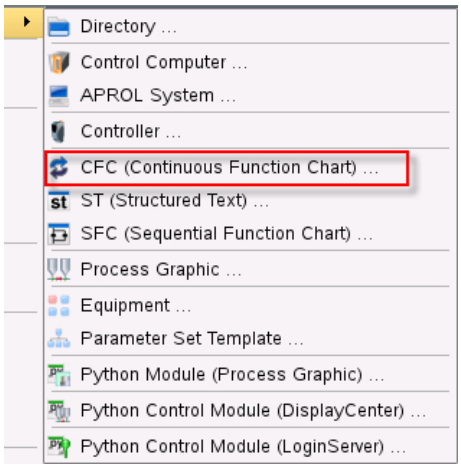


Figure 50: Creating a function chart

In the following dialog box, you must enter a unique name (Part Name), descriptive information (Description) and an instance name (Instance). The instance name is a brief label which the system uses to form the name of the process variables configured on the function chart.

Caution: Nomenclature

The instance name must start with a letter and be unique in the system (e.g. DF1, A2). A name can have a maximum of 48 characters. Since names are important when generating a task, they must conform to C programming language conventions. The use of umlauts, special characters, numbers without letters, etc. is not permitted! The validators included in APROL do not allow this either.

In addition, the corresponding task must be selected on the controller or control computer where the new function chart should later be executed.

The **"CFC Type"** property determines when and how the function chart is executed.

- **Init -** The function chart is only executed one time when the task is installed.
- **Cyclic -** The function chart is executed cyclically.
- **Exit -** The function chart is only executed one time when the controller is stopped or the task is deleted.

The **"CFC creation"** property determines which physical view or environment the function chart can be assigned to.

- **Simplified function chart creation for controller tasks with optional 'control computer' task part:**
  If this option is selected, a function chart is created where one part of the logic is executed in one controller task or in one control computer task. Whether the block should run in the controller task or the control computer task can be set later in the block's properties.

- **Simplified function chart creation only for 'control computer' task:**
  With this option, a function chart is created whose logic is only executed on the control computer.

The execution order of all blocks in the function chart can be customized with the **"Manual setting of block execution order"**. It is recommended to let **APROL** decide the order. A manually specification is rarely necessary (loops in CFC, etc.).

You can use the corresponding buttons in the **"Dimensions"** area to manipulate the size of the function chart. This is possible in all directions, even after engineering has already begun. Reducing the chart size can only be done with pages which don't include function blocks.

Note: Expanded function chart

A function chart can be a maximum of 25 x 25 pages. It can be extended in any direction including upwards and to the left, which is often very important during editing. Existing connection lines are extended with the auto-router.

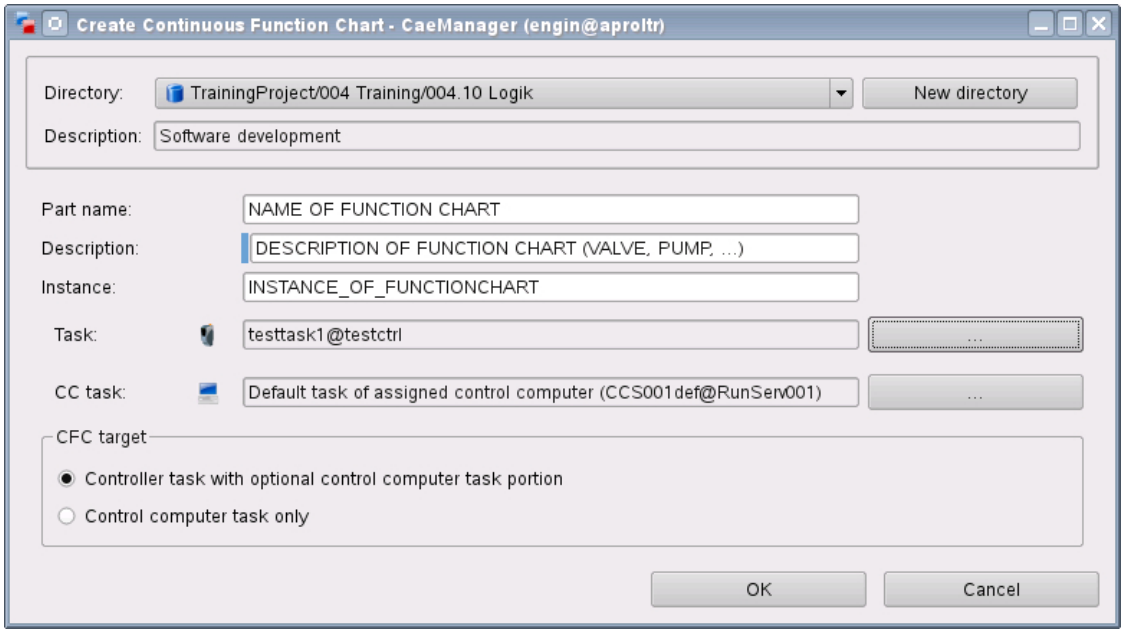


Figure 51: Creating a new function chart

7.3 Properties of a Function Chart

The global properties of a function chart are opened in the explorer view with the right mouse button (not in the open CFC). This dialog box allows you to make adaptations such as task allocation and renaming.

Note: Changing tasks

Changing the associated tasks in a function chart is not subject to the version management system. As a result, this change is then applied to all versions of the object.

7.4 Blocks

7.4.1 Inserting Blocks

In order for logic blocks to be placed, you have to be able to access the block libraries linked to the project. Libraries can be reached under the **"Context: Continuous Function Chart"** tab in the CaeManager navigation area. The desired blocks are selected from the library tree and dragged to the function chart with **"drag-and-drop"**.

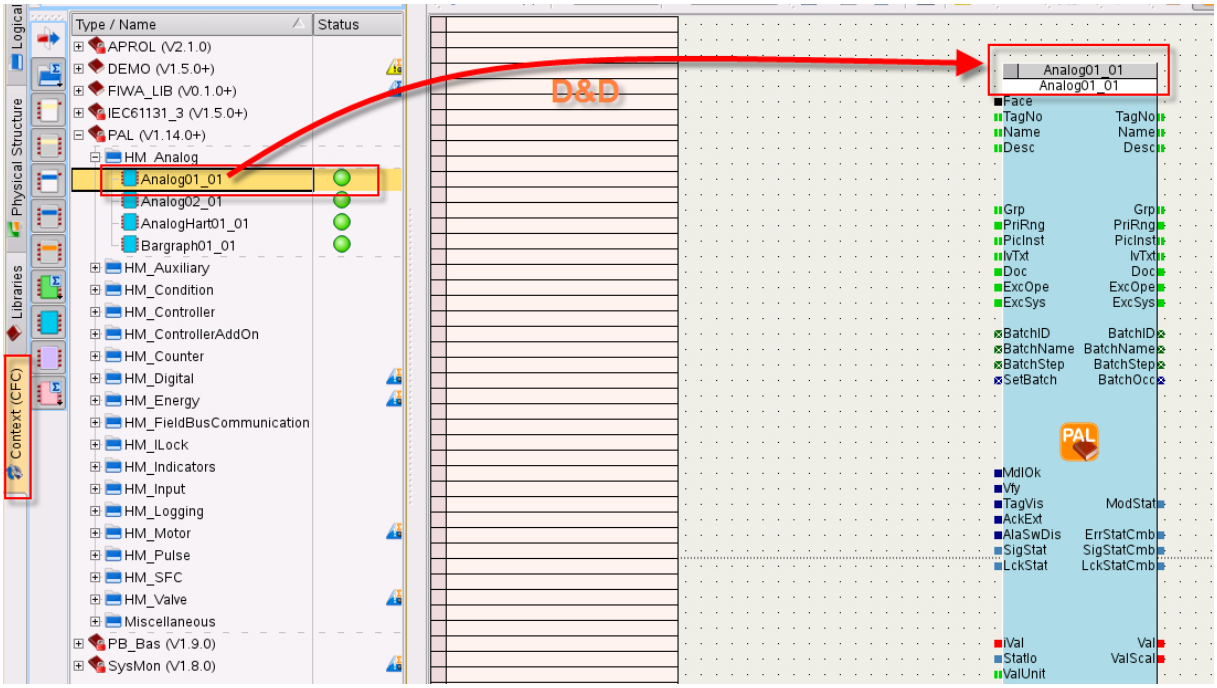


Figure 52: Inserting function blocks

Note: FC readability

Place the block so that the incoming and outgoing lines are as straight and clearly structured as possible. This makes it much easier to read the function chart.

Blocks must be given unique names (instances) because the same blocks can be used in a function chart several times. If you have placed a function block into a function chart it is automatically given a name. When this is done, the name of the function block type is given an underscore and a number. If this function block type appears again in this chart, the number is incremented by one. It is generally clearer if the instance of the block is modified manually, which allows it to be given a system-specific name that makes sense (TAG, MSR).

Note: Naming

The function blocks important for the logic should be given a unique function-related, name (tag names) to simplify debugging.

When a block is inserted, a copy of it is not really inserted into the function chart; only a reference to the block is created in the database and assigned individual parameters (instance name, constants, etc.). In the language of object-oriented programming, this would mean: **an instance of the selected (block) class is created and configured.**

The difference between a copy and an instance is important at this point since it has effects on the handling of modified function blocks. **If the class is modified, it changes the instance which comes from it.**

Changing a block in the library immediately affects each occurrence of that block in the project. If this should not be done, then the function block may not be changed, but has to be copied instead. Changes can then be made to the copy (this implies a new block with a new name). The changes made only come into effect when this new function block is used in the project.

After the function block has been positioned in the function chart, its **shortcut menu** can be opened (position the cursor on the function block and right-click). Various settings can then be made depending on the block type.

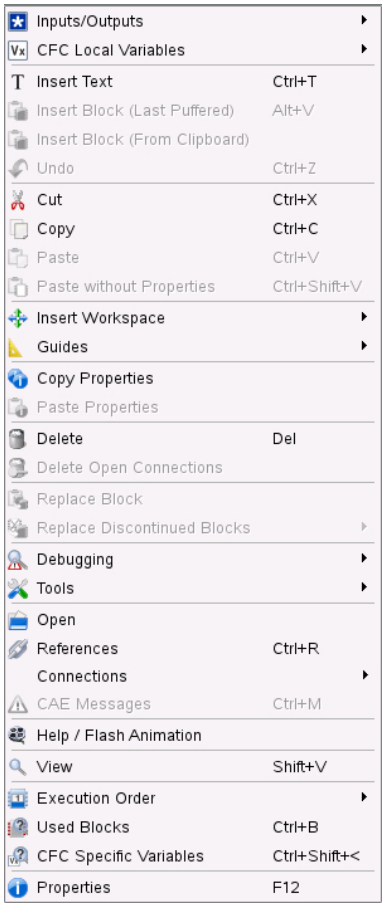


Figure 53: Block menu in the FC

7.4.2 Moving Blocks

A function block can be moved at a later time. To do so, click and drag the function block to the desired location. Several blocks and lines can be selected and moved using a selection box.

7.4.3 Function Block View

When working with hyper macros and picture blocks, it is possible to switch from the block view to the function chart or graphic view by double-clicking or selecting "View" from the block's shortcut menu.

The arrow symbol in the function chart's toolbar is only visible if you have switched to the function chart view for a block. Clicking on this button leaves the function chart view of the function block.

Figure 54: Arrow for exiting a hyper macro

Note: Hyper macro view

As we will see later, it is also possible with online debugging to change to the internal function chart view of a hyper macros.

For all other function blocks types, the **"View"** entry in the shortcut menu (or double-click) cannot be selected.

7.4.4 Properties of a Function Block

To modify the properties of a block, select **"Properties"** in the shortcut menu (or the middle mouse button). This window provides instance-specific information about the block.

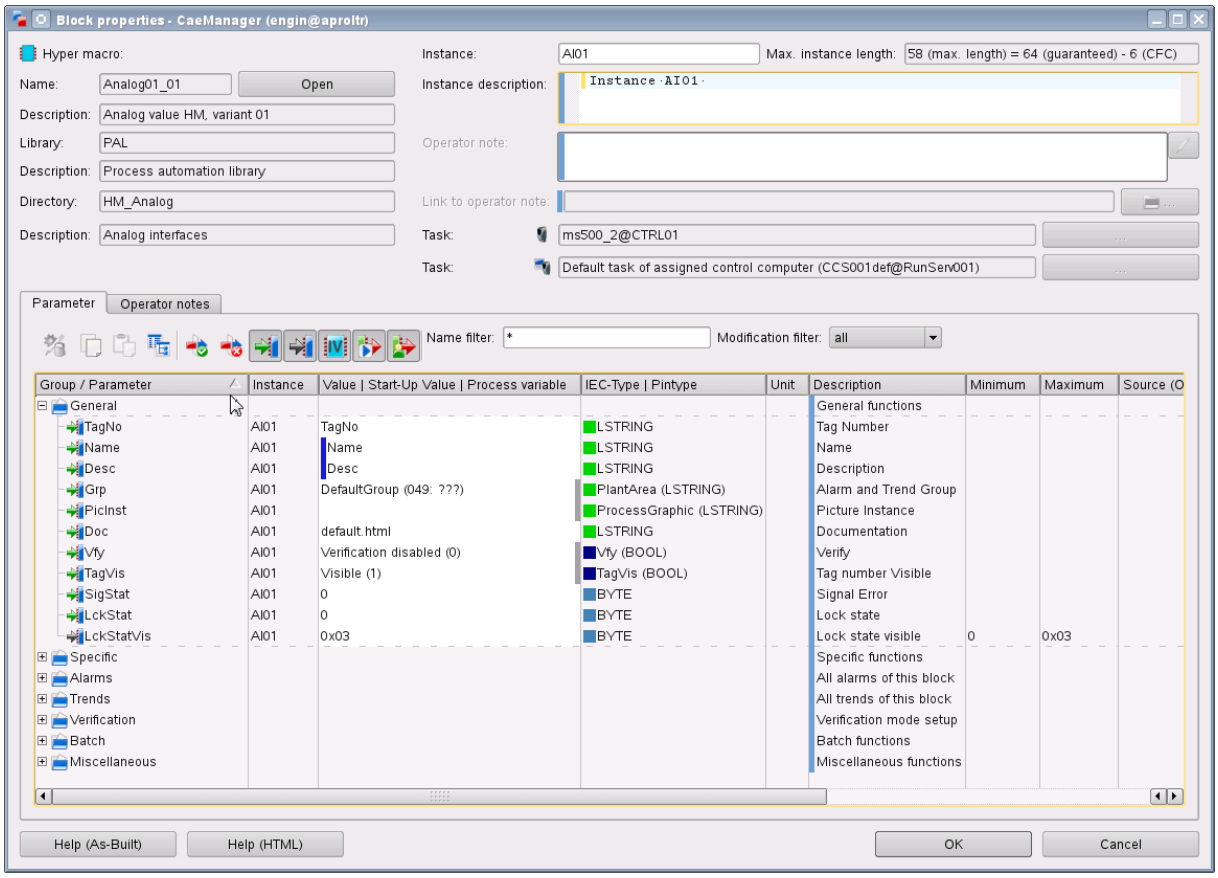


Figure 55: Function block properties

A different name than the function block name (instance) given automatically can be entered in the **"Instance"** field.

A task which is other than that of the function chart is configured with **"Task"**. This is only possible when the corresponding function has been selected in the properties of the function chart in **"CFC target"**. (controller and control computer task allocation must be made possible)

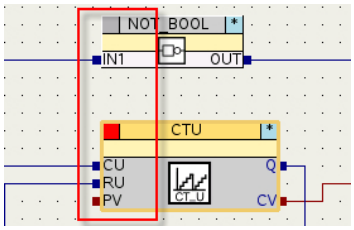


Figure 56: Modified task for the block

Note: Switching tasks

After exiting the dialog, a red square appears at the top left of the function block to indicate that the block's place of execution was manipulated.

Note: Operational security

For reasons of operational safety, it is not advised to run safety-oriented system functions on the control computer. This computer does not have – even on redundant systems – the operational safety that a controller can offer. The operational security of a controller can be raised by configuring it as being redundant.

"Input" lists the visible inputs on the function block. These can be wired or configured accordingly.

It is possible to connect different data types if the data types can be type converted in such a way that nothing is lost during the conversion process. In this case an implicit conversion takes place, which is identified by a small triangle located on the block input. A tooltip over the triangle indicates the data types that were used in the conversion. If this type of implicit conversion is not possible, then conversion blocks must be used. In this case, loss of data and precision is accepted.

"HM Constants" are not fed out of the block. They are assigned a fixed value which can only be changed during development.

Entries made for HM constants and inputs are checked by validators to prevent incorrect input values.

Note: Validators

It is not possible to enter syntactically incorrect input anywhere in APROL. Every input field is equipped with a validator for checking this. A red warning triangle is display when an incorrect entry has been made, and the entry cannot be saved.

"Help" opens up the help for the block. This help is opened in an HTML file using the browser configured under **"Extras / Options"**.

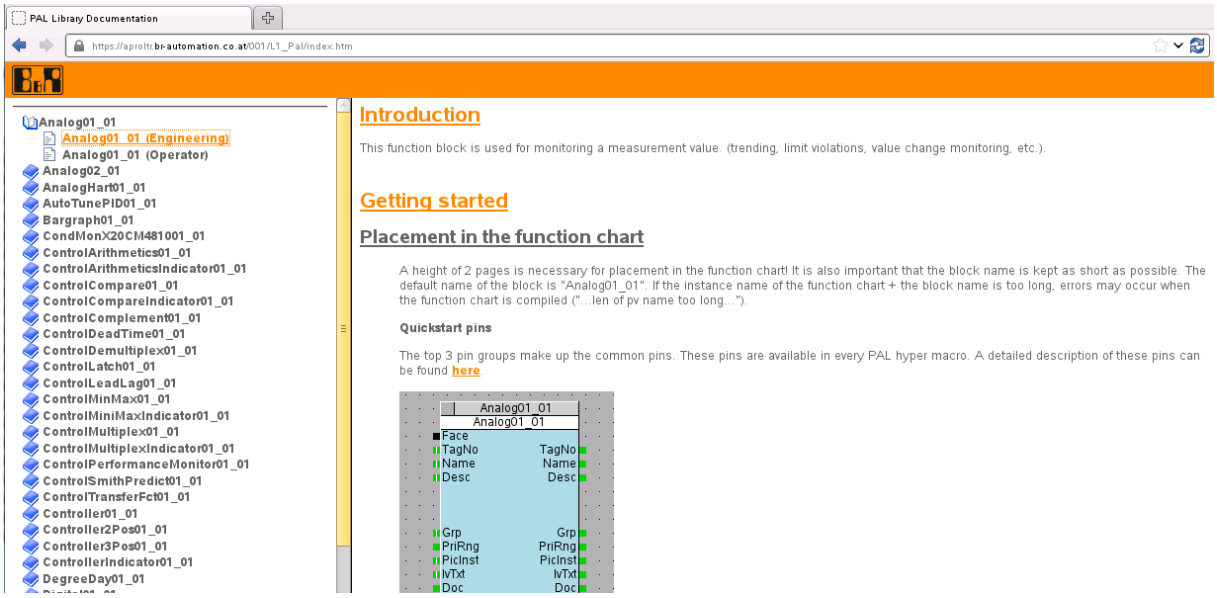


Figure 57: Block help

The following information is displayed in the list of function block inputs:

Name:	Name of input
IEC type:	Data type of the input.
Description:	Description of the input
Unit:	Unit / dimension of the input
I/O constant:	Instead of a supply line at the input, a constant (or the default value) can be used.

Detailed descriptions and information are available for the libraries provided by B&R. The developer designing the custom libraries is responsible for this description. In most cases, this is handled using the as-built documentation in library engineering. Each entry made and comment entered is automatically placed in the documentation.

7.4.5 Entering Constants

Entering constants has a high degree of importance for certain function blocks. Why and what must be input as a constant can be found in the function block description (Help item in the shortcut menu).

The **"Time"** data type has a particular input format. A length of one second can be entered in the following ways:

1000	Interpreted as ms
T#1s	Interpreted as Seconds (min, h, d, etc.)

With input help	The desired time value can be entered comfortably by opening the input help.

7.4.6 Deleting Function Blocks

To delete a function block, right click on the respective function block and select "**Delete**" (Entf).

7.4.7 Comments Directly in the Function Chart

To improve the readability of the function chart, it is recommended to save comments directly in the function chart. This is done with the **"Insert text"** shortcut menu. An input window is opened to enter the text, which is then placed at the desired position in the function chart.

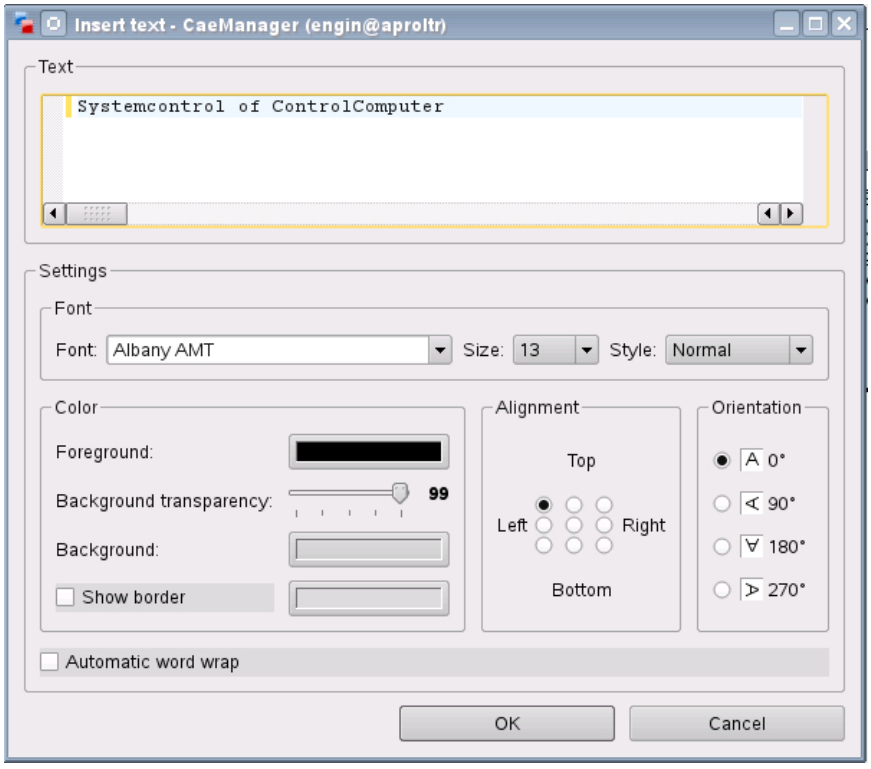


Figure 58: Entering text for a function chart

7.5 Connections

7.5.1 Drawing Connections

A simple left-click can be used to select the points to be connected. An auto-router automatically selects the most optimal path between these two points. To limit the variation possibilities of the auto-router, you can click and add supporting points in the function chart by left-clicking. This can be useful, for example, if lines have to be laid over large distances in a kind of cable tree.

7.5.2 Moving Connections

A supply line is moved the same way as a block. Click on the line, hold the mouse button and move it.

7.5.3 Supply Line Properties

The **"Connection properties"** dialog box can be opened by right-clicking on a supply line and choosing the **"Properties"** function (or with the middle mouse button).

The source is displayed in the top part of the dialog box and a list of targets for this supply line is displayed in the lower part of the dialog box. If you click on one of the targets or the source in the list, then the function chart automatically jumps to this position. The blocks and lines are highlighted by doing so.

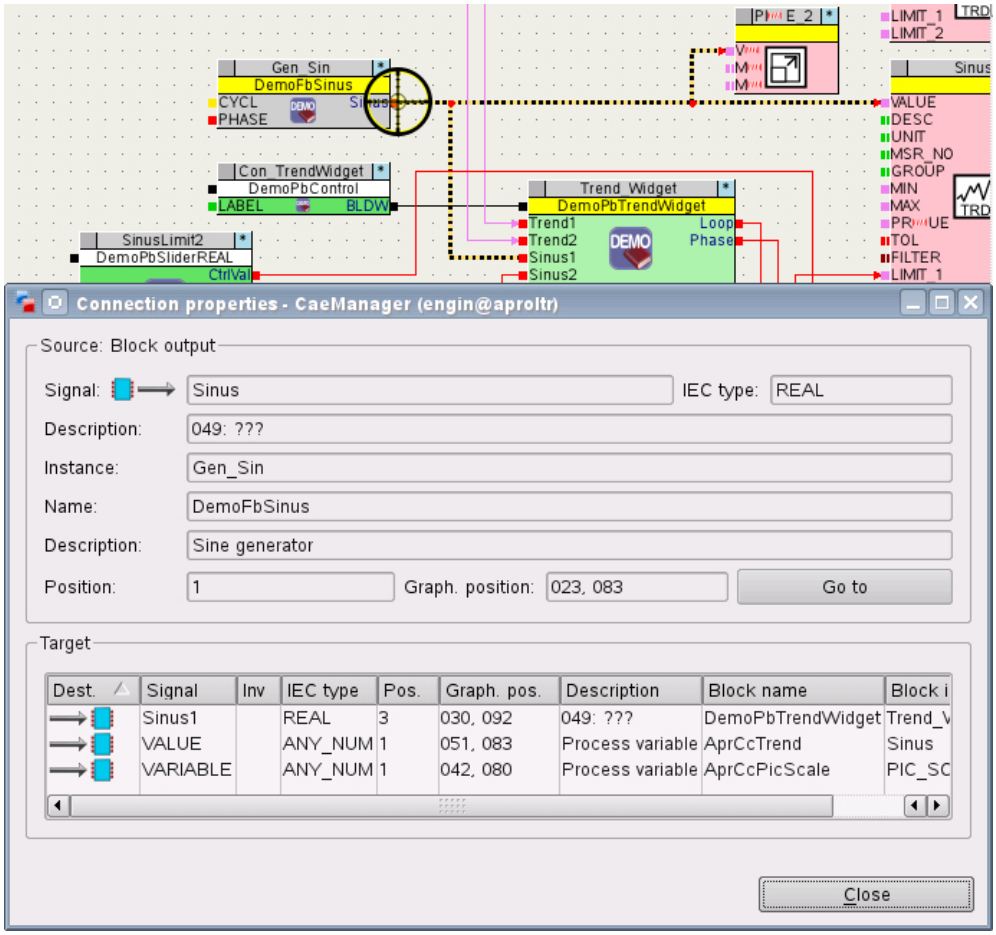


Figure 59: Connection properties

7.5.4 Deleting Connection Lines

To delete a supply line, select the line to be deleted, right-click, and select the **"Delete"** function.

7.6 I/O Borders

The I/O borders form the interface of the chart to the hardware I/Os of the controller, to other function charts (connectors), to the runtime system, and to the parameters which can be downloaded to the system with the ParameterCenter in the runtime system.

Hardware inputs, communication variables, connectors, system variables, parameters and system constants that should be read are entered in the input bar. Data that should be written, such as hardware I/Os, gateway I/Os, connectors and parameters (for parameter uploads), is entered in the output bar. Right-clicking on the input bar allows input options to be selected.

These variables can be placed anywhere in the CFC if desired. It is also possible to work with internal variables.

If you right-click on a variable in the function chart that you have already created, you can delete the input, or open the variable's cross-references with **"References / Properties"** .

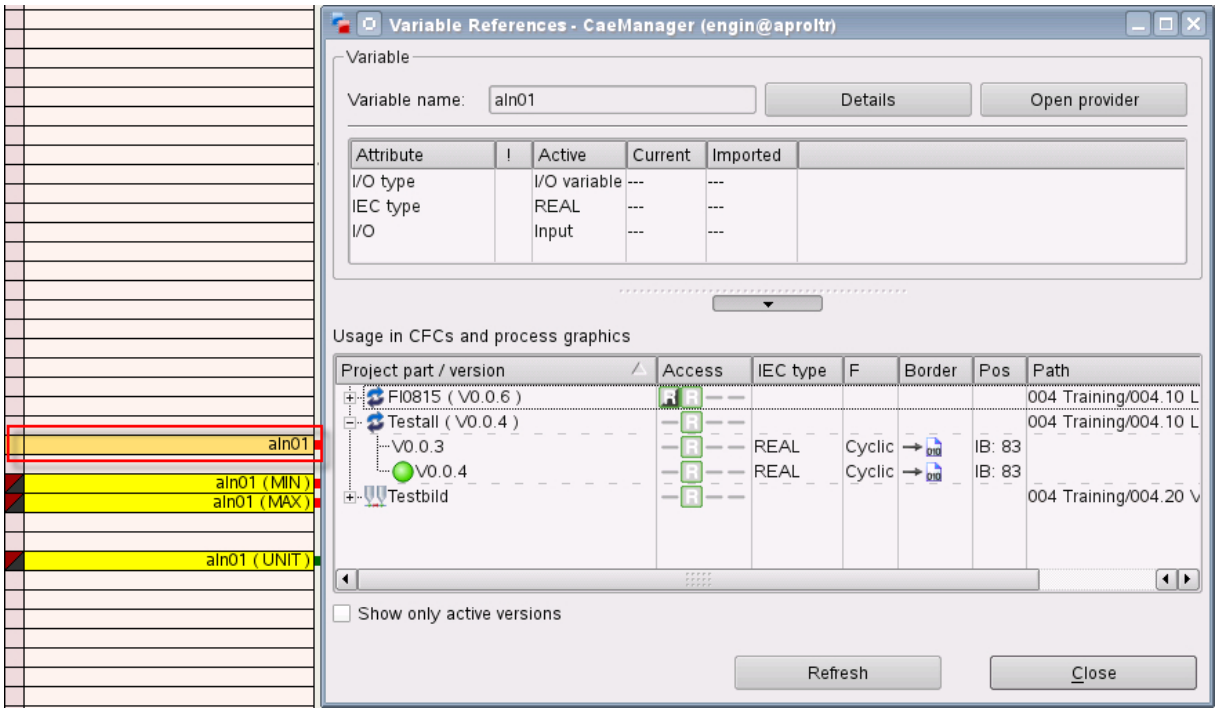


Figure 60: Properties and references of a chart's input

The input pin is colored yellow. Right-clicking on this cross-reference (or double-clicking) will immediately bring you to the corresponding function chart. The entry is also highlighted yellow there.

Delete deletes the specified input, where only its assignment to a connection is deleted, not the connection line itself.

An I/O is **shifted** by clicking and moving to the desired position with **drag-and-drop**. The connection line is shifted along with the I/O. The connection is deleted by holding the "Shift" key. All of this works the same way with multiple selections.

There are many more tips for function chart engineering. They can be seen in the "CaeManager Tips" or in the **APROL** online help. The "Videos" for the different features can be found there.

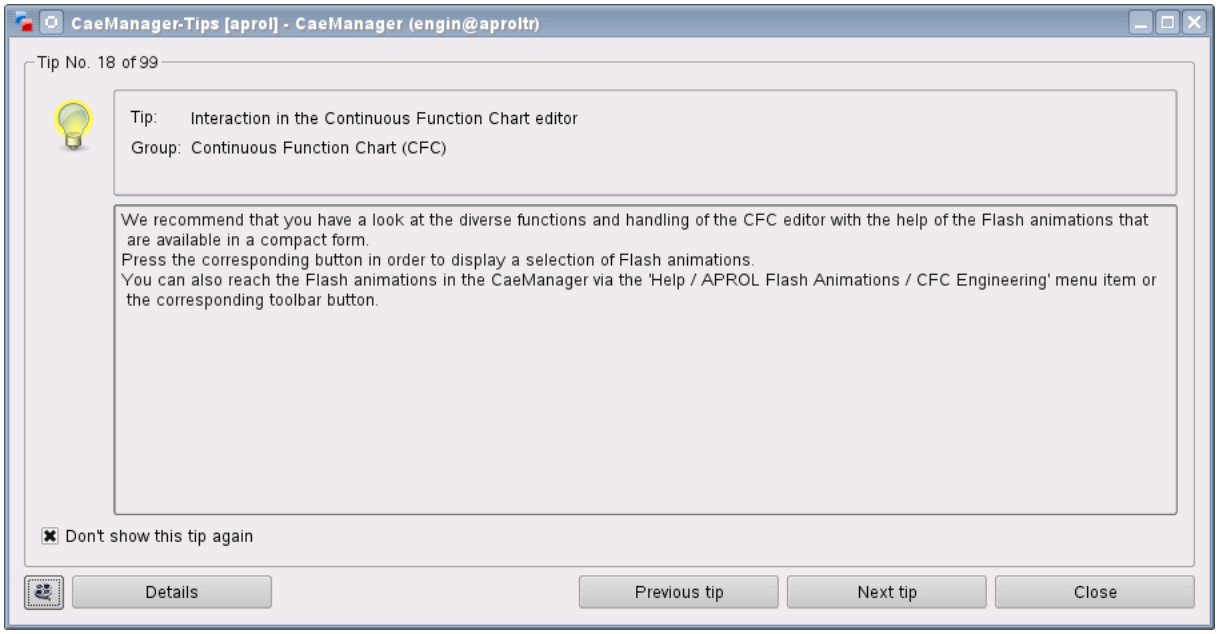


Figure 61: Tips for function chart engineering

7.6.1 Selecting a Hardware I/O

The following "Hardware I/Os" dialog box lists all hardware inputs that have been configured. Take this opportunity to limit the amount of data by defining a corresponding filter in the upper part of the window. **Wildcards "*"** can be used in the **"Name"**, **"Tag no."**, etc. input fields.

The "**Use extended I/O information**" area can be used not just to select the actual variables, but their properties, description and states as well.

The status of the input value is determined with analog inputs if the configured input signal is between +4 mA and +20 mA. In this case, open-circuit detection is handled with software. If an error occurs, a 1 is returned as the **WORD data type**. Otherwise, a 0 is returned.

The variable is placed on the border via double-click or **drag-and-drop**. The selection dialog remains open while doing so, so that other variables can be inserted.

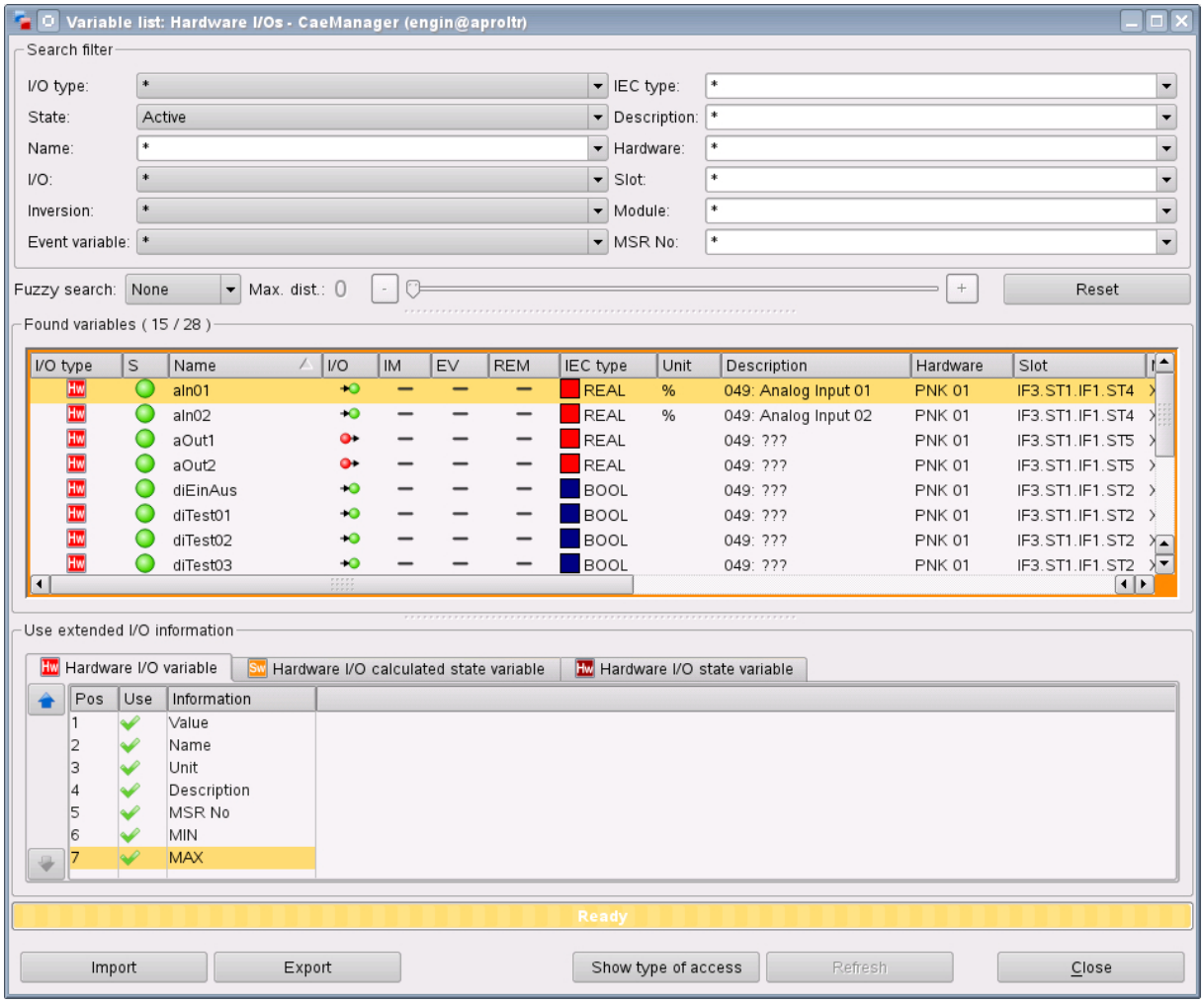


Figure 62: Inserting hardware I/Os

7.6.2 Selecting a Gateway I/O

System-wide connections are established using a gateway variable. All of the variables defined in the project on a control computer, controller, and gateway connection are available.

Selection takes place the same way as with the hardware I/Os.

7.6.3 Selecting a Connector

A connector can be used to establish plan-wide connections. The connector is created as a task-local variable and is then converted to a global variable if a connection is created beyond the task.

Selection takes place the same way as with the hardware I/Os.

If the desired connection cannot be found, it is possible to set up a new one.

Note: Local / global variables

A task-local variable only uses a memory location on the controller when at least one of its tasks is running. A global variable constantly requires a memory location because its current value may also be needed by other tasks.

The **"Create connector"** button opens the dialog for creating a new connector. The **"Name"**, **"IEC type"**, **"Description"**, and an optional **"Unit"** can be specified here. If both **"Activate"** and **"Place on border"** boxes are selected, the connector can be created and placed with **"Apply"**.

Note: Connectors

Make sure that the relationship between connector inputs and connector outputs is always a 1 to n relationship. Therefore, a connector can only ever be used in one case as an output, but in several cases as an input. Reciprocally, a connector must be used in at least one case as an input and in exactly one case as an output.

7.6.4 Selecting a System Variable

The program configured for the runtime system is executed on every computer configured in the runtime system (operator station 1 to n, runtime server, and gateway host) at runtime. **APROL** automatically generates system variables from the information configured in engineering, which is then provided to the user for monitoring purposes.

Some system variables of data type INT are provided and can be evaluated as shown in the following table.

Value	Information
0	No information about the status
1	Program running normally
2	Program not running normally
3	Program could not be started

System variables receive information about the server or operator station that is being monitored. Used in this example: Idle time, disk space, license status, etc.

Incidentally, these system variables are also used in the "SysMon" library (included in delivery), which makes it possibly to implement system monitoring very quickly.

Selection takes place the same way as with the hardware I/Os.

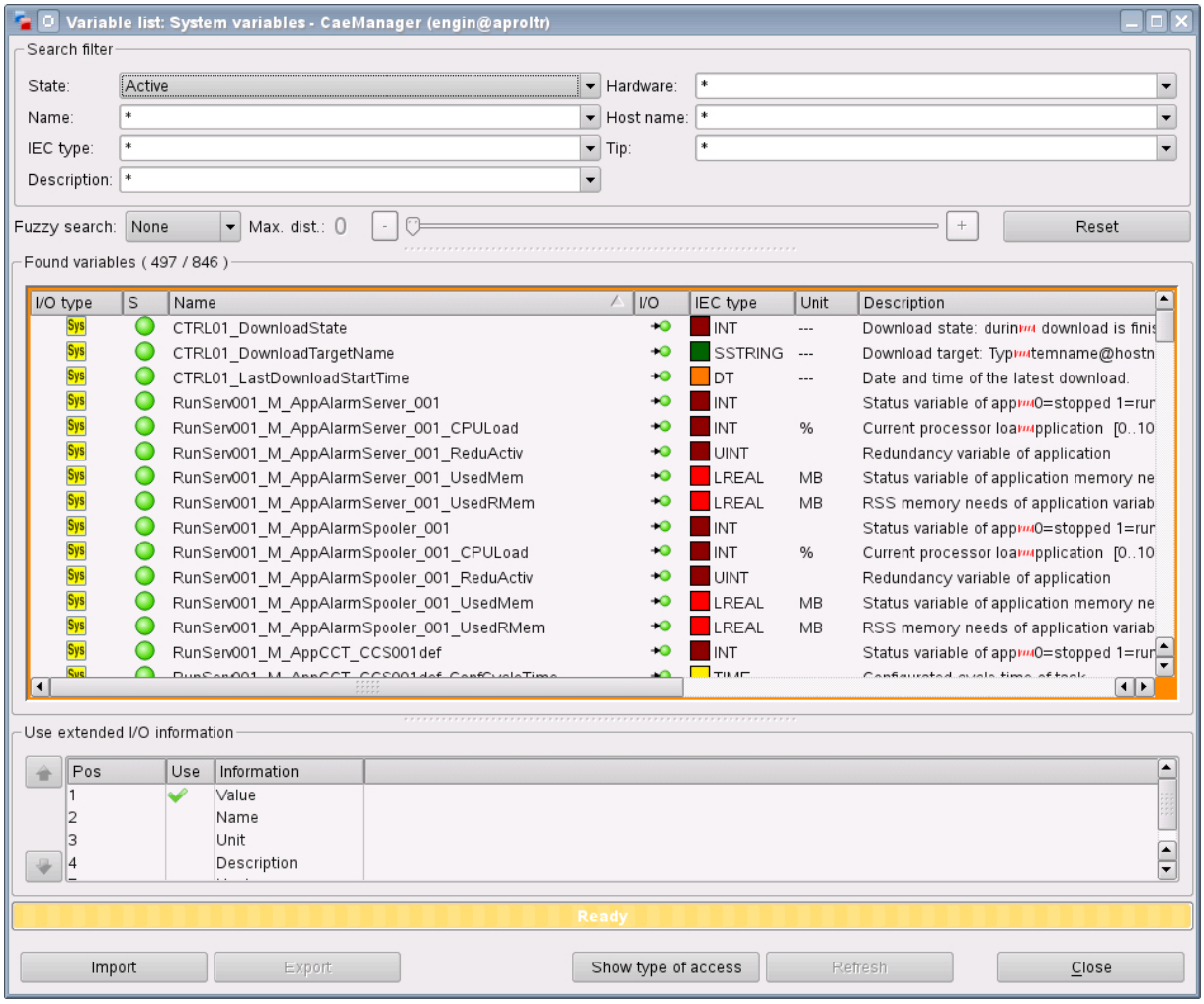


Figure 63: Selecting system variables

7.6.5 Selecting a Parameter

All variables defined as parameters in the project can be selected. Selection takes place the same way as with the hardware I/Os.

7.6.6 Selecting an Equipment Constant

All variables defined as system constants in the project can be selected. Selection takes place the same way as with the hardware I/Os.

7.7 Activating and Compiling a Function Chart

Several versions of an object can be saved in the **APROL** engineering database.

To let the system know which of the saved versions should be used to generate and later use a project, a particular version is **activated**. If the most recently saved version should be activated, select the **"Activate"** menu item in the object's shortcut menu, or **"Build"** in the toolbar/menu bar. This particular version must be activated in the version view of an object in the CaeManager engineering area. The version view can be opened by expanding the corresponding function chart with "+".

Compiling checks the object to see if it can be **generated**. Any configurations made for an object are checked for their plausibility and correct syntax. Compiling is done the same way as activation. Amongst others, the CFC C code of the controller and control computer is created in this process.

This is typical output when compiling a function chart.

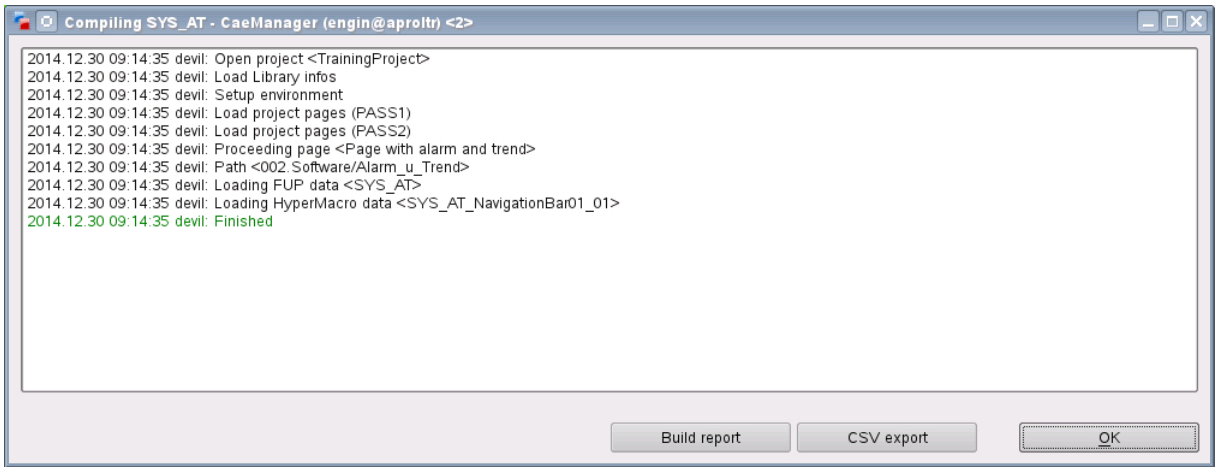


Figure 64: Typical output during compilation

Exercise 3:

Insert a new function chart into a directory you have created and give it a logical name. Assign the task to it (remember, the task must be unique and conform to the naming conventions). Change the size of the chart and get to know the different navigation possibilities.

Exercise 4:

Implement a blinking light which starts depending on a digital input.

Space for additions:

8 Build and Distribution

8.1 Versions

All function charts and other objects are subject to version management in **APROL**. This version management is handled automatically when saving a function chart. The version numbers are incremented automatically and cannot be manipulated. The versioning has quasi three digits The so-called "major number" and the so-called "minor number" can be influenced by the project creator.

8.2 Activation

During activation, you define an active version that is then used by the **"Compile"** function. This doesn't have to be the current version of an object. It is often the case that "temporary" versions are defined which are used for testing purposes. When doing so, you are able to continue working on the current version of a page at the same time without impeding a test.

Activation takes place with a right-mouse click on the desired version in the versions view of the project explorer.

The version view can be opened by expanding the corresponding object with "+".

The explorer displays all resulting versions of the selected object in the versions view.

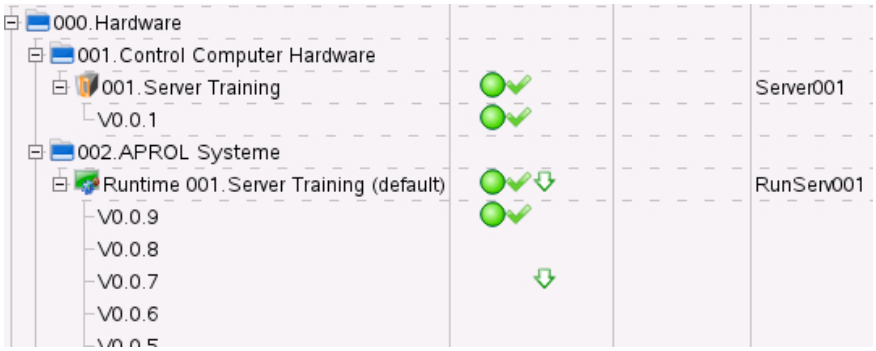


Figure 65: An object's version view

8.3 Compiling

Compiling takes place using the right-mouse button in the normal view or versions view on the activated version of a project page. This procedure executes various actions on the different types of pages of a project. These actions all fall under the general heading of "compilation".

The compilation id the basis for a successful build.

8.3.1 Hardware Compilation

When compiling hardware:

- All of the necessary configurations for the different hardware components is made available.
- Controller I/O addresses are determined for defined channels,
- Plausibility checks are carried out for connected hardware parts, power supply, and physical view assignments.

8.3.2 Function Chart Compilation

When compiling function charts, its programming language (function block language) is translated into a readable code (C code) and separated into programming units for the different target systems (controllers and control computers). This requires that the target systems are already established, i.e. have already been compiled. In addition, the libraries where the function chart gets it function blocks must be compiled.

Note: Code building

Temporary codes are stored in the project directory structure. See TM890 – The Basics of Linux.

8.3.3 Process Graphic Compilation

Process graphics are also subject to plausibility checks and released for generation.

8.4 The Confirmation

In addition to the various in-between versions of a project page, there are often older versions which are still functional. They can be identified accordingly with the **"Confirm"** function in the versions view of a page. This makes it possible to jump to an older (but maybe not as functional) version which is free of errors. Plants which require acceptance testing (see GAMP, FDA, etc.) are only allowed to run with confirmed versions after they have been accepted.

8.5 The Build

Different actions are carried out depending on the created instance.

For a controller, the programming units that result from compiling the function charts are merged with temporary code and interconnected with the tasks. These tasks are now in a state which can be loaded to the controller. A control computer must differentiate between the following configuration types:

- Logic contents of the function charts follow the same principle as described for the controller.
- Process control system function block configurations are separated (trend system, message system, etc.) and stored in databases.
- Information about the necessary communication between the control computer and the controller is put together in appropriate driver configurations.
- Process graphics and the dynamic diagram contents assigned to them are merged together and provided with the appropriate I/O supply.

All of this information is combined for the respective instances and is ready to be downloaded to them in this form.

The build procedure can be started with the "Build" menu item or from the shortcut menu of the project part in the navigation area.

There are four possibilities for this:

- **Build (configuration)** — Only the hardware configuration is generated
- **Build (task)** — The selected task is generated (only available in the physical structure)
- **Build (Task @ Target)** — Task of the selection hardware are generated (selection dialog)
- **Build all (Target)** — All tasks and the hardware configuration are generated

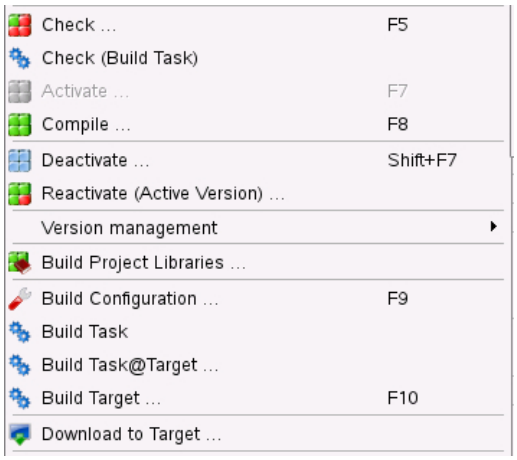


Figure 66: Starting the build process

Once the build has started, an output window lists all messages that occur during the process. Error messages are shown in red, warnings in orange.

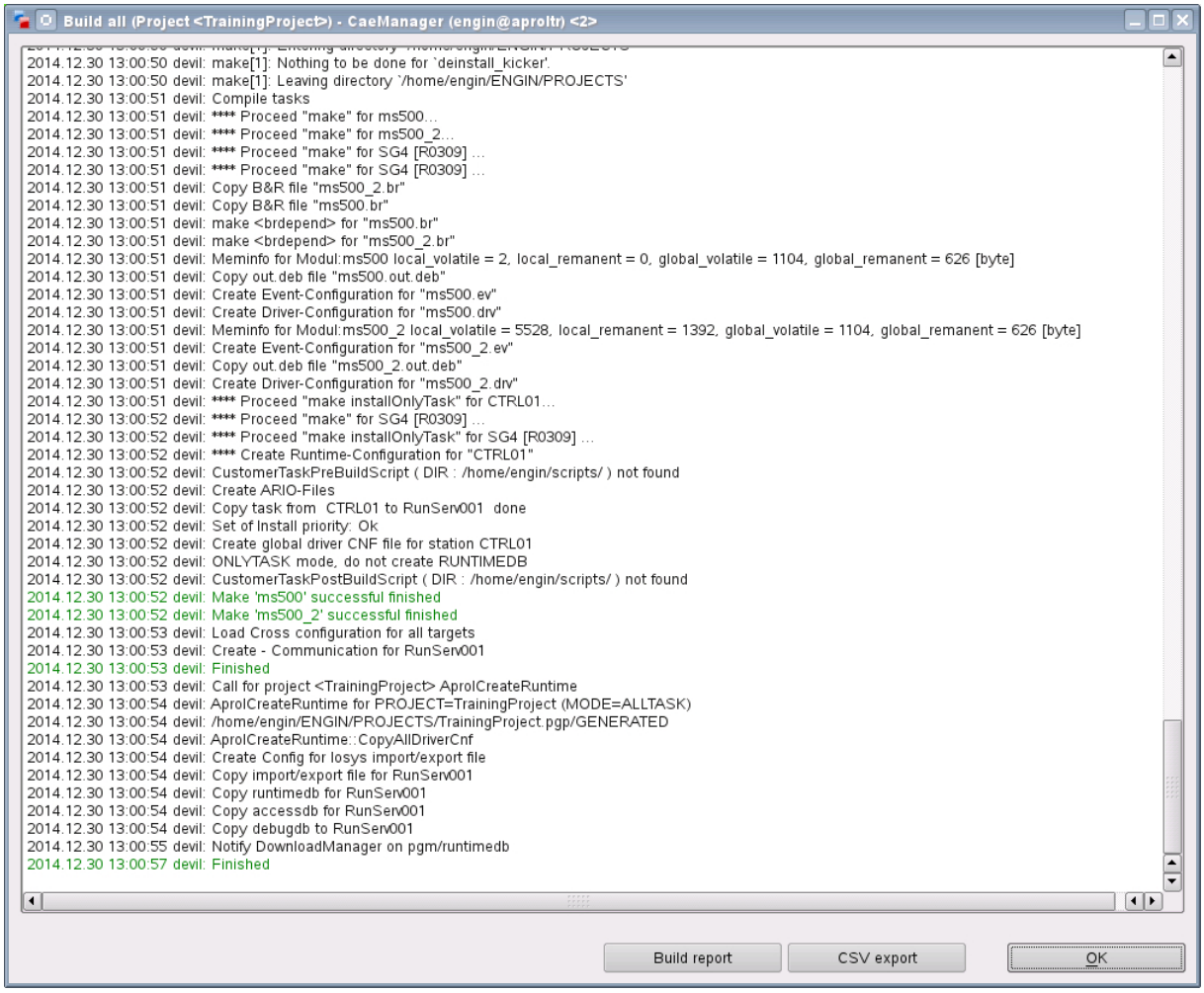


Figure 67: Output during a task build

8.6 Build all (Project) Menu Item

The **Build all (Project)** menu item is found under the **Build** menu in the CaeManager. If a project is open and you select this item, the program asks how the build should be carried out.

Note: Build all (Project)

Build all is useful if you are not sure if you have compiled and generated all of the necessary objects in a project. The APROL system also scans the engineering state and only compiles the necessary project parts.

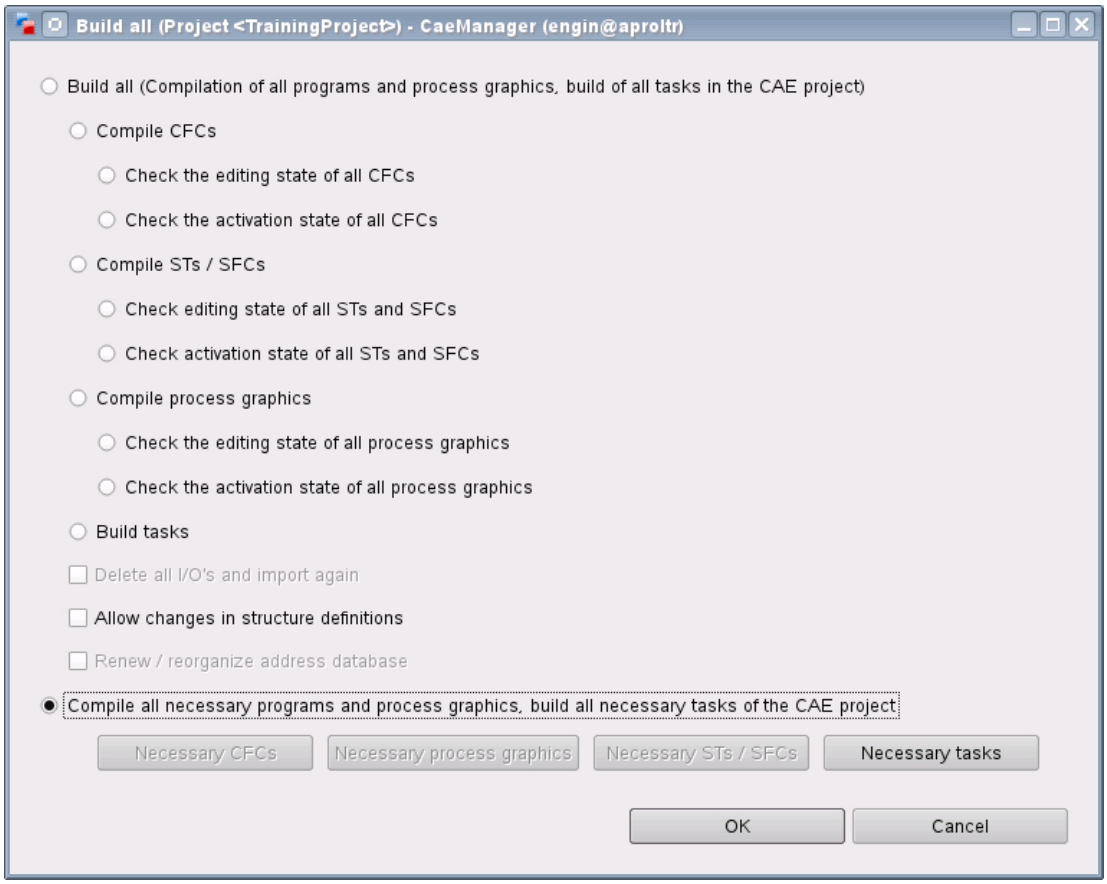


Figure 68: Selection possibilities for the build all (project)

In this example, all activated pages of the project are compiled, and the tasks for the controllers and (if present) the control computers are generated. The project is then ready to be downloaded to the different systems.

You can change the options so that pages are not recompiled or tasks are not created.

Note: Build all

This process can take some time when working with very large projects!

This menu item should only be used under the following conditions:

- You've lost track of the project and it takes too long to find out which plans or pages still need to be compiled.
- **APROL** has been updated.
- The menu item **"Compile and build necessary programs, process graphics and tasks of this project only"** is used so that only the changed objects are included. This means that the system automatically compiles the objects and tasks that are necessary. This provides the application developer with the best possible support and means that he does not have to keep track of what has to be compiled after his changes have been made. This is referred to as the "requirements list".

8.7 Downloading

Once all relevant project pages are compiled and all tasks needed for the project have been generated, the data created has to be transferred to the respective target systems (controller, runtime system, and operator system). The data is different for each target system. **Download** is the generic term used for transferring data in **APROL** (loading the generated project data from the engineering system "down" to the target systems). Downloading is handled with the **DownloadManager** program.

8.7.1 General Information About Downloading

The following data is transferred to the respective target systems:

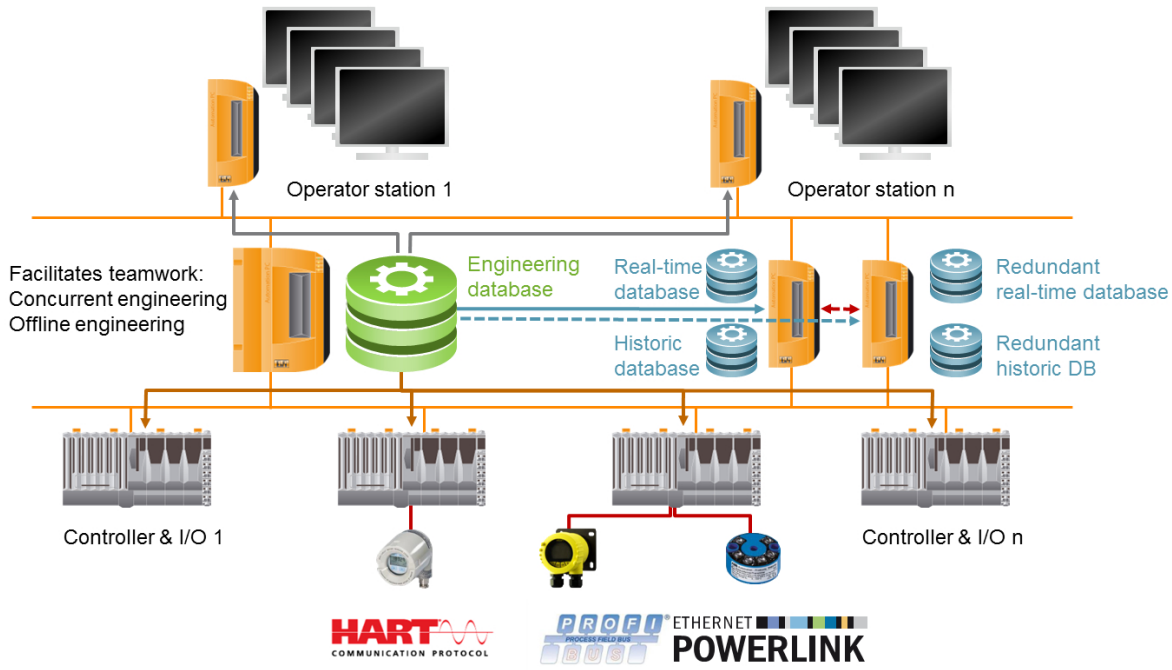


Figure 69: Download schematic

- Target controller
  Tasks, B&R libraries, customer libraries, and the system configuration for the controller CPU can be loaded to a controller. The tasks include the I/O structures (the connection between hardware I/O and PV).

- Runtime system target (control computer)
  The following information (files) is provided to the control computer:

Information	Note
Control system modules	The **APROL** programs which should run on the control computer in the runtime system.
Control computer task	Task(s) (generated from charts for the control computer and parts thereof) which have been created in the runtime system for operation and monitoring.
Configurations	Alarm, message, trend data, and ChronoLog data acquisition configurations (the configuration is gotten from the wiring of the corresponding control system function blocks).
User management	Information pertaining to user management (operation and monitoring permissions, login mechanisms).

- Operator system target (operator station)
  The following information (files) is provided to the operator station:

Information	Note
User interface applications	Programs provided for operation and monitoring on this station.
Process graphics	The visual interface for the process.
User management	Information pertaining to user management (operation and monitoring permissions, login mechanisms).

8.7.2 Downloading After Changes

The effects of changes when engineering the project can be very different and make it necessary to have various download procedures.

After changes have been made to the logic content of a function chart (and therefore a task), it may be that a simple task download to the controller is sufficient. However, if this change has an effect on (i.e. changes) the communication behavior of the task, then downloading to the controller is not enough. A changed I/O image on the controller always requires the process bus driver to be reconfigured.

Note: Download

All objects can always be registered for downloading, because only changed tasks are loaded.

Note: Download during operation

To prevent risk and to guarantee smooth operation of the system, be sure to carefully think through any type of download to a system that is running.

A bumpless download is always possible after logic changes. This should be taken into account when changing configurations.

9 The DownloadManager

The DownloadManager is used to download **APROL** engineering to the target system. The Download-Manager is started over the desktop, the KDE menu, or directly from the CaeManager.

In principle, there are 4 different types of download:

- **Operator download:**
  This is the download to the operator stations. In this case, all visual elements and online user management information is loaded to the operator environment.

- **Controller download:**
  This is the download of the compiled tasks and libraries (.br files) to the respective controllers.

- **Runtime download:**
  This is the download to the runtime environment from APROL. During this download, all background processes such as the control computer task, INA driver, AlarmServer, TrendServer etc. are loaded with the new configurations.

- **Gateway download:**
  This is the download to the gateway system. This is usually performed when connecting to 3rd-party systems.

The download settings (for different target systems (controller > IP address operator station > computer and target system (user)) are stored in the project engineering section of the CaeManager and made available for the DownloadManager when the task is built. These settings can also be changed temporarily if necessary.

Right-click on an object to prepare it for download. The explorer view for this object can then be opened to specifically load just one part of the object (e.g. only one controller task).

The download could appear as follows:

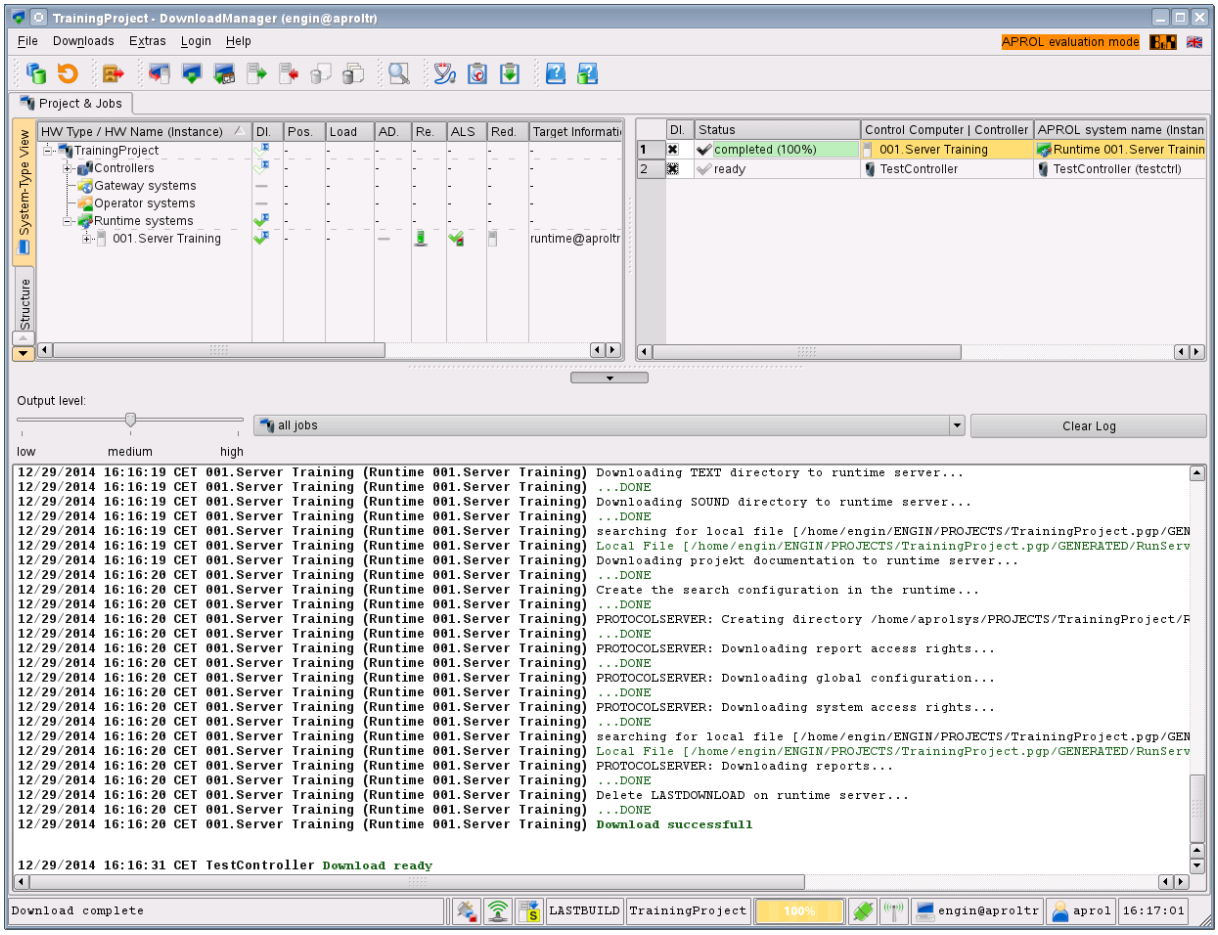


Figure 70: Download complete

Note: Error messages

Please take special care to check for any error messages occurring in the system.

A detailed description of the DownloadManager is provided in the **APROL** help documentation. Any additional information required can be found there.

The system is ready for operation after a successful download. The **StartManager** (started from the desktop or from the KDE menu) provides information about which processes are currently running and indicates their status as well. This tool can also be used to intervene in these processes.

Exercise 5:

Generate all of the necessary tasks so that the function chart you've created works and then distribute them to the relevant target systems.

Space for additions:

10 Chart Debugging

When testing before commissioning takes place, it is often necessary to track the information flow of the system more precisely to detect any configuration errors that may occur. For this reason, we have made it possible to debug function charts in the CaeManager. This debugging shows all connectors and variables which are transferred to the process control system directly in the chart itself.

Select this debugging icon from the toolbar to debug a function chart.

Figure 71: Switching debugging on and off

Note: Installed environment

Debugging only works if a run-capable Runtime environment is also installed. Compile and generate your configuration and execute a download to the controllers and the control computers.

Current values are shown next to the respective chart and block inputs and outputs.

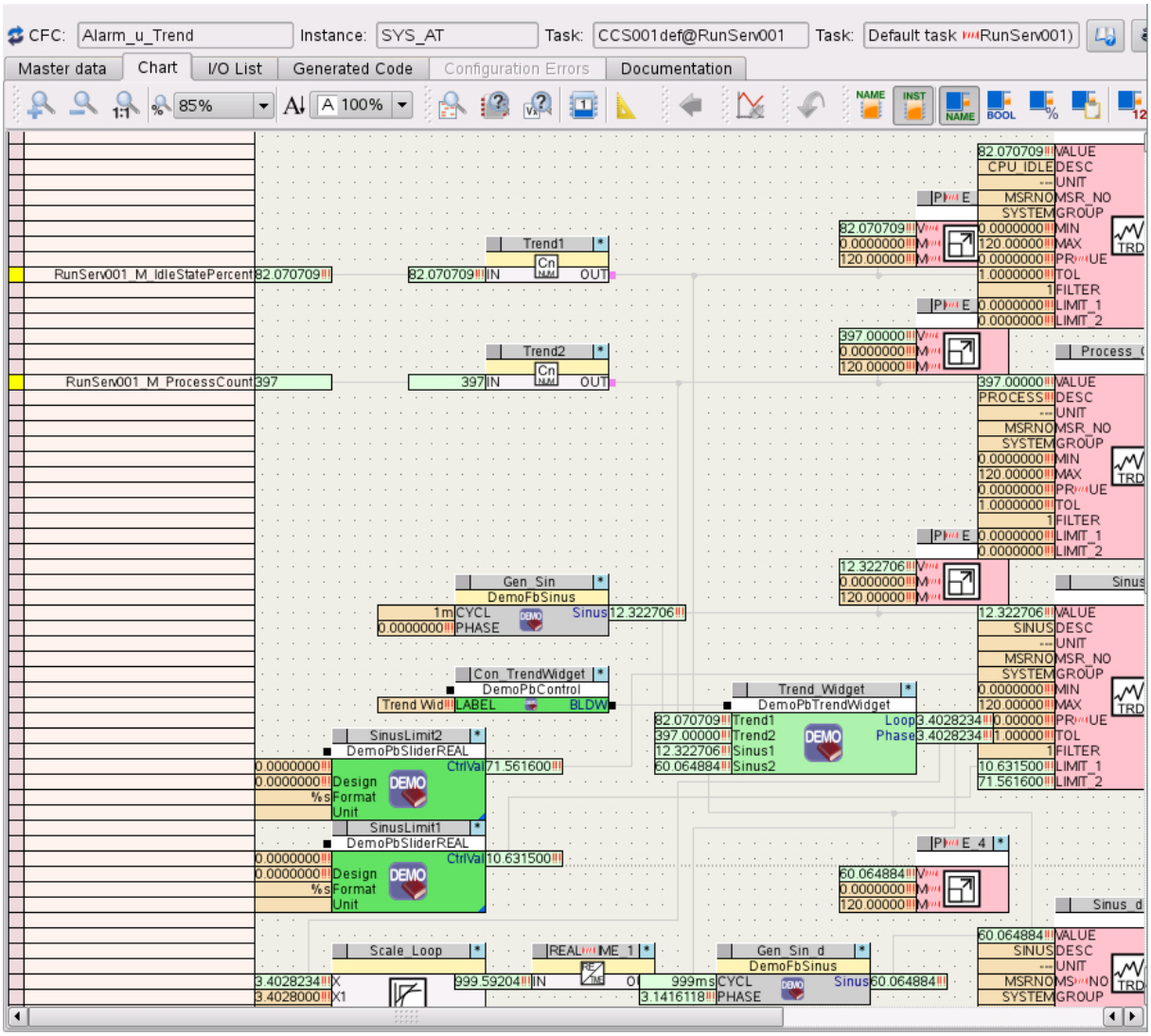


Figure 72: Function chart debugging

10.1 Setting Values

To set a value, select the respective supply line or a pin in the function chart with the right-mouse button and use the **"Change I/O"** function. The window which appears will allow you to modify the current value. Clicking on the **"Set"** button writes the desired value to the Runtime System.

Note that setting variables which represent the result of calculations may not always show the desired result. After one cycle, the value will be overwritten again with the new result of the calculations. Whether or not the value you set has a noticeable impact depends on how your function charts have been engineered. It is useful to use setting for e.g. testing exceptions to see what happens when exceeding limit values. You are even able to simulate what might occur with impossible values and values that don't make sense.

11 Creating a Process Control System Application

Creating a process control system application involves configuring a visualization application that has the ability to collect alarm, trend and other types of data. The following pages will help illustrate this.

11.1 Creating Process Graphics

After the function charts have been designed, it is time to arrange the process graphics. Process graphics form the interface (HMI) between the operator (operating personnel) and the process control system. Process graphics with the best possible structure allow the user to actively intervene, configure and react to running processes. This is done using the various dynamic element and input fields.

Process graphics are created in two steps. In the first step, you draw the static background or import an already finished image to be used as such. The second step entails linking the image with graphic blocks from a function chart (mostly embedded in a hyper macro). This is more of a graphical interpretation with the respective dynamization. The signals are already linked in the CFC (DCS functionality). A direct dynamization would also be possible. In this case, a link to the variables must be ensured in the image (SCADA functionality).

A process graphic is created in the directory that you have already selected with the **"Create part / Process Graphic"** menu item. A graphic **name**, description, and **instance** must now be entered.

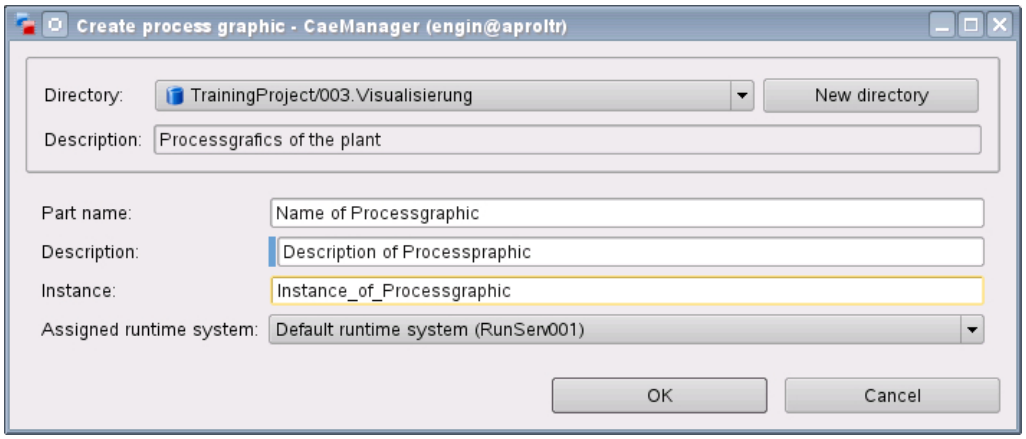


Figure 73: Creating a process graphic

Note: Instance

When specifying instance names for process graphics, the naming conventions for the C programming language must generally be adhered to. Otherwise, problems could occur during the build. The instance is required for functions such as switching graphics and graphics for alarms (see the help files for the AprCcAlarm block).

The static part of the process graphic (background image) mostly reflect the P&ID diagram (piping & instrumental diagram). There is no limit to the number of diagrams used.

According to R&I guidelines and the corresponding technical system subdivision principle, the procedure should be distributed among several clear process graphics which can be linked to one another. (switching graphics with graphic calls (graphic connect) graphic instance)

After opening the process graphic (double-click on the object), the **"DisplayEditor"** graphic editor is opened with another double-click (or right mouse button **"Edit"**) on the process graphic preview.

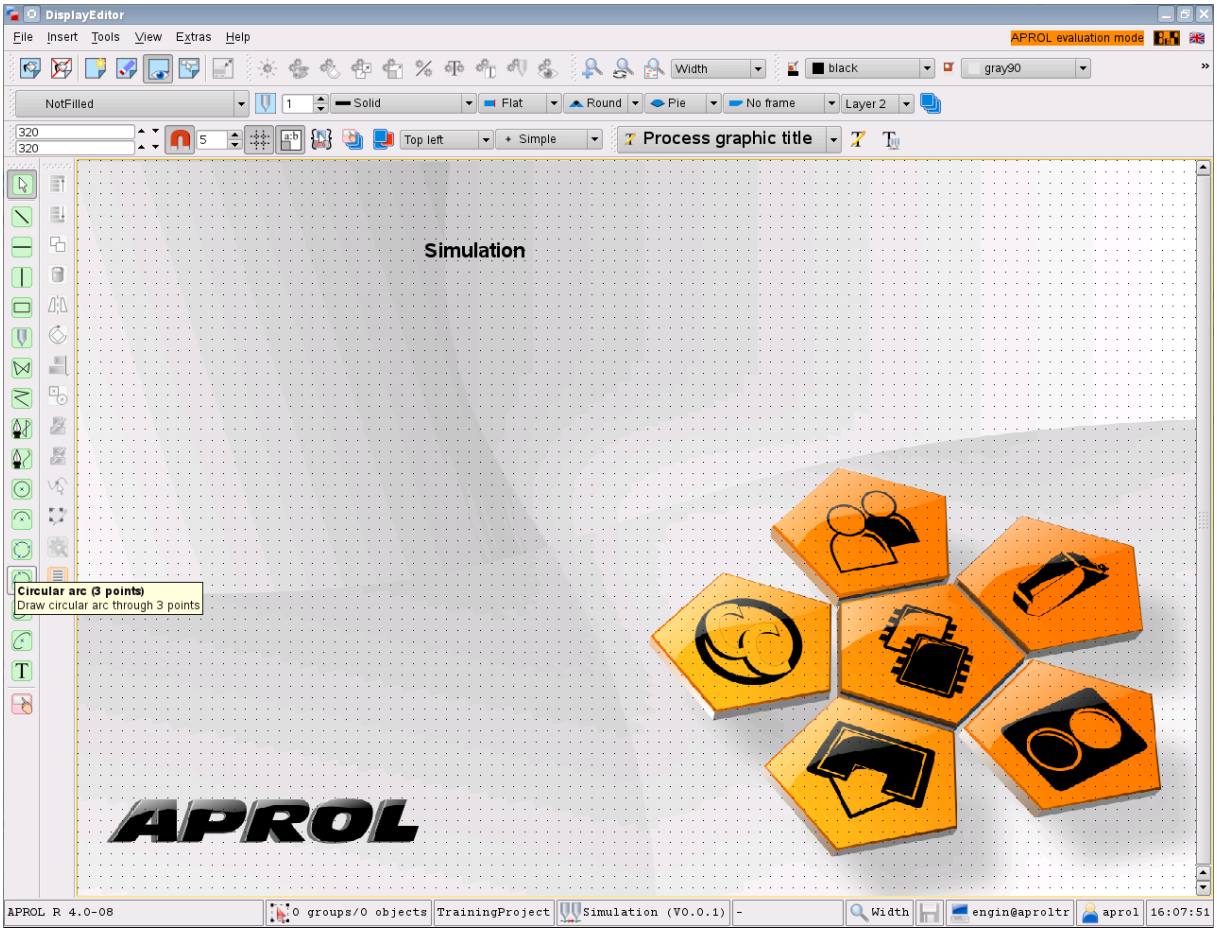


Figure 74: DisplayEditor

Some of the DisplayEditor functions:

- General drawing functions

Figure 75: DisplayEditor toolbar

- Working with layers

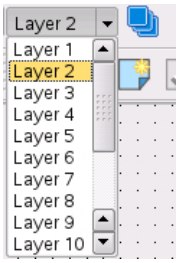


Figure 76: Layering

- Multi-page process graphics

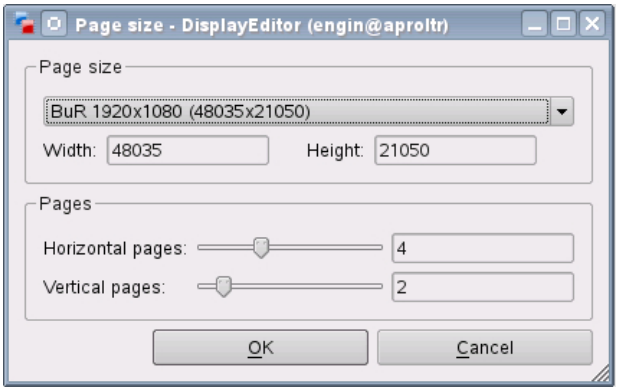


Figure 77: Configuration of a multi-page process graphic

A multi-page process graphic can be created with the **"File / Page Size"** menu item.
This can be very useful to get a complete view of a system. Process graphics can now be displayed on multiple screens in the operating environment (multi-screening).

- Insert background images
  The DisplayEditor makes it easy to integrate images that were created using different tools (e.g. CorelDraw, etc.). The images are often defined in the pre-planning phase and are of course prepared in graphical form for the potential functional specification and eventual design specifications.

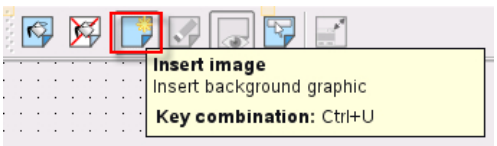


Figure 78: Adding the background picture

If you click on the file selection button in the next dialog box, then the project-specific Image Container will open. This has to contain all of the images that will be used in the project.

It is only necessary to create an image once with the Image Container, so that an internal link is created each time the image is used. This saves memory and increases performance.

It is possible to fill up the Image Container by simply dragging and dropping images from the file system. Once this is done, the images can be used immediately.

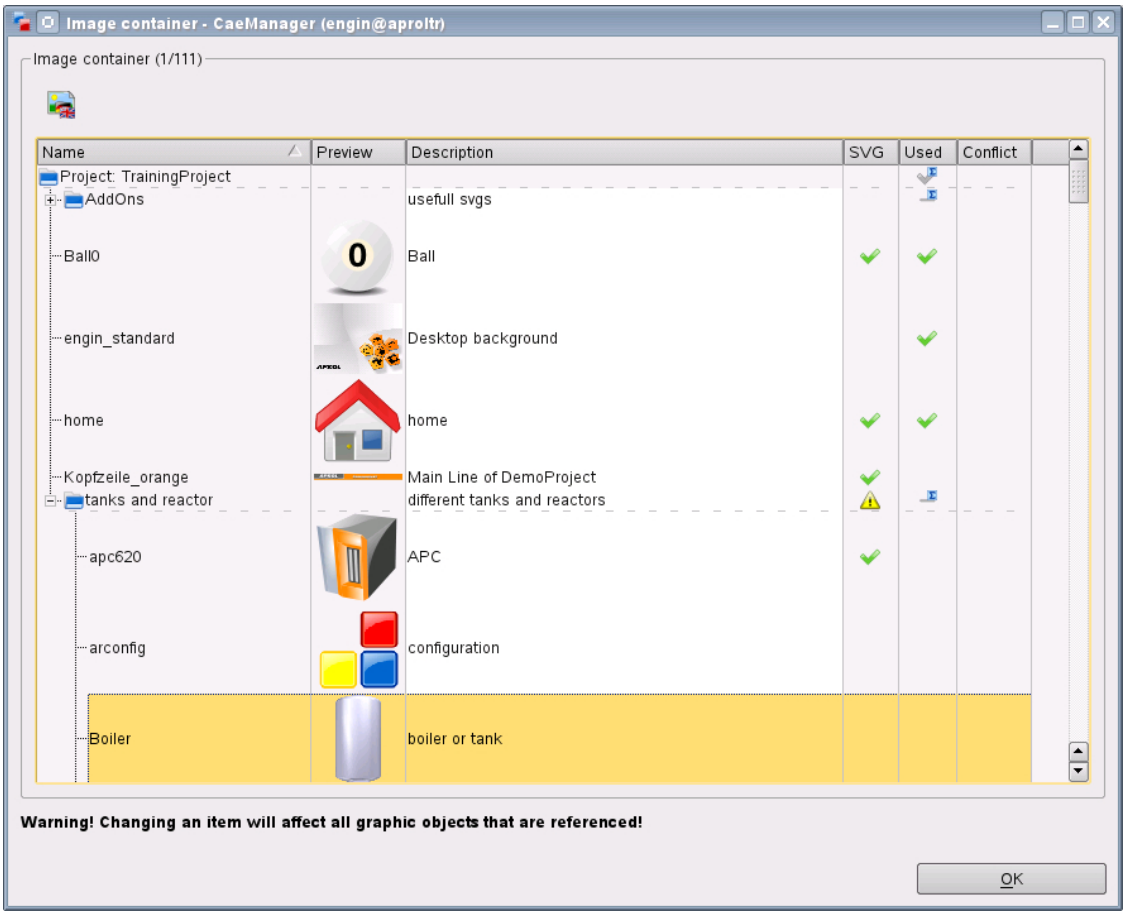


Figure 79: Image Container

- Adding graphic objects (e.g. logos, etc.)
  It can also be helpful to use pre-designed graphic objects in the process graphic. They are also selected from the Image Container.

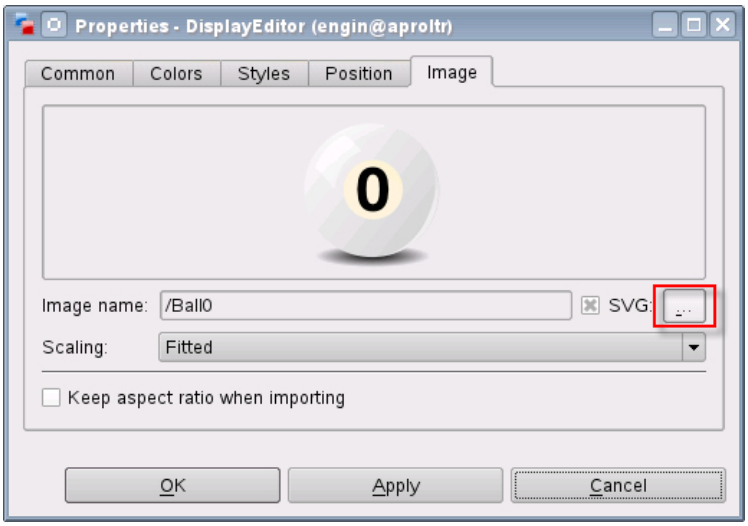


Figure 80: Adding graphic objects

- Interactive element
  Most controls and widgets are used in library engineering. Nevertheless, at least the "Load-PictureControl" function is usually required in project engineering to change from one process graphic to another. The instance of the process graphic which should be shown is entered here. Working with and creating controls and widgets will be covered in a later seminar (TM812).

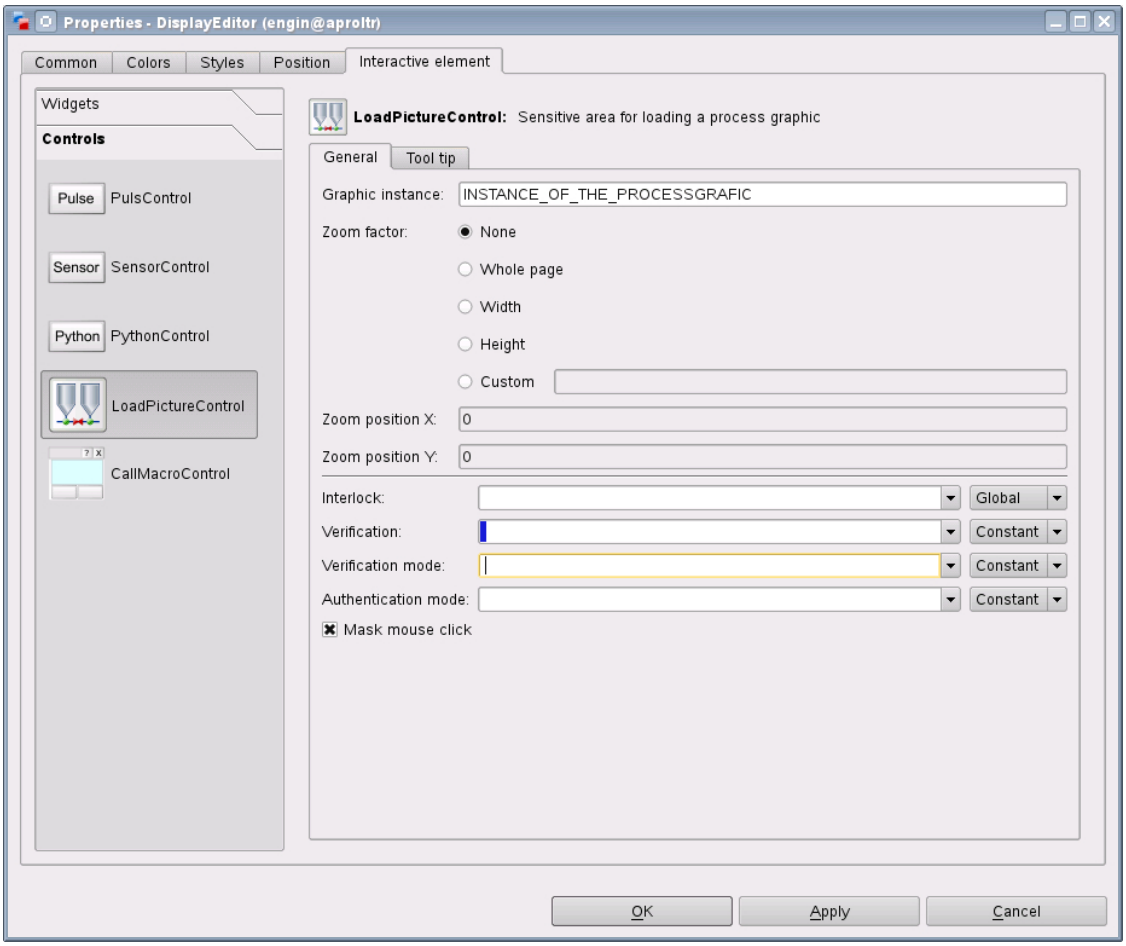


Figure 81: Setting up an image

This list represents only a small portion of all the different possible functionalities. As with other graphics programs, you must become somewhat familiar with the intuitive operation of the DisplayEditor.

Detailed information can be found in the **APROL** online documentation.

Exercise 6:

Create a process graphic. It's up to you to decide whether you want to draw a picture in the DisplayEditor, integrate a different graphic from somewhere else or use a combination of both possibilities. Carry out a build and load the process graphic to the operator system using the Download Manager. The RUNTIME environment can then be started and the process graphic viewed here.

Space for additions:

12 Hyper Macros / Typicals

12.1 What is a Hyper Macro?

Using CAE (Computer-Aided Engineering) allows a considerable amount of time to be saved engineering a project whenever it is possible to standardize a system or the tasks. In practice, you will try to find the greatest common denominator for all drives and measuring points on a system. Consequently, all installed drives (pumps, stirring devices, motors, fans, etc.) will be handled the same for the most part. In this way, for example, not only is manual operation of all installed drives the same, but all the faceplates are the same as well. Only the locking conditions change from case to case.

Measuring points can be roughly broken down into two groups: analog measuring points with number displays (which can always be the same), and binary measuring points like float switches which only have one signal lamp.

So we save time if we only create identical functions once and then make them available anytime. To solve this task, APROL provides **hyper macros**.

Hyper macros explained:

A hyper macro is a function chart which has been turned into a logic function block. **Functionalities** for the **controller**, the **control computer**, the **dynamic visualization**, **alarms**, **trends**, and **bus traffic** for a typical application (e.g. installation drive) are fully programmed on this type of function chart.

There is no assignment to actual hardware I/Os at this time because the block should be able to be used universally. The placeholders on the chart input and output borders are the pins of the finished block, which are then linked with the corresponding process variables. In the project, a hyper macro is used like any other block, and only the input and output images are read or written.

A very fast configuration process results from this since the logic for a motor is handled by one hyper macro in the function chart. It supplies, configures, and then only needs to store the dynamic part to the process graphic. This is all there is to engineering a typical.

Hyper macros are embedded in libraries and can therefore be taken from them for configuration purposes (as explained for function chart engineering).

Hyper macros also have their own **"Properties" (also the middle mouse button)** with a few different configuration options since pins can be connected and hyper macro constants can be used. Variables are used as hyper macro constants (defined during library engineering) and may not be changed online in the RUNTIME environment.

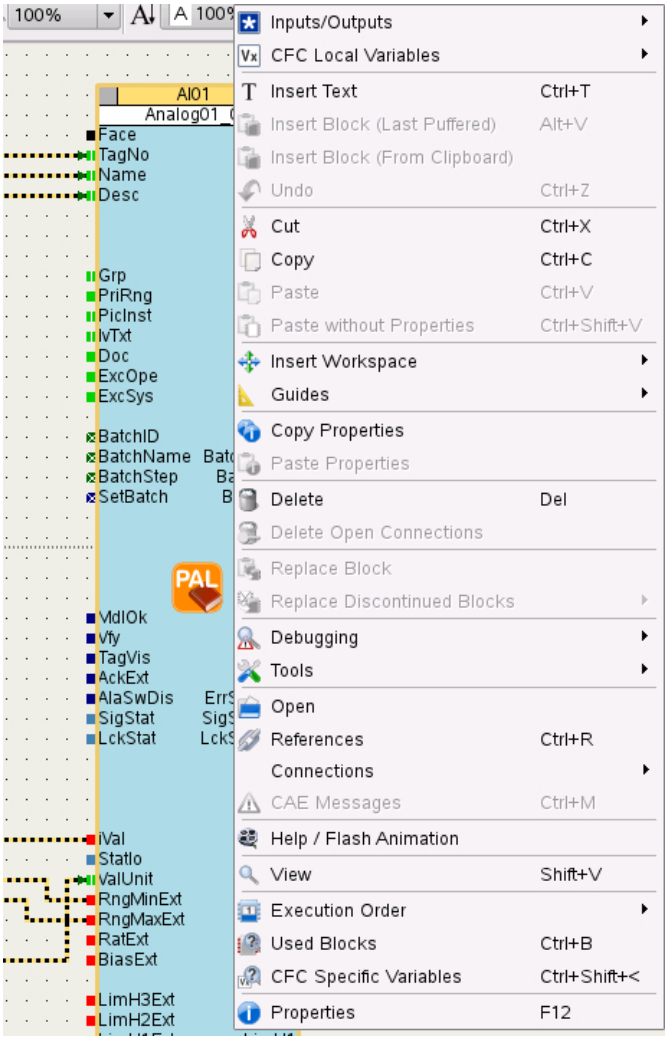


Figure 82: Opening hyper macro properties

A different functionality makes it possible to use hyper macros within hyper macros in … . This makes it possible to implement typicals for functionalities that are used repeatedly in a project in order to simplify and speed up the engineering effort.

Note: Hyper macros within hyper macros

During library engineering, the instance names of blocks used in the hyper macro should be as short and concise as possible so that they can be nested as deeply as possible.

If a hyper macro has been successfully configured and supplied with parameters, then the only thing missing is the "link" between the dynamic diagram elements on the function chart and an existing process graphic. To do this, open the process graphic and select a dynamic visualization element from a function chart underneath the second tab and place it on the process graphic **drag-and-drop**.

There is a grid available with a "snap-to-grid" function to help you place the dynamic diagram element precisely. Any elements placed can be scaled, aligned, and rotated via their properties menu.

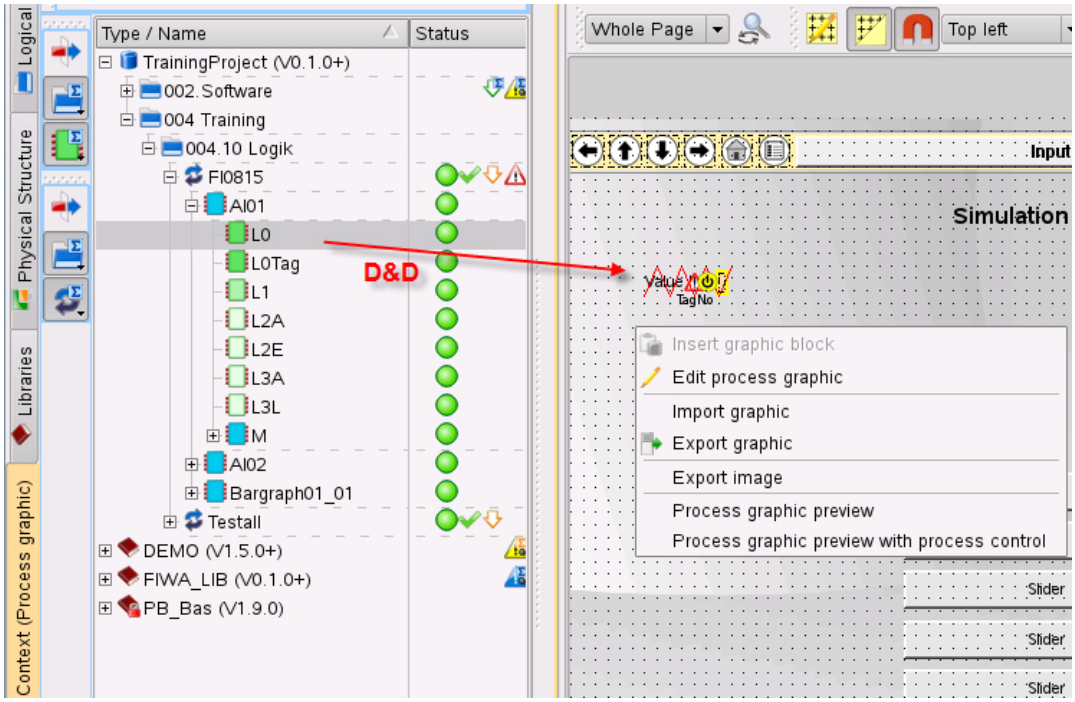


Figure 83: Linking logic and the process graphic

Exercise 7:

Place the Analog01 hyper macro from the PAL library on a function chart you have created and configure it. Refer to the hyper macros help files when doing this. *Place the internal picture macro "L0" onto your process graphic.*

Additional steps: Build phase, download phase, RUNTIME environment > Check.

Space for additions:

13 Summary

What have we achieved so far?

The CaeManager, as the central engineering tool, is the core of the engineering environment. For this reason, it is important to be knowledgeable of its structure. The **APROL** help system should always be the first place you look if problems occur.

All of the aspects of project engineering are handled using this tool. Work is faster and more efficient as a result. These advantages become even more evident when the user fully understands the power of the hyper macro concept.

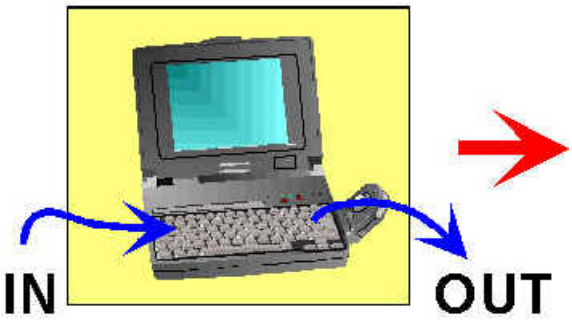


Figure 84: So much input – Now it's off to the RUNTIME environment!

13.1 Objectives Checklist (Short Questions on the Topic)

Which basic configurations need to be carried out in the CaeManager when starting a project?

What is the significance of the instance?

What target systems are there?

What is a local variable?

What is a controller-to-controller connection needed for?

What is a hyper macro?

What is a dynamic visualization element?

13.2 Solutions to the Objectives Checklist

Please take note of the necessary additional information.

14 Appendix

Seminars and training modules

The Automation Academy provides targeted training courses for our customers as well as our own employees.
At the Automation Academy, you'll develop the skills you need in no time!
Our seminars make it possible for you to improve your knowledge in the field of automation engineering.
Once completed, you will be in a position to implement efficient automation solutions using B&R technology. This will make it possible for you to secure a decisive competitive edge by allowing you and your company to react faster to constantly changing market demands.

Automation Studio seminars and training modules

Programming and configuration	Diagnostics and service
SEM210 – Basics SEM246 – IEC 61131-3 programming language ST* SEM250 – Memory management and data storage SEM410 – Integrated motion control* SEM441 – Motion control: Electronic gears and cams** SEM480 – Hydraulics** SEM1110 – Axis groups and path-controlled movements** SEM510 – Integrated safety technology* SEM540 – Safe motion control*** SEM610 – Integrated visualization*	SEM920 – Diagnostics and service for end users SEM920 – Diagnostics and service with Automation Studio SEM950 – POWERLINK configuration and diagnostics* If you don't happen to find a seminar on our website that suits your needs, keep in mind that we also offer customized seminars that we can set up in coordination with your sales representatives: SEM099 – Individual training day Please visit our website for more information****.****: **www.br-automation.com/academy**

Overview of training modules

TM210 – Working with Automation Studio TM213 – Automation Runtime TM223 – Automation Studio Diagnostics TM230 – Structured Software Development TM240 – Ladder Diagram (LD) TM241 – Function Block Diagram (FBD) TM242 – Sequential Function Chart (SFC) TM246 – Structured Text (ST) TM250 – Memory Management and Data Storage TM400 – Introduction to Motion Control TM410 – Working with Integrated Motion Control TM440 – Motion Control: Basic Functions TM441 – Motion control: Electronic gears and cams TM1110 – Integrated Motion Control (Axis Groups) TM1111 – Integrated Motion Control (Path Controlled Movements) TM450 – Motion Control Concept and Configuration TM460 – Initial Commissioning of Motors TM500 – Introduction to Integrated Safety TM510 – Working with SafeDESIGNER TM540 – Integrated Safe Motion Control	TM600 – Introduction to Visualization TM610 – Working with Integrated Visualization TM630 – Visualization Programming Guide TM640 – Alarm System, Trends and Diagnostics TM670 – Advanced Visual Components TM920 – Diagnostics and service TM923 – Diagnostics and Service with Automation Studio TM950 – POWERLINK Configuration and Diagnostics TM280 – Condition Monitoring for Vibration Measurement TM480 – The Basics of Hydraulics TM481 – Valve-based Hydraulic Drives TM482 – Hydraulic Servo Pump Drives TM490 – Printing Machine Technology In addition to a printed version, our training modules are also available on our website for download as electronic documents (login required): Visit our website for more information: **www.br-automation.com/academy**

Process control seminars and training modules

Process control standard seminars	Process control training modules
SEM841 – Process Control Training: Basic 1 SEM842 – Process Control Training: Basic 2 SEM890 – Advanced Process Control Solutions	TM800 – APROL System Concept TM811 – APROL Runtime System TM812 – APROL Operator Management TM813 – APROL XML Queries and Audit Trail TM830 – APROL Project Engineering TM890 – The Basics of LINUX Visit our website for more information: **www.br-automation.com/academy**

* SEM210 - Basics is a prerequisite for this seminar.
** SEM410 - Integrated motion control is a prerequisite for this seminar.
*** SEM410 - Integrated motion control and SEM510 - Integrated safety technology are prerequisites for this seminar.
**** Our seminars are listed in the Academy\Seminars area of the website.
**** Seminar titles may vary by country. Not all seminars are available in every country.