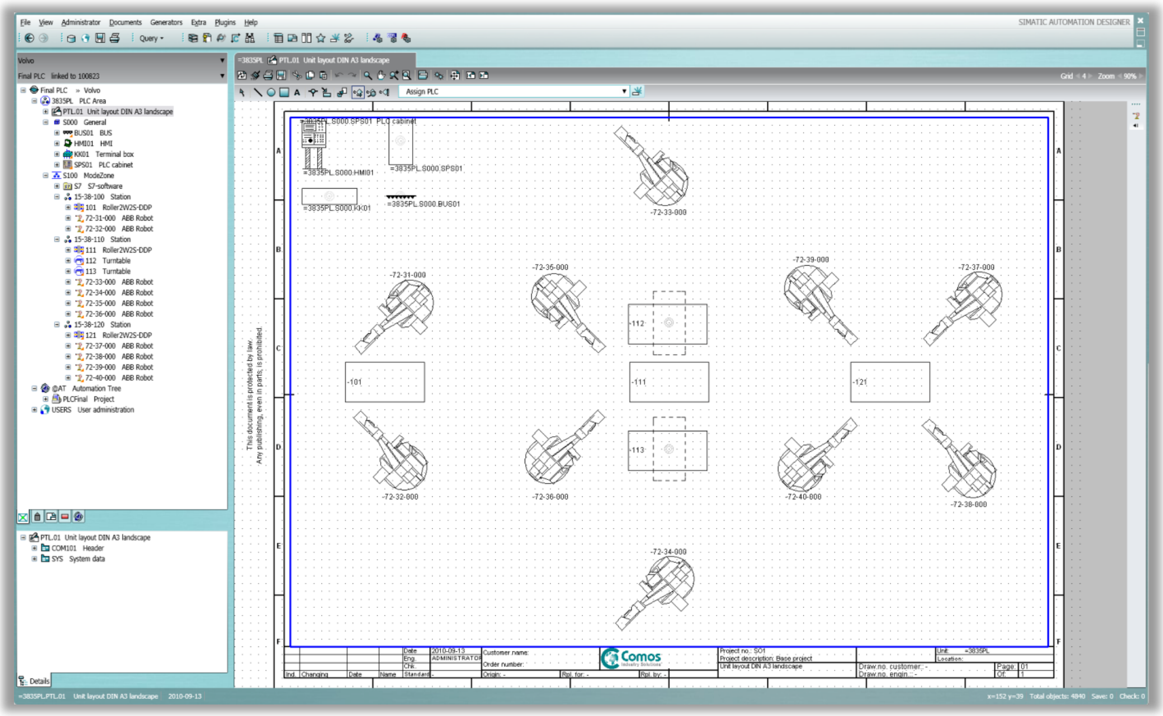
CHALMERS



Automatic generation of PLC programs using Automation Designer

Based on simulation studies and function block libraries

Master of Science Thesis in the Master Degree Program, Production Engineering

Mikael Andersson
Erik Helander

Department of Signals and Systems
Division of Automation
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2010
Report No. EX063/2010

Automatic generation of PLC programs using Automation Designer
Based on simulation studies and function block libraries

Mikael Andersson
Erik Helander

Examiner: Petter Falkman

Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31-772 10 00

The cover picture shows a 2D layout inside Automation Designer

Department of Signals and Systems
Göteborg, Sweden October 2010

Abstract

New virtual commissioning tools are constantly being developed to help production engineers in their line of work. One of these new tools is Siemens Automation Designer. A goal for VCC (Volvo Car Corporation) is to be able to do the whole commissioning process virtual and not have to test the programs in the real factory many times. Without several real time tests the ramp up time could be decreased significantly and more time could be spent on producing cars.

An important issue which needs be solved is the possibility to reuse information. By using Automation Designer reusable templates, containing information that always stays the same, can be created. The templates can also include scripts which can ensure that naming standards is kept. Depending on what information the templates are built up with, Automation Designer is able to generate PLC code and HMI screens.

In this report three stations at VCC in Torslanda, Göteborg, is used in a case study to build up templates inside Automation Designer. The goal with the study is to test if PLC code and HMI screens can be generated from the software and still be according to the VCC standard. Information from earlier simulation studies in Process Simulate is also imported into Automation Designer to see which information that can be reused.

The final result is a generated PLC project where code is built up and generated by Automation Designer and library function blocks are collected from the VCC function block library.

Keywords: PLC code generation, HMI screen generation, Automation Designer

Acknowledgements

The master thesis had great support from the guys working with Automation Designer at Siemens AG Industry Automation Division in Germany. A special thanks to Michael Geck for his help and support throughout the whole project. We would like to thank VCC for giving us the opportunity to do our master thesis at the company. Especially John Selander, our supervisor at VCC, and Stefan Axelsson for helping us with administration and necessary education for the master thesis. We would also like to thank Mathias Sundbäck at VCC for giving us information about virtual simulation.

Last but not least we would like to send a big thank you to our examiner at Chalmers University, Petter Falkman, who helped us to find the master thesis and has given us feedback through the master thesis.

Table of Contents

List of Figures

List of Tables

Abbreviations

AD	Automation Designer
CAD	Computer Aided Design
FBD	Function Block Diagram
FFI	Fordonsstrategisk Forskning och Innovation (Strategic Vehicle Research and Innovation)
HMI	Human Machine Interface
PD	Process Designer
PLC	Programmable Logic Controller
PS	Process Simulate
S7	STEP 7, Siemens' PLC programming tool
TCP	Tool Center Point
VCC	Volvo Car Corporation
VINNOVA	Verket för Innovationssystem (The Swedish Governmental Agency for Innovation Systems)

1 Introduction

Product lifecycle management (PLM) is a business strategy that helps companies to share and store information of products and common processes to facilitate the development of new ones. Virtual commissioning, which is the programming and testing of PLCs, robots and other in a virtual environment, is a large part of PLM and it is increasing in the large production companies due to its ability to decrease ramp up times (Smith, 2009). New virtual commissioning tools are constantly being developed to help production engineers in their line of work. One of these new tools is Siemens Automation Designer.

1.1 Background

Preparing an automatic production system takes a lot of time and to be able to decrease this time virtual simulation studies are used more frequently. Traditionally a lot of work is done twice because of the lack of integration between software and systems. For a virtual simulation study all logic is needed to be created to be able to do a proper simulation and to investigate the station behavior. However, this logic cannot be transferred to the PLC software directly.

When creating the PLC program the station logic from the simulation study cannot be satisfactory reused. A step with manual transfer of the logic is needed which leads to both double work and possible human errors. It is also hard to verify the PLC code before the real station is built in the factory because of the lack of integration with the virtual simulation study.

A goal for VCC (Volvo Car Corporation) is to be able to do the whole commissioning process virtual and not have to test the programs in the real factory many times. Without several real time tests the ramp up time could be decreased significantly and more time could be spent on producing cars.

This master thesis is a part of the FFI (Strategic Vehicle Research and Innovation) initiative for VCC. FFI is a research program launched by the Swedish governmental agency for innovation systems (VINNOVA), to help the Swedish car production companies develop a competitive and sustainable vehicle industry (Vinnova, 2010).

1.2 Purpose

The purpose is to generate PLC code and HMI (Human Machine Interface) screens in an earlier phase of the commissioning. Also to be able to reuse information created in Process Simulate, stored in a common database and decrease the ramp up time by using the software Automation Designer.

1.3 Objectives

To generate PLC programs and HMI screens with Siemens Automation Designer version 9.1, based on already created specifications and generic function libraries.

Then evaluate how well the program coincides with the programming structure VCC uses today and through the evaluation come up with suggestions how Automation Designer and VCC work methodology can be changed to work in conjunction.

Another aim of the project is also to learn Automation Designer well enough to be able to educate VCC personnel in the basics of the software.

1.4 Delimitations

- EPLAN drawings will not be a part of this project.
- PLC code and HMI screens will only be generated for Siemens products.
- Generated code shall be based on existing VCC libraries.
- The stations 15-38-100, -110 and -120 in the TA-factory at VCC Torslanda are the only stations that will be used as a case study in Automation Designer.

1.5 Research questions

- What is possible to do with Automation Designer today?
- What does VCC need to change in their way of working to be able to use Automation Designer?
- What needs to be changed in Automation Designer to fit VCC work methodology?
- How much information can be transferred from Process Simulate to Automation Designer?
- To what degree can VCCs current PLC function blocks be integrated with Automation Designer?

2 Station 15-38-100, 110, 120 at VCC Torslanda

The stations named 15-38-100, 15-38-110 and 15-38-120 are situated in the TA factory at VCC in Torslanda, Gothenburg Sweden. These three stations are included in the case study for this master thesis.

2.1 Layout of the three stations

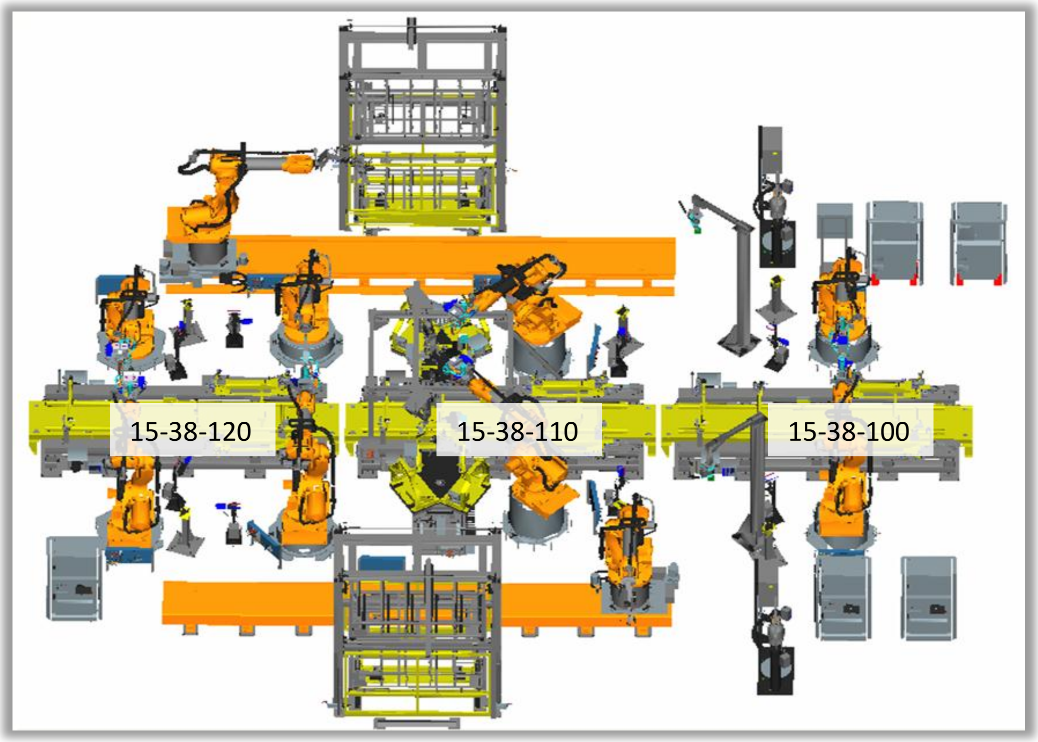


Figure 1 – Process Simulate layout of the stations included in this master thesis

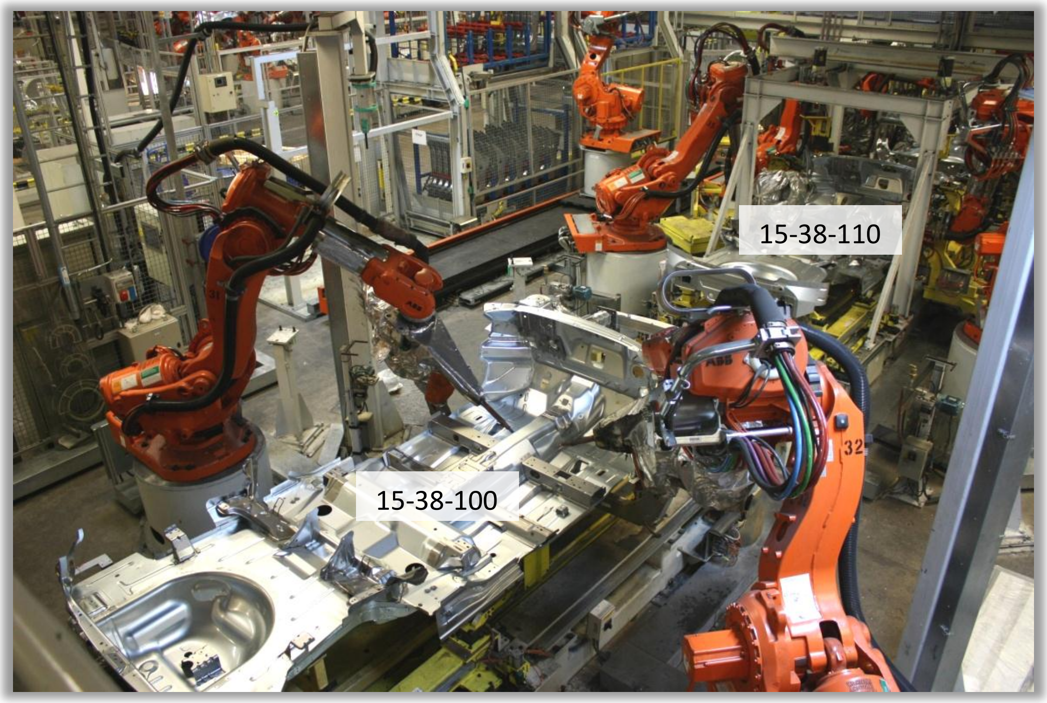


Figure 2 - Real layout of the stations included in this master thesis

2.2 Work description

The stations shown in Figure 1 are a part of a line producing the underbody for the car models V70 and S80. The main operations for the three stations are spot welding performed by robots. The PLC controls when a robot is allowed to work, however the PLC does not handle the robot operations. When a robot has been given the confirmation that it is allowed to start the robot is completely driven by the robot program until the operations are finished. The robots also control the special equipment they use during their work, as for example the gluing guns in station 15-38-110.

Following is a detailed description of each station. The information presented is only concerning the PLC, since this is the focus of this master thesis.

2.2.1 Station 15-38-100

Station 15-38-100 consists of two spot-welding robots, one conveyor, one decoder and one lift (see Figure 1). The robots perform enhanced spot-welding on parts assembled onto the underbody in the previous stations. A decoder reads a tag to check if the arriving model is the same as expected. The tag is placed on the skid that carries the car body, to determine which model it is and to be able to track it through the production.

To be able to handle different car models in this station there are two steering pins that needs to adjust position relative to the arriving underbody. This is done by cylinders controlled by the PLC before a new underbody arrives to the station.

Below is a brief description of the PLC operations in station 15-38-100:

1. Get information of which model is arriving from the previous station or buffer
2. Adjust the steering pins
3. Start the conveyor to bring in a new underbody to the station
4. Lower the underbody down to fixate it for the robot welding
5. Start the robot programs to spot-weld the underbody
6. Get a confirmation from the robots that the work is finished
7. Lift up the underbody
8. Check if the next station is empty
9. Start the conveyor to send it to the next station

2.2.2 Station 15-38-110

In station 15-38-110 the A-pillars of the car is assembled onto the underbody. The station consists of two spot-welding robots, two robots with grippers, one conveyor, one decoder, two turn-tables, one lift and one fixture. The fixture is placed above the underbody (see Figure 1 and Figure 2) and helps the turntable to fixate the A-pillars. As for station 15-38-100 a decoder checks which model the underbody belongs to.

The two robots with grippers are starting their operation cycle when an underbody arrives to the previous station (station 15-38-100). This is needed to decrease the cycle time for this station. The robots collect one part each from the racks (see Figure 1) and travels to the gluing gun on special designed conveyors (see Figure 3). The gluing operation is controlled by the robot, but the PLC is starting and stopping the heater of the gluing gun when for example a shift ends or a longer stop is needed. Parts are always available in the racks due to a two level system. When the top level is empty a gate is raised and the robot can start collecting part from the lower level. Forklift drivers load the upper level with new parts and give a confirmation to the PLC that the upper level is refilled with the help of a barcode scanner. Later when the lower level is empty the gate will go down and the robot will start collecting parts from the upper level again.

The spot-welding robots perform welding operations to fasten the A-pillars. The parts are both glued and welded onto the underbody.

Below is a brief description of the PLC operations in station 15-38-110:

1. Receive information of which car model that will arrive next from the previous station (station 15-38-100)
2. Start the gripper robots operations
3. Start the conveyor to bring in a new underbody to the station
4. Lower the underbody down to fixate it for the robot welding
5. Get a confirmation that the gripper robots are finished and that the metal parts are placed in on the turntable.
6. Turn the turn table and press the metal parts onto the underbody
7. Close the clamps of the fixture above the underbody to hold the parts
8. Start the robot programs to spot-weld the underbody
9. Get a confirmation from the robots that the work is finished
10. Release the clamps of the above fixture and turntable
11. Turn the turn table to its home position
12. Lift up the underbody
13. Check if the next station is empty
14. Start the conveyor to send it to the next station

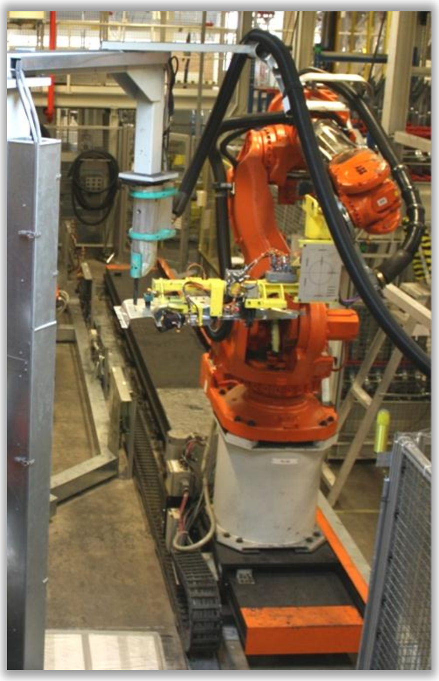


Figure 3 – ABB Robot applying glue on a metal part in station 15-38-110

2.2.3 Station 15-38-120

The tasks performed in s are enhanced spot-welding as in station 15-38-100 (see Figure 4). The station consists of four spot-welding robots, one conveyor and one lift. When the underbody is finished at this station it is stored into a buffer before it heads to the next station. However this buffer is not a part of this master thesis and will not be considered.

Below is a brief description of the PLC operations in station 15-38-120:

1. Start the conveyor to bring in a new underbody to the station
2. Lower the underbody down to fixate it for the robot welding
3. Start the robot programs to spot-weld the underbody
4. Get a confirmation from the robots that the work is finished
5. Lift up the underbody
6. Check if the buffer is ready to receive
7. Start the conveyor to send it to the buffer

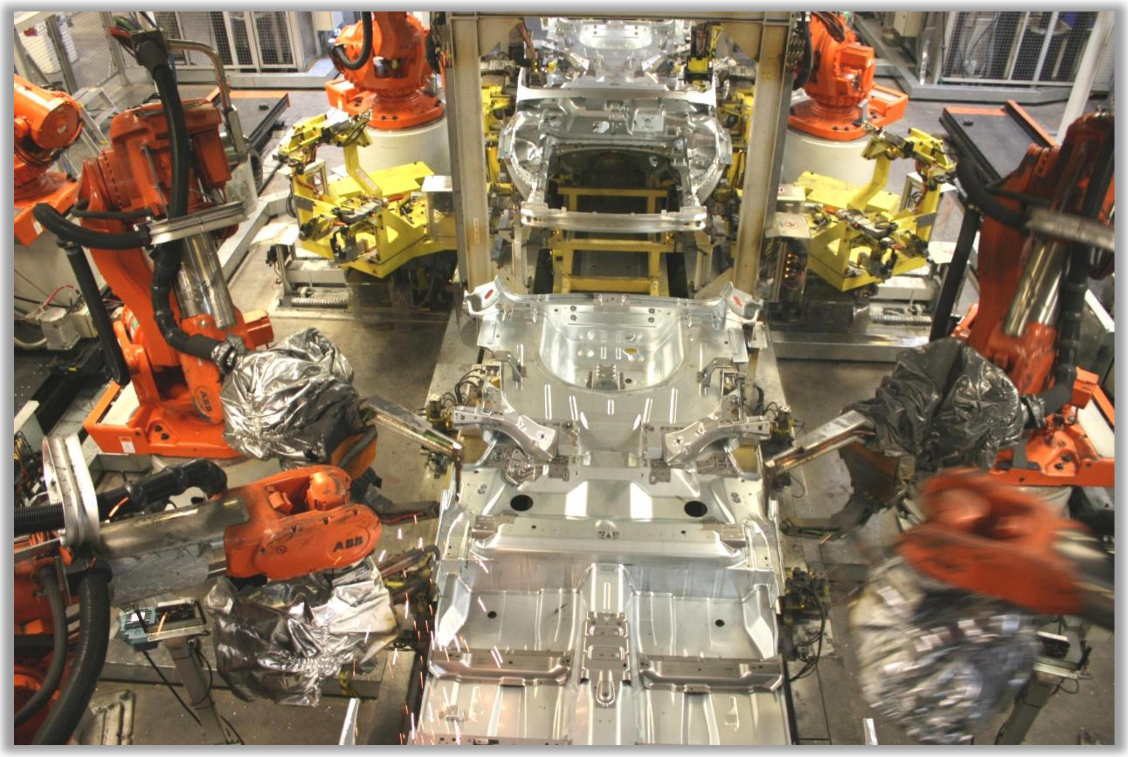


Figure 4 - Spot-welding in station 15-38-120

2.3 Tree structure for station 15-38-100, 110, 120

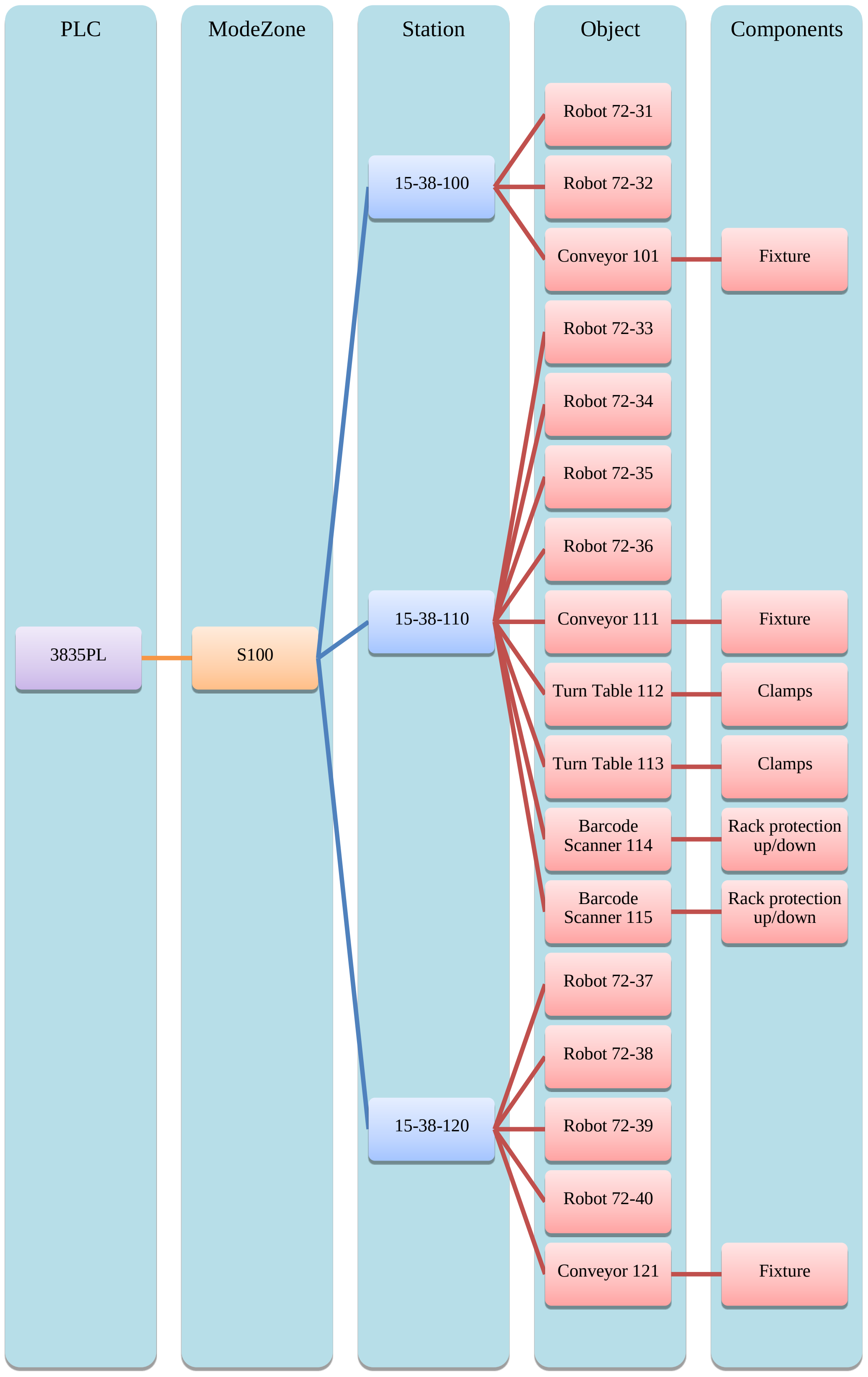


Figure 5 - Illustration of the tree structure for station 15-38-100, 110, 120

3 Frame of Reference

This chapter will describe a theoretical background of the different concepts, definitions and software's used in this master thesis. Since the main topic of the report, automatic generation of PLC programs using Automation Designer, is a fairly new concept there have been some difficulties to find publications and literature dealing with this topic solely.

3.1 Programmable Logic Controller (PLC) for Industrial Automation

A PLC (Programmable Logic Controller) is a digital computer used for controlling an automated production system such as factory assembly lines. The PLC controls equipment like robots, fixtures and conveyors and the PLC programs are traditionally done late in the commissioning process.

3.1.1 IEC 61131

The main standard for PLC is the IEC 61131 developed by the International Electro-technical Commission (IEC). The standard is divided into the following five parts (PLCopen, 2008):

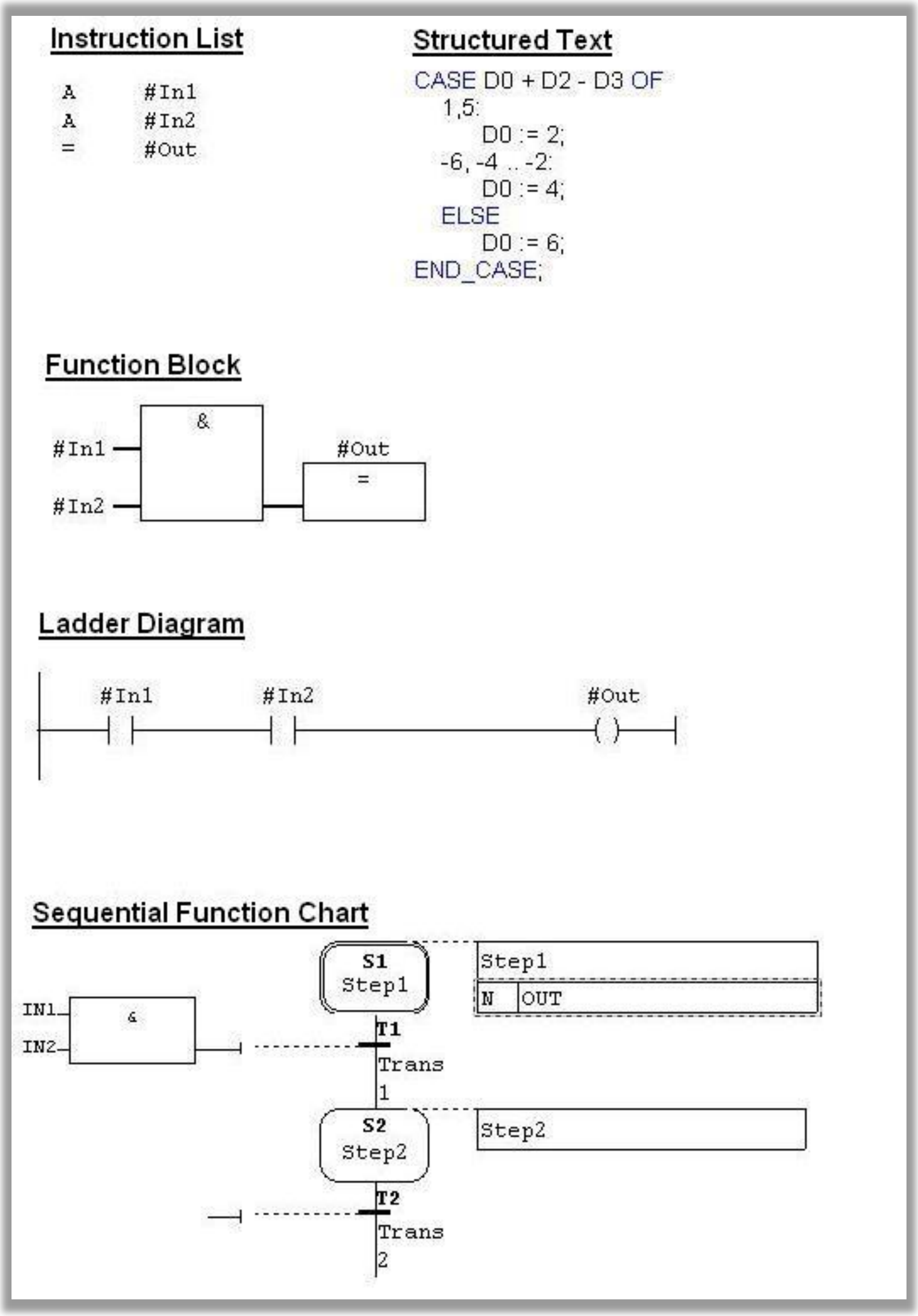
1. General information
2. Equipment requirements and tests
3. Programming languages
4. User guidelines
5. Messaging service specification

The programming instructions at VCC are partially based on the IEC 61131-3 standard (Volvo Car Corporation, 2008) that handles the allowed programming languages and supplies guidelines on how to get the most out of each language (Ashton, 2007). The supported programming languages in IEC 61131-3 are:

- Textual
    - Instruction list (IL)
    - Structured text (ST)
- Graphical
    - Function block diagram (FBD)
    - Ladder diagram (LD)
- Sequential function chart (SFC)

SFC is used as a way to structure the internal organization of the program (see Figure 6) while the textual language with instruction list and structured text are inter-operable languages. The graphical language with function block diagram and ladder diagram are also inter-operable languages but they differ a lot from the textual when it comes to programming. Some samples of the different programming languages are shown in Figure 6.

All programming languages have their advantages but for reuse from one program to the next the graphical languages is preferred. Function block diagrams are very easy to use when storing certain programming functions, e.g. robot communication. With this language it is possible to build up a database with different block or diagrams that the programmer can use for a lot of different projects and save time by reusing already created code for different functions. This is also possible to do for the textual languages but it is much easier to connect different graphical languages (Selander, 2010).



Instruction List

```
A       #In1
A       #In2
=       #Out
```

Structured Text

```
CASE D0 + D2 - D3 OF
   1,5:
        D0 := 2;
  -6, -4 .. -2:
        D0 := 4;
   ELSE
        D0 := 6;
END_CASE;
```

Figure 6 - Programming structure for the different PLC languages

3.1.2 Function Block Diagrams (FBD)

The function block diagram (FBD) language is one of the PLC programming languages stated in the IEC 61131-3 standard (PLCopen, 2008).

A standard FBD is built up by input/output variables, through variables, internal variables and the behavior of the FBD (Torsten Heverhagen). An illustration of a standard FBD is shown in Figure 7.

- **Input variables**
  The input variables are all external signals that are necessary for the FBD to be able to perform its specified function. Input variables can only be set from outside the FBD.
- **Output variables**
  Output variables are the signals that are generated in the FBD through the internal logical behavior. The output variables are often the desired result from the FBD and they are the only variables in the FBD that can be read from the outside.
- **FBD behavior**
  The internal logical behavior is determined by the FBDs specified function. The internal logic can be programmed in one or a combination of the existing inter-operable PLC programming languages.
- **Internal variables**
  Internal variables are only used inside the FBD and cannot be accessed from the outside.
- **Through variables**
  Through variables are used to when several FBDs in a row uses the same input variable. The value or data type of a through variable never changes when passing through a FBD

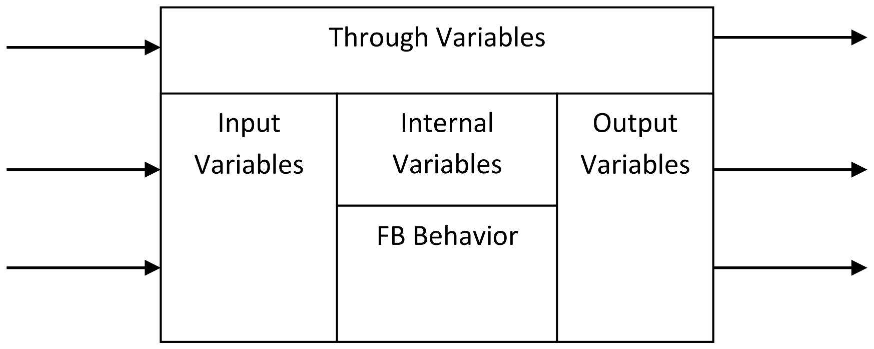


Figure 7 - General description of a function block

3.1.3 Different ways of structure the PLC code

When programming a PLC with function blocks there are two ways to structure the program, flat structure and hierarchal structure. A flat structure is often easier to program but for larger projects it tends to get very complex and hard to overview, therefore it is also harder to maintain. The hierarchal structure works in an object oriented way and requires more planning but it allows for a wider use of library blocks. A library block is a function block with functionality reused frequently and therefore stored in a library. In a larger company with a lot of similar processes the use of library blocks will in the long term save a lot of time and simplify the maintenance work. A hierarchal structure is good when organizing the PLC program and its code for objects such as robots or conveyors. It will enable the possibility to group together objects in a desired way. Figure 8 represents the difference between a flat and a hierarchal structure.

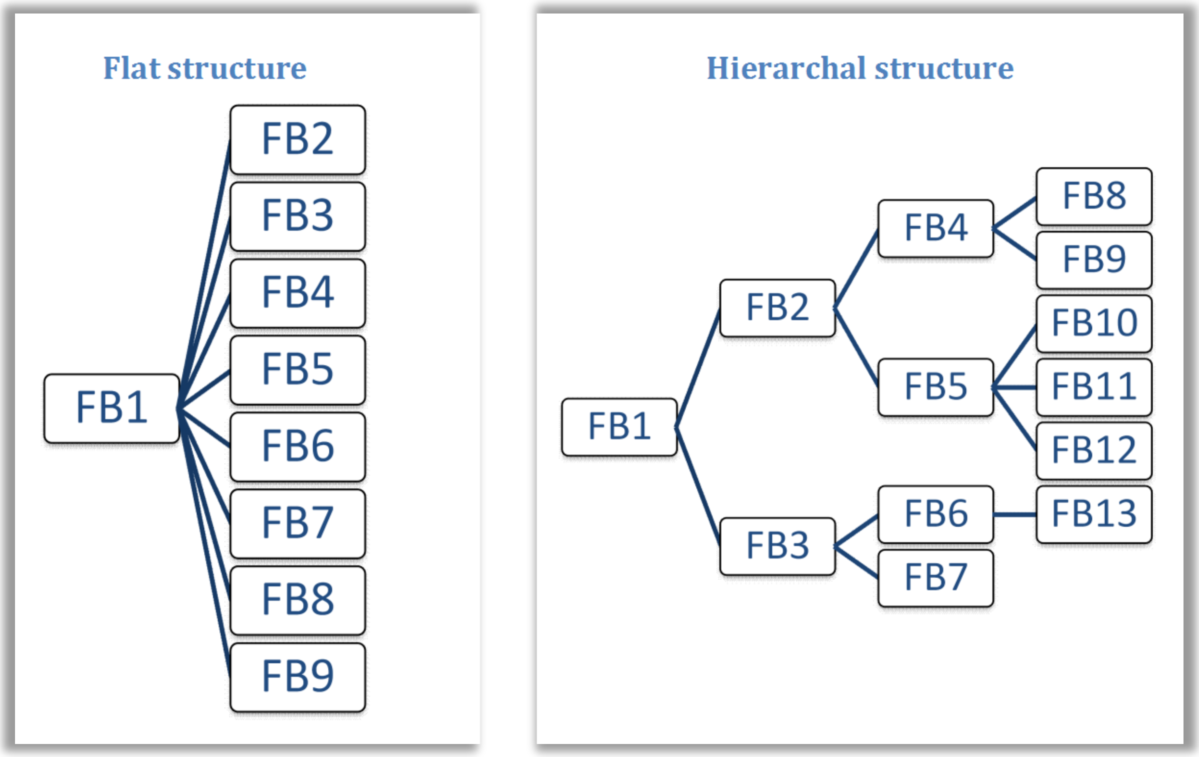


Figure 8 - Example of flat and hierarchal structures

3.1.4 SIMATIC STEP 7

STEP 7 is a software package for programming SIMATIC PLC´s from Siemens. The standard package includes symbol editor, SIMATIC Manager, NETPRO Communication Configuration, Hardware Configuration and Hardware Diagnostics. STEP 7 supports the programming languages Ladder Diagram, Function Block Diagram and Statement List (Siemens AG Automation and Drives, 2006). However all the programming standards in IEC 61131-1 can be implemented in the program with expansion packages.

3.1.5 Methods for verification and validation of PLC programs

To verify a PLC program is to check that the program is created in the right way e.g. according to a standard (Frey & Litz, 2000). The methods to do this are to investigate the code yourself and also to have someone else to look at it. However, this person needs to have a wide knowledge about the standards followed in the project (Selander, 2010). Another step in the verification process is to compile the PLC code to make sure that there are no syntax errors.

The validation is to investigate that the PLC program works and behaves according to the specification (Frey & Litz, 2000). The first step is to test the program in a virtual PLC. This is done by running the code virtually in the PLC software and manipulate the input signals. Then study the behavior of the program and the output signals that will appear. After this step the only way to make sure that the PLC program works correctly is to test it on the real production line to find the remaining errors and correct them. This trial and error process have to be completed before the production can start. This process needs to be as optimized as possible because of the costs of having a production line halted for testing is very high (Selander, 2010).

3.1.6 Requirements for automatic generation of PLC code

When generating PLC code automatically it is important that the generated code not only works in ideal conditions when the process or production plant is up and running. The generated code shall also be able to handle parts such as interlock logic, safety instructions, start-up and shutdown sequences (Güttel, Weber, & Fay, 2008). Otherwise a lot of manual work will still be needed afterwards.

In a scientific study made by Güttel et al. four requirements have been established to be able to auto generate PLC code. The first requirement is that the plant structure must be available in a computer readable form. This is necessary because to be able to generate e.g. interlock logic for a conveyor the connection between its sensors needs to be defined. The second requirement is that a process description needs to be available. The description also needs to be in a machine readable form where Güttel et al. suggest using the IEC 61131-3 language SFC and "PLCopen-XML" as the representation of the format. The third requirement is that a well-defined and standardized function block diagram library needs to be available. This is very important because it is within these function block diagrams the functionality is represented of the components in the process or production plant. The last requirement is that the knowledge of today's process engineers needs to be extracted and put down into computer readable rules. This must be done to ensure the quality of the generated PLC code.

3.2 Human Machine Interface (HMI) for Industrial Automation

HMI in this paper is the interaction between the PLC system and the operator. The interaction is presented by a screen with dynamic icons, figures and text. The HMI screen is presented on a panel or a standard PC positioned in the production area. An operator can monitor the production and control it to a certain level by the help of an HMI panel or PC. To be able to be an interface between the PLC and the operator the HMI system has to be able to do the following tasks (Siemens AG Industry Sector, 2008):

- Visualize the process
- Control the process with help of an operator
- Display upcoming alarms
- Archive alarms and process values
- Log alarms and process values
- Manage process and machine parameters

3.2.1 SIMATIC WinCC flexible

WinCC flexible is a Windows based engineering software created by Siemens for producing HMI screens. The screens are available for use on SIMATIC HMI operator control and monitoring devices as well as standard PC´s (Siemens AG Industry Sector, 2010). WinCC flexible is available in four different versions which are Micro, Compact, Standard and Advanced. The difference between the versions is the number of SIMATIC panels that the software is applicable with. Micro has the least number of compatible SIMATIC panels while Advanced supports all SIMATIC panels and is the only version that is PC compatible (Siemens AG Industry Sector, 2010).

SIMATIC WinCC flexible is closely integrated with the PLC software SIMATIC STEP 7. It is possible to manage WinCC flexible projects within STEP 7 and communication settings, tags and alarms can be shared between the software’s (Siemens AG Industry Sector, 2010). The integration is made by the program SIMATIC Manager, a component of STEP 7. The coupling in the production between the HMI screens and PLC is made via PPI, MPI, PROFIBUS DP and PROFINET (TCP/IP) (Siemens AG Industry Sector, 2010).

3.3 Simulation Studies

This chapter will cover the work methodology at VCC when it comes to simulation studies. The chapter will also include the type of software used in the simulation studies.

3.3.1 Technomatix Process Simulate/ Process Designer

Process Simulate (PS) and Process Designer (PD) are two product lifecycle management (PLM) software tools from Siemens. The software’s makes it possible to plan the manufacturing process virtually in a 3D environment before it is implemented in the factory. With PS and PD it is possible to reuse information from earlier manufacturing process planning’s and this will decrease the time of work when a factory needs to be redesigned or complemented with new equipment (Sundbäck, 2010).

Process Designer is used to virtually design the production layout. This includes importing CAD (Computer Aided Design) models of the factory and its components and machines into the software. Then to design the line, model capabilities of the process and balance the line (Siemens PLM Software, 2008).

Process Simulate is the next step when performing a virtual simulation study. The software is a tool for verification, optimization and commissioning of the designed factory created in PD (Siemens PLM Software, 2008). PS consists of five segments designed to verify and optimize in a certain area of the factory. The different segments are Assembly, Human, Spot Weld, Robotics and Commissioning (Siemens PLM Software, 2008). The software simulates the logical behavior of the production and sequencing of the operations (Siemens PLM Software, 2008).

3.3.2 Simulation Studies at VCC

VCC uses Process Simulate and Process designer in the first steps of the commissioning phase. The software’s are mainly used to verify if a new car model is possible to produce in the factory. When the engineers start working with their simulation projects the car is not yet completely developed. The specifications available at the project start up for the virtual simulation studies usually changes and the models needs to be modified later on in the commissioning process (Sundbäck, 2010).

The engineers work is to investigate how much equipment that is possible to reuse and how much new equipment that is needed to produce the new car model. Studies on the possibility to modify existing lines and if enough space is available to place the new equipment needed also has to be performed. If there is a lack of space a new line needs to be built inside the factory. But sometimes that is not an option for the company and this result in redesign of the parts that could not be produced or assembled on the modified current production line (Sundbäck, 2010)

Another assignment for the simulation engineers is to check the reachability for the robots. Robot programs are also produced to simulate the production lines to get a clear view whether it is possible or not to produce the new car at the production line designed by the engineers. When the simulations are validated and the solutions are found, the external line builders take over and start developing the real production line that will be installed in the factory (Sundbäck, 2010).

4 PLC programming at VCC

Most of the PLC programming at VCC is done by contracted line builders. However when the PLC code has been delivered to VCC, all maintenance and minor changes to the code is performed in-house. To ease this work VCC has developed a standard for the structure of the PLC programs that all their suppliers must follow.

4.1 VCC PLC programming philosophy

VCC PLC programs are built up by function block diagrams in a tree structure (see Table 1 and Table 2). Each function block diagram represents either a real object, such as a conveyor or a robot, or a function such as a mode zone or a bit to word converter (Volvo Car Corporation, 2008).VCC has built up a library of function block diagrams that are designed to be as general as possible and to fit in all possible station configurations. VCC provides this library to their line builders who are required to use it as far as possible. The function block diagrams are used in conjunction with the tree structure (see Table 1) so that each function block diagram only speaks to one level above and below in the tree. A description of each level can be found in Table 2. If this standard has been followed properly the engineers at VCC have it much easier navigating the code during maintenance and upgrades (Selander, 2010).

Table 1 - Description of the PLC tree structure at VCC

Level	1	2	3	4	5	6	7	8
	PLC							
		EStop Zone						
			Mode Zone					
				Objects				
					Objects			
						Components/Functions		
							Components/Functions	

Table 2 - Description of the different levels in the VCC PLC tree structure from Table 1

Level	Description
PLC	The PLC level is the topmost level that is represented within the PLC code. It deals with the basic functions of the PLC and handles the relationship between the different mode zones.
EStop Zone	An emergency stop zone covers all objects that need to be stopped if the emergency stop is triggered. One PLC can control several EStop Zones.
Mode Zone	A mode zone contains all objects that have to be run in the same mode simultaneously e.g. automatic or manual mode. It is within the function block diagram for the mode zone the majority of the interlock calculations for the objects is made.
Object	The objects are the machines that are used to build and transport the car and its parts. VCC has a library with function block diagrams that covers most of the objects used. But for some of the more complex or unique objects the FBDs have to be manually created.
Components	The components are physical parts of the objects that have a specific function and are used in multiple instances. For example a conveyors motor. VCC have several different types of conveyors that use the same type of motor.

Functions	Function block diagrams used to execute a specific operation commonly used by several objects. An example of this is the "transfer in" operation that is used by the conveyors.

4.1.1 Handling alarms at VCC

To make sure workers and equipment are safe and that troubleshooting goes as fast as possible VCC has defined four types of alarms for their PLC programs:

- Level A: Safety function tripped. All machines must immediately stop.
- Level B: All machines in automatic mode are stopped. Manual mode is still allowed.
- Level C: Error that requires intervention from an operator. E.g. machine has run out of assembly parts.
- Level D: Error that does not requires intervention from an operator. E.g. machine is waiting for an interlock requirement to be fulfilled.

If a level A or B alarm is generated, an operator needs to manually reset the alarm after the cause has been fixed. Otherwise the alarm won't go away. Each function block diagram shall be able to generate its own alarms. These alarms are summarized in each level and sent upwards in the PLC structure. A function block diagram can stop itself and the blocks below directly. However, it can only report errors upwards in the structure and the function block diagram on a higher level will have to decide if any other consequences are needed.

4.1.2 Siemens STEP 7 programming standard at VCC

VCC has decided to only use hardware and software from Siemens when building new installations and has therefore specified a detailed Siemens PLC programming standard that is based on their general standard (Selander, 2010).

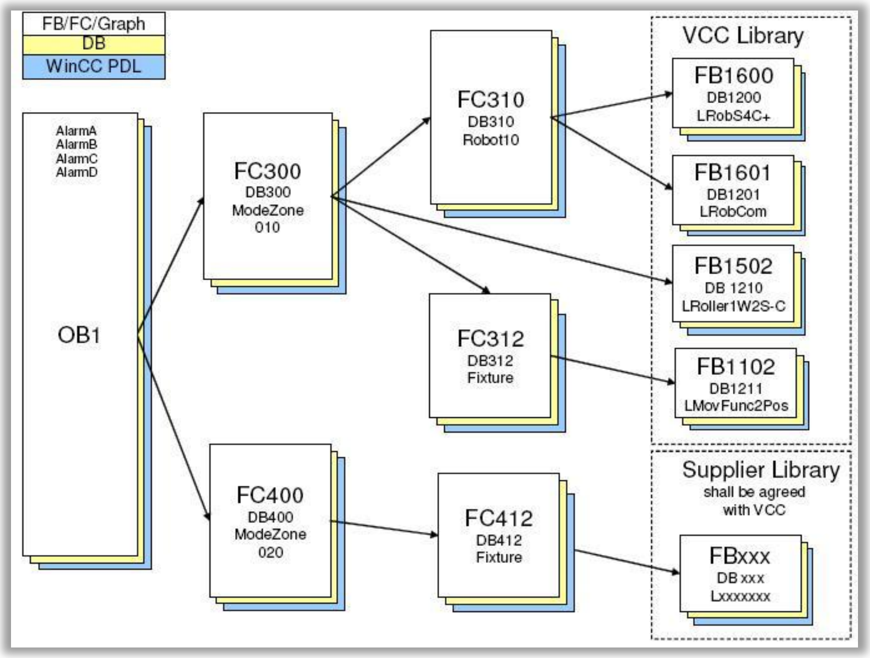


Figure 9 - Structure of Function Block Diagrams for VCC in STEP 7

The programming in STEP 7 shall be done with function block diagrams in according with the general standard. In STEP 7 there are two types of function block diagrams called FB and FC. The main differences between them are that FB must have a specified data area called data block (DB) connected to it. A FB can also store static variables between each scan cycle. At VCC all FC created by line builders must also have a DB (Volvo Car Corporation, 2008). This is to make it easier finding the variables connected to each FC. VCC uses FB for functions that are reusable and these blocks are stored in a library. The FCs are used for unique project specific functions (see Figure 9). It is required in STEP 7 that each FB, FC and DB is assigned a unique number that works as an identifier. VCC has set in their standard a convention for how numbering shall be made in their projects. The most important numbering rules can be found in Table 3.

Table 3 - Standard for FB, FC and DB numbering

Type	Adress	Description
FB/FC		
Local functions – General	251-299	Project specific block for general functions.
Local functions - Mode Zone	300-549	Project specific blocks for blocks within a mode zone.
S7 Graph	550-599	SFC blocks
Local library	600-999	Library blocks that are unique to Torslanda or Ghent.
Standard library	1000-2048	VCC library blocks.
DB		
Local functions – General	251-299	Project specific DB. Always the same number as its connected FC.
Local functions - Mode Zone	300-549	Project specific DB. Always the same number as its connected FC.
S7 Graph	550-599	DB for SFC blocks
General	1000-1199	Instance DB for library blocks that is not part of a mode zone
Mode Zone 1	1200-1399	Instance DB for library blocks that is a part of mode zone 1
Mode Zone 2	1400-1599	Instance DB for library blocks that is a part of mode zone 2
Mode Zone 3	1600-1799	Instance DB for library blocks that is a part of mode zone 3

FBs and FCs are built up of networks containing PLC code. Networks are used to structurize the code. A network could for example contain the call of another FB or the calculation of an interlock variable. In the VCC standard (Volvo Car Corporation, 2008) there are rules for in which order the networks shall be placed and what they shall contain (see Table 4).

Table 4 - Description of VCC network structure in STEP 7

Network	Description
1. Inputs	Functions whose main purpose is to read and move inputs.
2. Alarms	Functions dealing with alarm creation.
2.1. Enable Alarms	Disabling alarms if a higher priority alarm is active.
2.2. Alarm generation	Functions dealing with alarm creation.
3. Communication	Functions whose main purpose is communication.
3.1. Other systems	Functions for communication with other systems e.g. other PLC:s or the manufacturing execution system.

3.2. Devices	Functions for displays and BUS nodes
4. Machine status	Functions whose main purpose is machine status.
4.1. Detected	Functions whose main purpose is to deal with detectable machine status from inputs.
4.2. Parameter	Functions whose main purpose is to deal with machine status derived from memorized variables.
4.3. Operator selection	Functions whose main purpose is to deal with machine status from operator selections
4.4. Calculated	Functions whose main purpose is to deal with machine status calculated using other statuses as inputs.
4.5. Modes	Functions dealing with the different production modes.
4.6. Indications	Functions whose main purpose is to indicate the machine status to the operator.
5. Equations	Equations that does not fit in any of the other areas.
5.1. Interlocks	Equations dealing with interlocks.
5.2. Flow equations	Equations dealing with the process flow.
6. Sequences	All sequences
7. Submodules	Sub levels that is not part of any other dedicated areas.
7.1. Non machine	Sub levels who doesn't deal with machines.
7.2. Command summary	Function to sum-up all signals to steer each output
7.3. Machine	Sub levels dealing with machines e.g. robots or rollers.
8. Outputs	Functions dealing with the outputs from the function block
9. Alarm summary	Summary of all A, B, C and D alarms.

4.1.3 HMI programming standard at VCC

For creating the HMI screens Siemens WinCC is used. Since WinCC and STEP 7 are highly integrated the screens shall be created in the same project as the PLC code. For each FC and FB in the project there shall be a corresponding screen (see Figure 9) as well as for the OB1. VCC provides finished screens for the FB from their library and a starter pack containing templates to be used for the project unique screens. For each FB and FC there is also an icon that is to be used in the screen one level higher in the tree structure to display the status of the object (Volvo Car Corporation, 2008). Each FB and FC has two variables of the type word that are used only by the HMI to read the necessary information to the screens and icons respectively. The words are interpreted in WinCC through visual basic scripts. The result is a communication between the function block diagrams (FCs and FBs) and the HMI screen where the screen shows the information such as alarms to the operator. WinCC flexible is used in the same way by VCC, but all screens and all other functions are not yet finalized.

4.2 PLC programming/HMI in the commissioning process at VCC

The development of the PLC programs and HMI screens are always outsourced by VCC to external line builders. The line builders are also responsible for other things like mechanical and electrical construction of the production line. The line builders have close contact with the engineers at VCC and need to follow the standards that are available. In some special cases VCC have been forced to do some PLC programs and HMI themselves (Selander, 2010).

The first thing a line builder needs to do before he can start programming the PLC code is to think through and present the structure of the program and which function blocks that will be used from VCC’s programming library. The structure needs to be approved at VCC before the new PLC code can be developed. In the end the developed PLC code and HMI also needs to be presented to VCC and be approved before it can be implemented into the production line (Selander, 2010).

5 Automation Designer

Automation Designer (AD) is a software from Siemens that is developed on the base of Comos. The software is an engineering platform, mainly for the manufacturing industry, to help decrease the time for engineering and start commissioning earlier. Automation Designers main purpose is to help the engineers take the step from the digital factory to the real plant and to reuse as much information as possible at all times (Siemens AG Industry Sector, 2010). Because of the lack of information about Automation Designer, this chapter has been reviewed by Michael Geck and Steffen Weber at Siemens AG Industry Automation Division in Nürnberg, Germany.

5.1 What is possible to do in the software?

Automation Designer is developed to decrease the ramp up time in a factory when new equipment needs to be installed or a new line is going to be developed. This is done through reuse of information and a possibility to generate the information needed to implement the equipment or the line. The software is intended to be the last step before implementation of the prepared work in the physical factory.

Automation Designer has got three main features (Siemens AG Industry Sector, 2010):

- Functional, Layout based Engineering
- Generators for PLC Code, EPLAN electric P8 and WinCC flexible
- Data Connection to earlier engineering phases like mechanical simulation

All three features are connected to each other in the software but it is possible to work with only one or two of them. It is important to know that the user must have a valid license of the software´s in the list above to be able to use the generated information. STEP 7, EPLAN and WinCC flexible is not part of Automation Designer; however it is possible to work with the different features inside AD without having the software´s installed.

Automation Designer is also a powerful tool when it comes to documentation. A lot of different documents such as list of equipment in the factory and other are possible to generate with the push of a button if you have the right information inside the software. In the following chapters the different features of Automation Designer is described further.

5.1.1 PLC code generation

Automation Designer (AD) is able to generate PLC code to the Siemens software SIMATIC STEP 7. Along with the PLC code AD is also capable to generate the hardware and symbol table for the project. The symbol table is generated automatically within AD and then exported along with the generated PLC program. However the hardware must be selected by the help of a hardware wizard. The hardware wizard inside Automation Designer has got the same functionality as the one in STEP 7.

To be able to generate PLC code, the software needs to know which code to generate. This is done by adding function block diagrams to different templates (more information about templates in chapter 5.2.5). The function blocks can either include the PLC code it should have or be empty. In the case where it is empty, AD can collect its code from a function block library during the generation of the PLC program. The idea with the code generation is that function block diagrams are connected to templates and Automation Designer connects these blocks together into a PLC program. Afterwards the generated program is editable in STEP 7. More information about what is needed to make this work is described in chapter 6 Methodology.

5.1.2 Generation of HMI pictures

Automation Designer is using the XML format for importing and exporting data to and from WinCC flexible. The software is able to generate complete HMI projects with up to four different levels of details. Tags and other information can be connected inside Automation Designer to the right signals from the PLC structure before the user exports the HMI pictures. This anticipates that Automation Designer includes PLC code for the desired project.

To be able to generate HMI pictures the software needs to know which icons and other information to generate in every level of the HMI. This is solved through import of XML files to the Templates in the software. These XML files should include the icons and other information that represents the template object in the different levels of the generated HMI project. An example of this is shown in Figure 10.

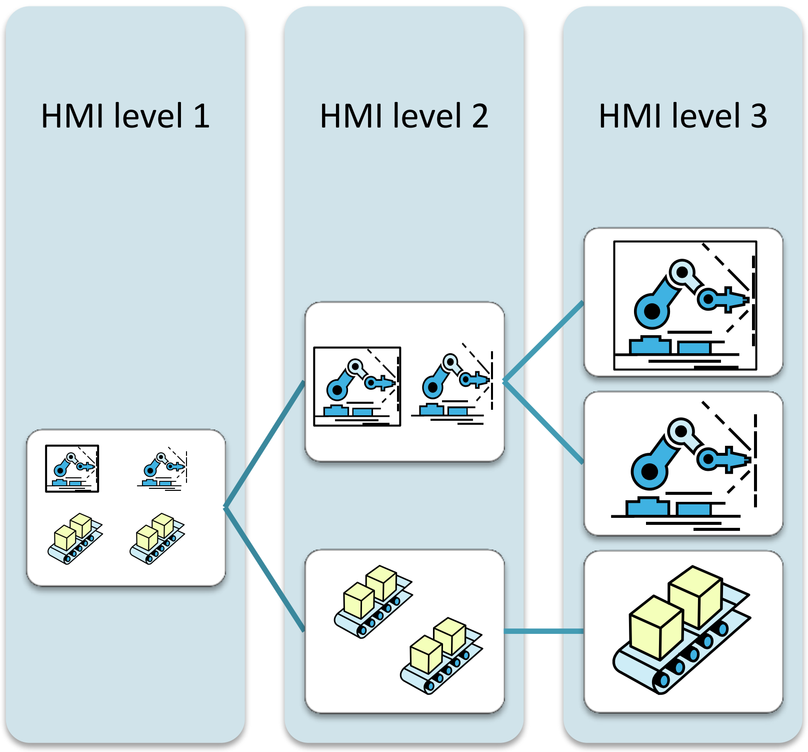


Figure 10 - Icons for different HMI levels

Another thing that is necessary before generating an HMI project is that Automation Designer needs to know where to place the different objects relative to each other on the HMI screens. This is solved through a 2D-layout inside AD (more information of this in 5.2.7). The user places his objects in the right place on the layout and draws polygon lines around the objects. This is done to explain to the software which objects that should be in the different levels.

The HMI project is generated to one single XML file that can be imported to WinCC flexible and afterwards be edited if necessary. The idea with the HMI generation is that XML files are connected once to a template object. Then every time this object is used in a project the software will use the same HMI information for generating HMI pictures.

5.1.3 Generation of EPLAN drawings

Beside the integrated E-CAD functionality Automation Designer is currently supporting EPLAN electric P8. To generate electrical drawings to EPLAN is done in a similar way as the HMI. The idea is that single electrical documents are connected to templates inside AD. By then placing the templates on the 2D-layout, the software is able to connect the different equipment represented by the templates and create electrical drawings which can be exported to EPLAN and later be modified if necessary.

The connection of the electrical drawings to template objects is only supposed to be done once and then be reused in every new project inside Automation Designer.

5.1.4 Reuse information from earlier simulation studies

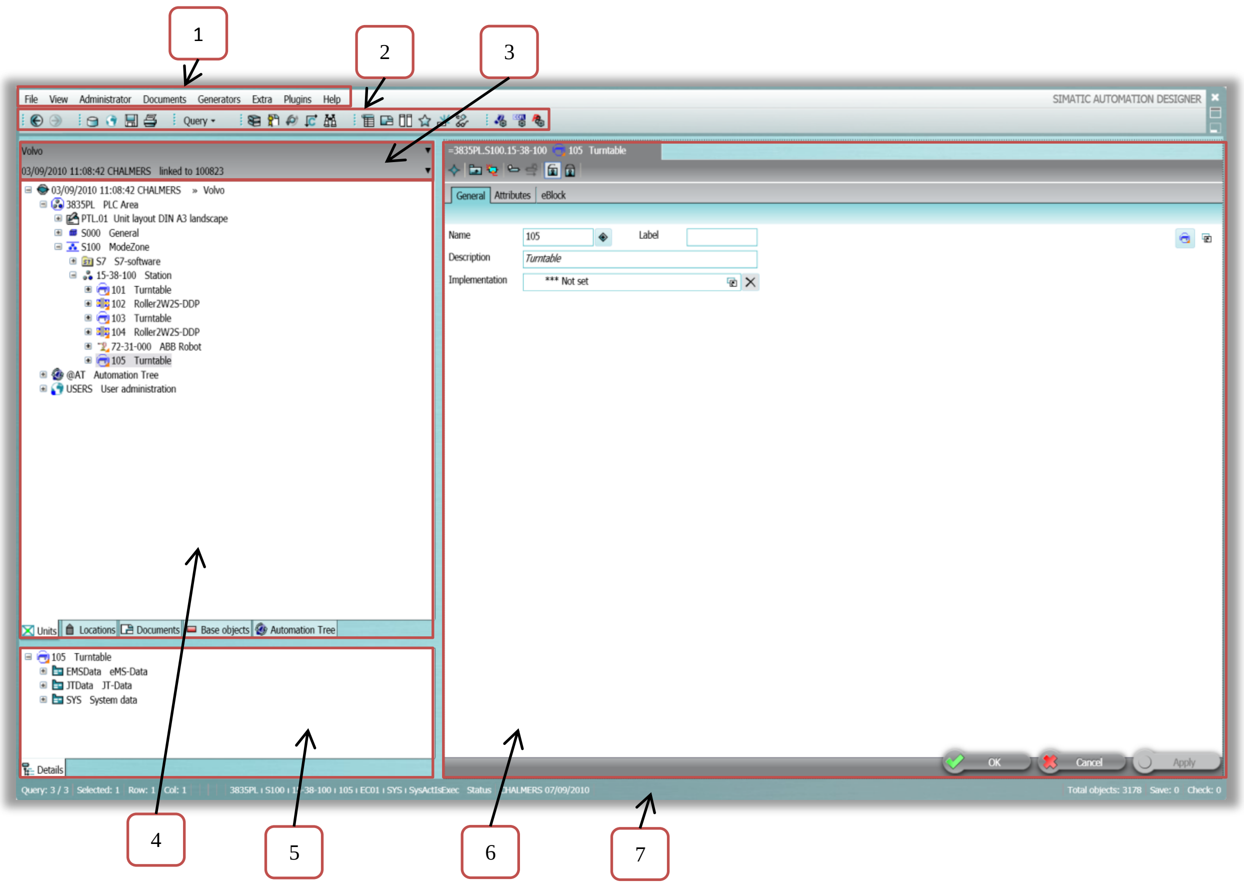
One of the main purposes with Automation Designer is to help the engineer to reuse information from earlier simulation studies. This is possible if the simulation study is made in Process Simulate or Process Designer. Then the simulation study can be exported in an XML file and be imported into Automation Designer. By doing this the engineer will get the structure and the objects needed in his engineering project (more information of different types of projects in chapter 5.2). The key factors to make this work are that a database with template objects is already developed where every object in the simulation study is represented and also that the names from the simulation study are connected to the right templates.

Another thing that the engineer can gain from importing an XML file from Process Simulate or Process Designer is to get the sequences from the simulation. The sequences can be stored and used inside Automation Designer with a special feature called Sequence Designer. With this tool the engineer is able to complete the sequences and include information such as interlocks that may be left out from the simulation study. By doing this Automation Designer is able to generate SFC in addition to the function blocks to complete the final PLC code for export to STEP 7.

5.2 Engineering with Automation Designer

When introducing Automation Designer into companies there are two things that are very important to keep in mind. First is to come up with a structure for the engineering projects that fits well with the company's perception of their factories. All objects created later on in the project will have to fit into this structure so a well-defined structure will help a lot. The second thing is to specify all relevant naming conventions that are in use in the company and if possible create new rules where there are none. An Automation Designer project relies heavily on scripting and with naming conventions the complexity of these scripts can be reduced.

5.2.1 User interface and layout of Automation Designer


1
2
3
File View Administrator Documents Generators Extra Plugins Help
Query
SIMATIC AUTOMATION DESIGNER
Units Locations Documents Base objects Automation Tree
Details
General Attributes eBlock
OK Cancel Apply
4
5
6
7

Volvo
03/09/2010 11:08:42 CHALMERS linked to 100823

03/09/2010 11:08:42 CHALMERS » Volvo
3835PL PLC Area
PTL.01 Unit layout DIN A3 landscape
S000 General
S100 ModeZone
S7 S7-software
15-38-100 Station
101 Turntable
102 Roller2W2S-DDP
103 Turntable
104 Roller2W2S-DDP
72-31-000 ABB Robot
105 Turntable
@AT Automation Tree
USERS User administration

105 Turntable
EMSData eMS-Data
JTData JT-Data
SYS System data

=3835PL.S100.15-38-100 105 Turntable

Name 105
Label
Description Turntable
Implementation *** Not set

Query: 3 / 3 Selected: 1 Row: 1 Col: 1 3835PL ı S100 ı 15-38-100 ı 105 ı EC01 ı SYS ı SysActIsExec Status CHALMERS 07/09/2010 Total objects: 3178 Save: 0 Check: 0

Figure 11 - User interface and layout of Automation Designer

1. **Menu bar**
2. **Symbol bar –** Contains shortcuts to some of the most used functions in the menu bar.
3. **Recent projects –** Quick access to the most recent opened projects and layers.
4. **Navigation tree –** Enables the user to navigate between objects in the hierarchal tree structure of the project.
5. **Detail view –** Shows the selected objects attributes without having to open it in the configuration area.
6. **Configuration area -** Displays the current opened object.
7. **Status area –** The status of the currently opened project

5.2.2 Base project

In the base project all the base objects and templates are created and stored. The base project is the database of Automation Designer and works as a foundation for the engineers. Only a selected few people with great knowledge of both Automation Designer and the work methods the company uses for their automated systems should be allowed to make modifications and add new things into the base project. To build up a base project requires a lot of time and resources in the beginning but if prepared well it will save time in the long run.

It is in the base project the structure of the engineering project is created. This is done partially in the templates but also by assigning which objects should appear and be addable when right clicking on a specific object.

To avoid destroying working objects when doing changes in the base project, Siemens has added a layer functionality. The layers works as a copy of the original database where changes can be made and checked before releasing and implementing it to the original database. There are tools included so the user can see which objects have been changed before releasing a layer.

5.2.3 Engineering project

While the work in the base project is to create general objects and templates it is in the engineering project that they are specified and fitted together to create a representation of a real part of the factory. It is here most of the work will be done after the base project has been built up. If the templates and base objects have been done well the users in the engineering project only needs to know what parts that needs to be added and they won't need deep knowledge of the involved technologies such as PLC-code, HMIs and Electrical drawings.

As in the base project it is possible to use layers in the engineering project to avoid destroying functional parts of an existing project.

5.2.4 Base objects

Base objects are the building blocks of an Automation Designer project. The base objects can be anything from a representation of a physical object to a function for linking together function blocks. Siemens provides a large library with base objects that can handle most of the problems the engineers will encounter but if need there is the possibility to create your own or copy and modify one from the library.

A base object is built up by scripts and attributes. The attributes can be grouped together in tabs to get the related information stored in one place. This information can be used or generated by scripts to get the desired function of the object. Automation Designer uses the VB-script language for its internal macros. An attribute can be locked in the base project so a user in the engineering project can't change it. It is also possible to hide attributes which makes them visible only in the base project. Another thing which is possible is to set a limitation of the maximum characters for the names of the base objects. This could solve the problem of having to long names in the PLC code. The limitation would stop an engineer working in an engineering project to set a name that would be too long and cause problems in the STEP 7 export.

5.2.5 Templates

A template is used to group together base objects to create more complex objects. For example a template for a conveyor can include sensors, motors, PLC-code, electrical drawings among other things. In each template there is the possibility to add unique values to the base objects attributes so the same base objects can be used in several templates but with different purposes. New scripts can also be added that will only be applied in the current template.

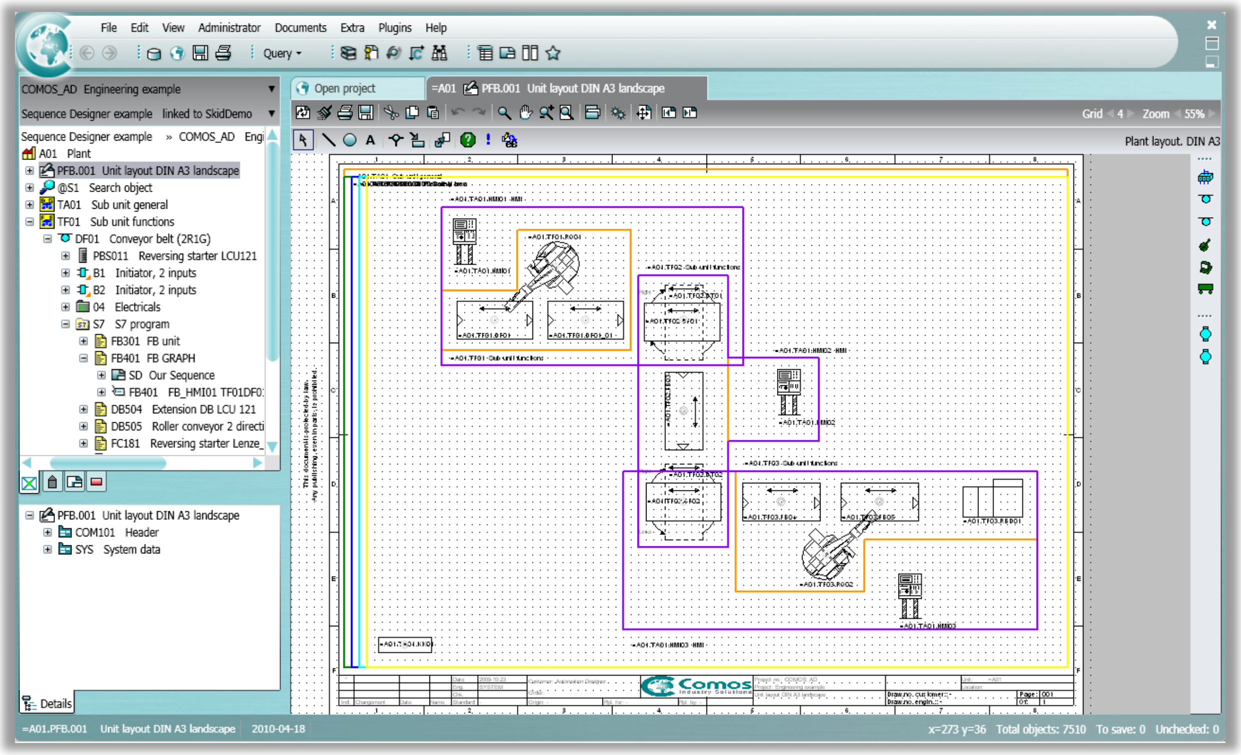
5.2.6 eBlocks

eBlocks are special base objects that Siemens provides. These are connected to templates and are used to either assign connections between different templates or to copy in other templates to an existing one in the engineering project. Using eBlocks to connect templates is for example used when assigning which objects that shall be connected to a certain terminal box or to connect PLC blocks to create a calling structure.

The “copy in eBlocks” is used in a template when there are objects that is not always wanted or when a choice between different parts should be made. In the case with the conveyor there are several different types of motors available. When adding the conveyor in an engineering project a motor type has to be chosen to be copied in. Instead of creating five large, but only slightly different conveyor templates, one large can be created and five small templates for the motors.

5.2.7 2d-layout and polygons (areas)

In Automation Designer there is the possibility to create overview drawings of the production facility. Within these drawings there is the possibility to quickly assign eBlocks to objects using polygons (see Figure 12). The polygons are drawn around the desired target objects and a source object which the objects shall be connected to is chosen. The different levels in the HMI hierarchy can be assigned by this method. One large polygon for the top level is drawn around all objects and the lower levels are divided into smaller polygons in accordance with the designer’s specifications.



File Edit View Administrator Documents Extra Plugins Help
Query
COMOS_AD Engineering example
Sequence Designer example linked to SkidDemo
Open project
=A01 PFB.001 Unit layout DIN A3 landscape
Grid 4 Zoom 55%
Plant layout. DIN A3
Sequence Designer example » COMOS_AD Engi
A01 Plant
PFB.001 Unit layout DIN A3 landscape
@S1 Search object
TA01 Sub unit general
TF01 Sub unit functions
DF01 Conveyor belt (2R1G)
PBS011 Reversing starter LCU121
B1 Initiator, 2 inputs
B2 Initiator, 2 inputs
04 Electricals
S7 S7 program
FB301 FB unit
FB401 FB GRAPH
SD Our Sequence
FB401 FB_HMI01 TF01DF0
DB504 Extension DB LCU 121
DB505 Roller conveyor 2 directi
FC181 Reversing starter Lenze_
PFB.001 Unit layout DIN A3 landscape
COM101 Header
SYS System data
Details
=A01.PFB.001 Unit layout DIN A3 landscape 2010-04-18
x=273 y=36 Total objects: 7510 To save: 0 Unchecked: 0

Figure 12 - 2D layout with polygons

6 Methodology

This chapter will briefly describe how the base project and engineering project was created in Automation Designer (AD) for this case study. It will also describe the chosen database structure in AD. In Appendix A and B there is a detailed step by step description of how base objects and templates are created and how to generate STEP 7 code and WinCC flexible screens.

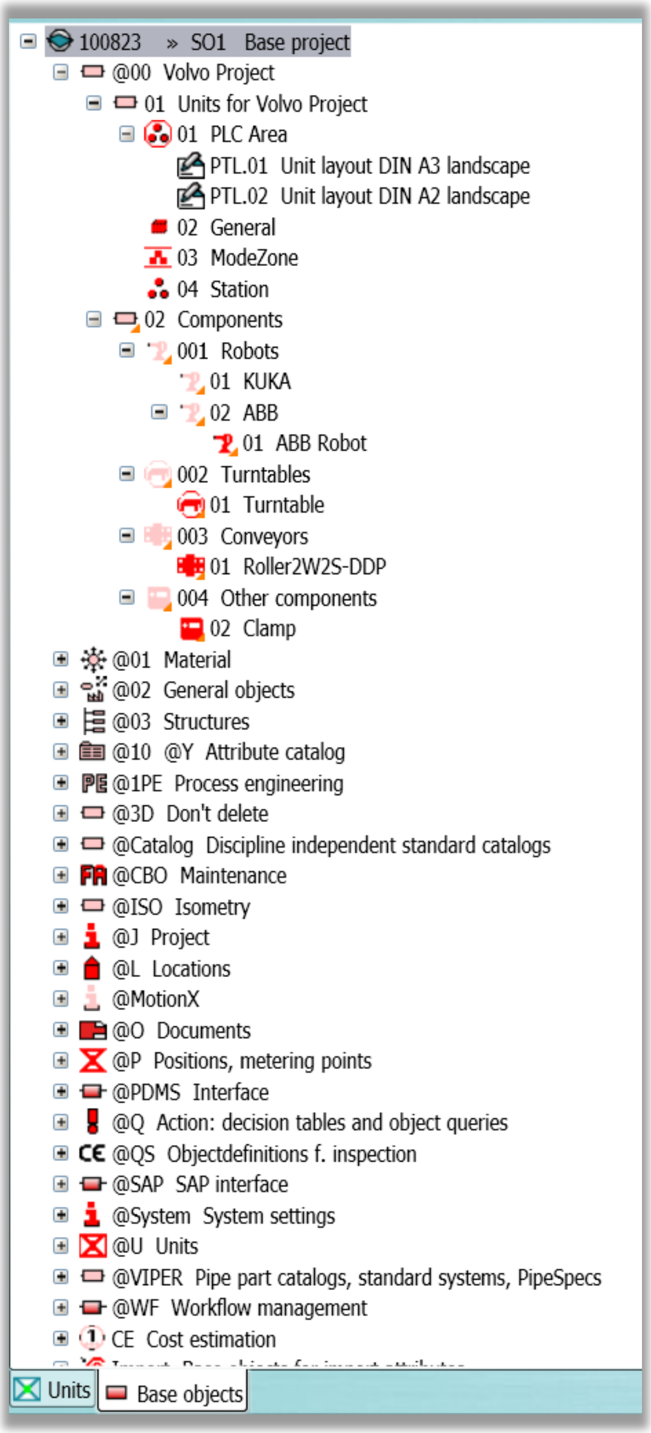
6.1 Creating the Base project for this case study

The base project is, as described in Chapter 5.2.2, the core of Automation Designer. It is in fact a database where all the information needed to create engineering projects and be able to generate PLC code and HMI screens is stored. The idea is that the base project, which includes templates and base objects, is only created once and then not modified unless there is a need to add new objects. In this case study the base project is developed to match the VCC structure, however as Automation Designer is a completely new software and a new way of working, no structure was available. Therefore a structure is developed in cooperation with Siemens AG in Germany to match VCC´s way of work as good as possible.

6.1.1 Base objects

Base objects are the lowest level in Automation Designer. Every object such as robots, conveyors, etc. is represented by a base object in the lowest level. However Siemens provides most of the base objects needed, as described in chapter 5.2.4. Most of the work for this case study, when it comes to base objects, is to modify the objects that already existed such as PLC, HMI and Sensors amongst other. When Siemens library base objects are used the objects have been copied to a separate folder and then modified. The most common modifications that have been done is scripting of names.

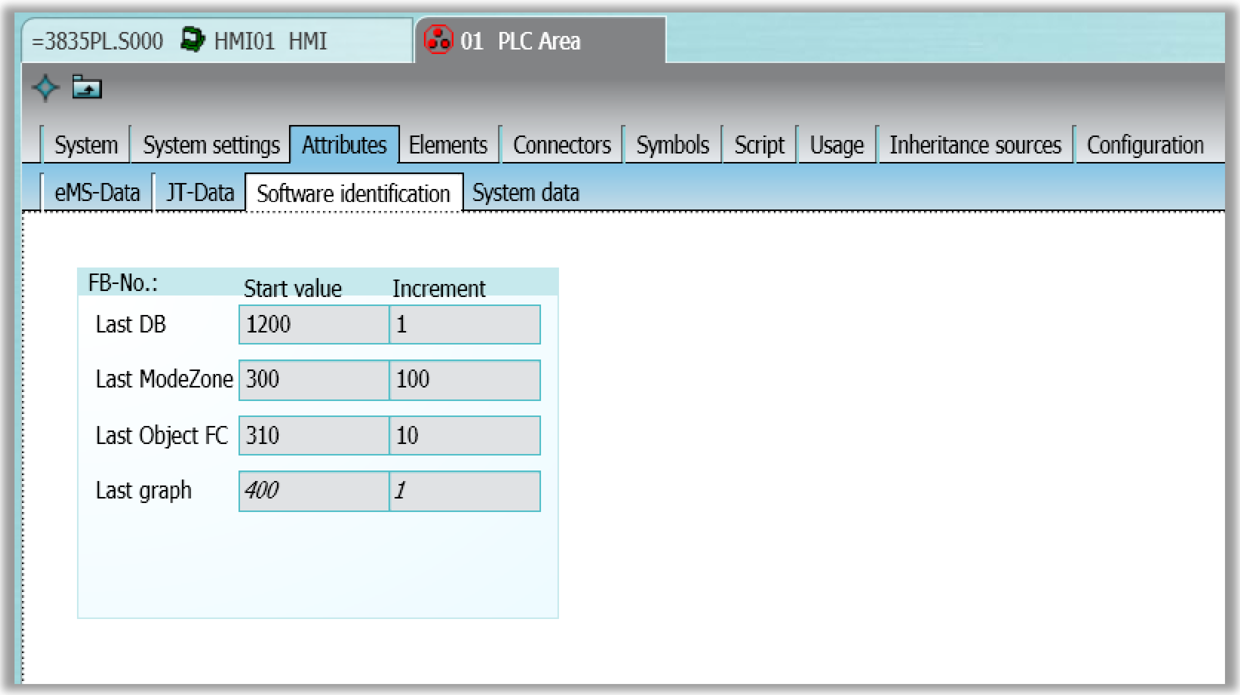
All base objects needed for this case study was not available in Automation Designer. Therefore eight new base objects were created and these are PLC Area, ModeZone, Station, General, ABB Robot, Conveyor, Turntable and Clamp (see Figure 13). As well as for the library



```
⊟ 100823  » SO1  Base project
  ⊟ @00  Volvo Project
    ⊟ 01  Units for Volvo Project
      ⊟ 01  PLC Area
            PTL.01  Unit layout DIN A3 landscape
            PTL.02  Unit layout DIN A2 landscape
          02  General
          03  ModeZone
          04  Station
    ⊟ 02  Components
      ⊟ 001  Robots
            01  KUKA
        ⊟ 02  ABB
              01  ABB Robot
      ⊟ 002  Turntables
            01  Turntable
      ⊟ 003  Conveyors
            01  Roller2W2S-DDP
      ⊟ 004  Other components
            02  Clamp
  ⊞ @01  Material
  ⊞ @02  General objects
  ⊞ @03  Structures
  ⊞ @10  @Y  Attribute catalog
  ⊞ @1PE  Process engineering
  ⊞ @3D  Don't delete
  ⊞ @Catalog  Discipline independent standard catalogs
  ⊞ @CBO  Maintenance
  ⊞ @ISO  Isometry
  ⊞ @J  Project
  ⊞ @L  Locations
  ⊞ @MotionX
  ⊞ @O  Documents
  ⊞ @P  Positions, metering points
  ⊞ @PDMS  Interface
  ⊞ @Q  Action: decision tables and object queries
  ⊞ @QS  Objectdefinitions f. inspection
  ⊞ @SAP  SAP interface
  ⊞ @System  System settings
  ⊞ @U  Units
  ⊞ @VIPER  Pipe part catalogs, standard systems, PipeSpecs
  ⊞ @WF  Workflow management
  ⊞ CE  Cost estimation
```

Figure 13 - Automation Designer database with created Base objects

objects used in this case study, these new objects needed scripts to get the right name when it is copied into an engineering project. Otherwise every object in the engineering project will have the same name. This could be solved by setting the names manually but this study tries to create as smart objects as possible which will decrease the work for the engineering in the engineering project.



FB-No.:	Start value	Increment
Last DB	1200	1
Last ModeZone	300	100
Last Object FC	310	10
Last graph	*400*	*1*

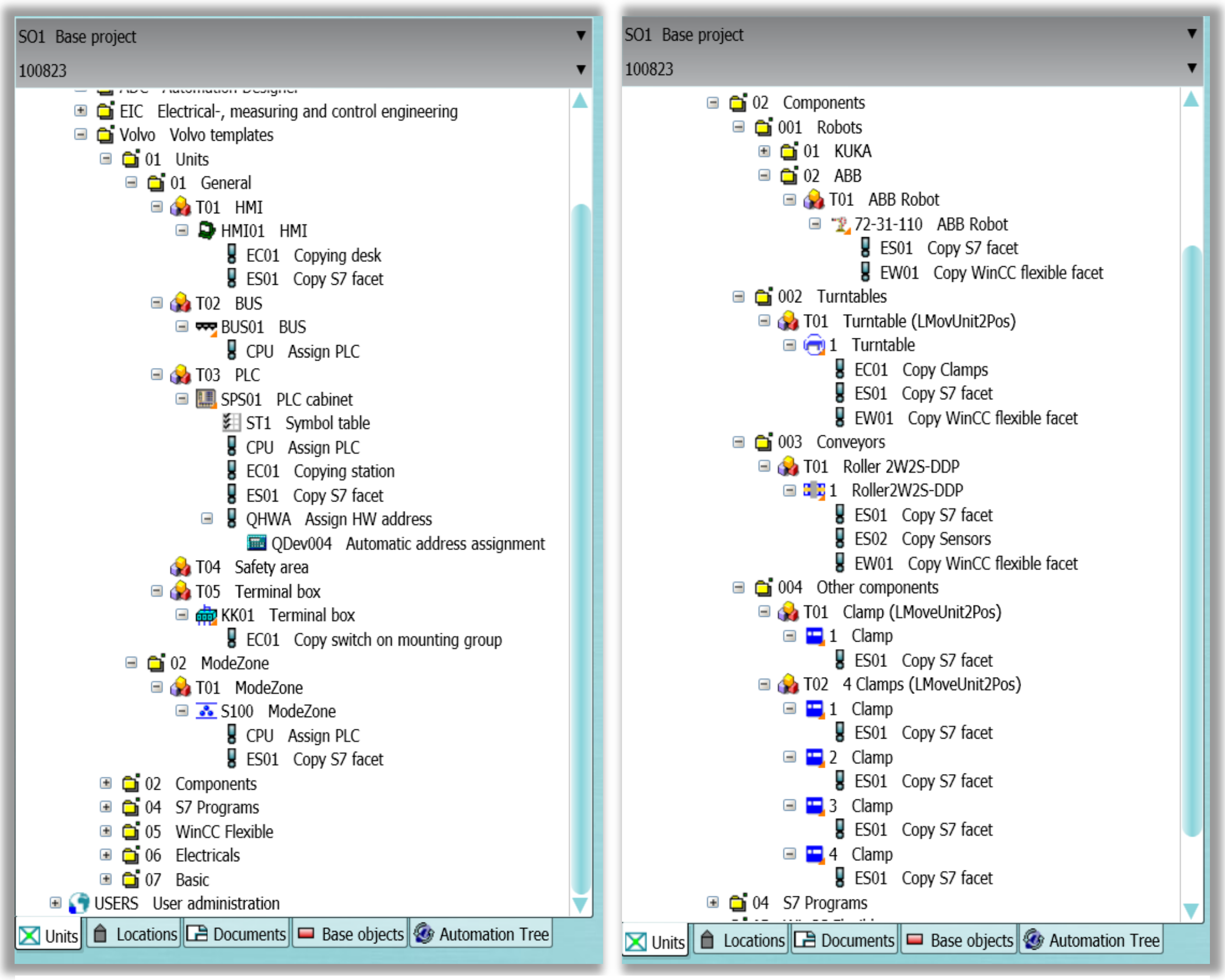
Figure 14 - Created attributes for the Base object PLC Area

Another thing done on the Base object PLC Area is to create attributes on it, other than the default ones already existing (attributes are described in chapter 5.2.4). The necessary attributes are concerning the PLC code. When STEP 7 function blocks are imported to an object in the engineering project the system needs to know which number it should assign the data blocks and the FC's. This is remembered by the help of the attributes displayed in Figure 14. Scripts are created to set the names for the data blocks and FC's in Automation Designer and the scripts are reading the attributes to know which number to assign the blocks. Attributes are also created on the Conveyor because it needs to have some specific DB names as inputs to its calling function block. The reason for this functionality is because of the VCC library block for the Conveyor that requires this.

6.1.2 Templates

A template can be seen as a collection containing information needed to represent an object (more information about templates in chapter 5.2.5). Working with template is a way to reuse information which always stays the same for an object e.g. HMI icons, HMI detail screens and library FBs. In this case study three templates have been created for every object. These are:

- Device template
- S7 Programs template
- WinCC flexible template



```
SO1  Base project
100823
  ⊞ EIC  Electrical-, measuring and control engineering
  ⊟ Volvo  Volvo templates
    ⊟ 01  Units
      ⊟ 01  General
        ⊟ T01  HMI
          ⊟ HMI01  HMI
              EC01  Copying desk
              ES01  Copy S7 facet
        ⊟ T02  BUS
          ⊟ BUS01  BUS
              CPU  Assign PLC
        ⊟ T03  PLC
          ⊟ SPS01  PLC cabinet
              ST1  Symbol table
              CPU  Assign PLC
              EC01  Copying station
              ES01  Copy S7 facet
            ⊟ QHWA  Assign HW address
                QDev004  Automatic address assignment
          T04  Safety area
        ⊟ T05  Terminal box
          ⊟ KK01  Terminal box
              EC01  Copy switch on mounting group
      ⊟ 02  ModeZone
        ⊟ T01  ModeZone
          ⊟ S100  ModeZone
              CPU  Assign PLC
              ES01  Copy S7 facet
    ⊞ 02  Components
    ⊞ 04  S7 Programs
    ⊞ 05  WinCC Flexible
    ⊞ 06  Electricals
    ⊞ 07  Basic
  ⊞ USERS  User administration
Units | Locations | Documents | Base objects | Automation Tree
```

```
SO1  Base project
100823
  ⊟ 02  Components
    ⊟ 001  Robots
      ⊞ 01  KUKA
      ⊟ 02  ABB
        ⊟ T01  ABB Robot
          ⊟ 72-31-110  ABB Robot
              ES01  Copy S7 facet
              EW01  Copy WinCC flexible facet
    ⊟ 002  Turntables
      ⊟ T01  Turntable (LMovUnit2Pos)
        ⊟ 1  Turntable
            EC01  Copy Clamps
            ES01  Copy S7 facet
            EW01  Copy WinCC flexible facet
    ⊟ 003  Conveyors
      ⊟ T01  Roller 2W2S-DDP
        ⊟ 1  Roller2W2S-DDP
            ES01  Copy S7 facet
            ES02  Copy Sensors
            EW01  Copy WinCC flexible facet
    ⊟ 004  Other components
      ⊟ T01  Clamp (LMoveUnit2Pos)
        ⊟ 1  Clamp
            ES01  Copy S7 facet
      ⊟ T02  4 Clamps (LMoveUnit2Pos)
        ⊟ 1  Clamp
            ES01  Copy S7 facet
        ⊟ 2  Clamp
            ES01  Copy S7 facet
        ⊟ 3  Clamp
            ES01  Copy S7 facet
        ⊟ 4  Clamp
            ES01  Copy S7 facet
  ⊞ 04  S7 Programs
Units | Locations | Documents | Base objects | Automation Tree
```

Figure 15 - Basic templates and their contents, Units to the left and Components to the right

The reason for not creating only one template for each object is that the user may only want to work with the PLC part and then there can be errors if the WinCC flexible part is connected to the object and not configured. Another reason is when developing the database, with its templates and base objects, it is much easier if there is a possibility to choose which part to test and search for errors. With different templates for the same equipment it is possible to control the equipment in different ways. One example of this is the Conveyor where only one device template is created but different S7 templates. By connecting different S7 templates functionality such as different speeds and directions can be chosen for one device template.

The device templates are represented by two groups, Units and Components. The units group has got templates for the ModeZone and General, but not for the PLC Area and the Station (see Figure 15). The reason for this is due to the VCC Standard where no PLC code should be connected to these levels. These levels are only represented in the hierarchy of the navigation tree in the engineering project and therefore base objects of these two levels are created but not templates. The General level contains the PLC cabinet, BUS, HMI hardware and Terminal box (see Figure 15). However, the safety area should be below General as well but safety configuration is not a part of this master thesis.

The reason for creating a General template containing information about PLC, HMI, BUS etc. is that the engineer will have a much better overview of the structure in the engineering project. There is a possibility to place these things directly below the level PLC Area, but this will result in a lot more objects on the same level and make the engineering harder if larger projects will be created in the future.

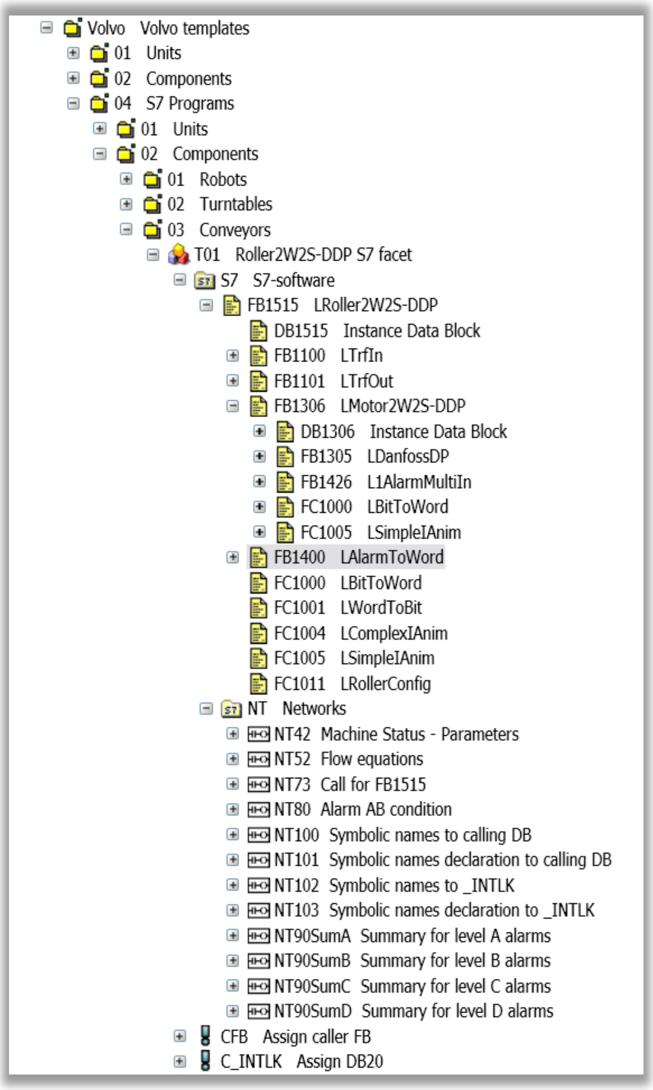
The components group is represented by four different folders; Robots, Turntables, Conveyors and Other Components. The first three is containing one template each for the represented object, while Other Components are containing two different Clamp templates, Clamp and 4 Clamps (see right picture in Figure 15). The reason for this is that there is a possibility to choose how many clamps a Turntable shall have in the engineering project. The Turntable can either have one clamp or four clamps on it and therefore one template for each possibility is needed.

6.1.2.1 S7 programs templates

To be able to generate PLC code, Automation Designer needs to know what to generate and which connections there should be between different blocks. Templates including STEP 7 software are created for the PLC, ModeZone, ABB Robot, Turntable, Clamp and Conveyor. A reason for creating S7 software templates is that the same template can be used by many objects. An example of this is the base object templates Clamp and 4 Clamps where both are using the S7 template Clamp.

In Figure 16 the S7 template for the conveyor is displayed. As seen in the figure the template T01 has got two subfolders, S7-software and Networks. The S7-software folder contains every block the conveyor type for this case study needs to work properly. The template is build up in the same calling structure as the conveyor block FB1515 in VCC´s function block library.

Inside the instance datablocks all variables are included with initial values. However the FB´s and FC´s that is available in the VCC function block library has got no code connected to them. These blocks are instead collected from a library when an STEP 7 export is performed in the engineering project. To make this work three things are needed. At first a STEP 7 function block library with the blocks included must be available and be connected to the automation tree inside the engineering project. Second the symbolic adress and symbolic name

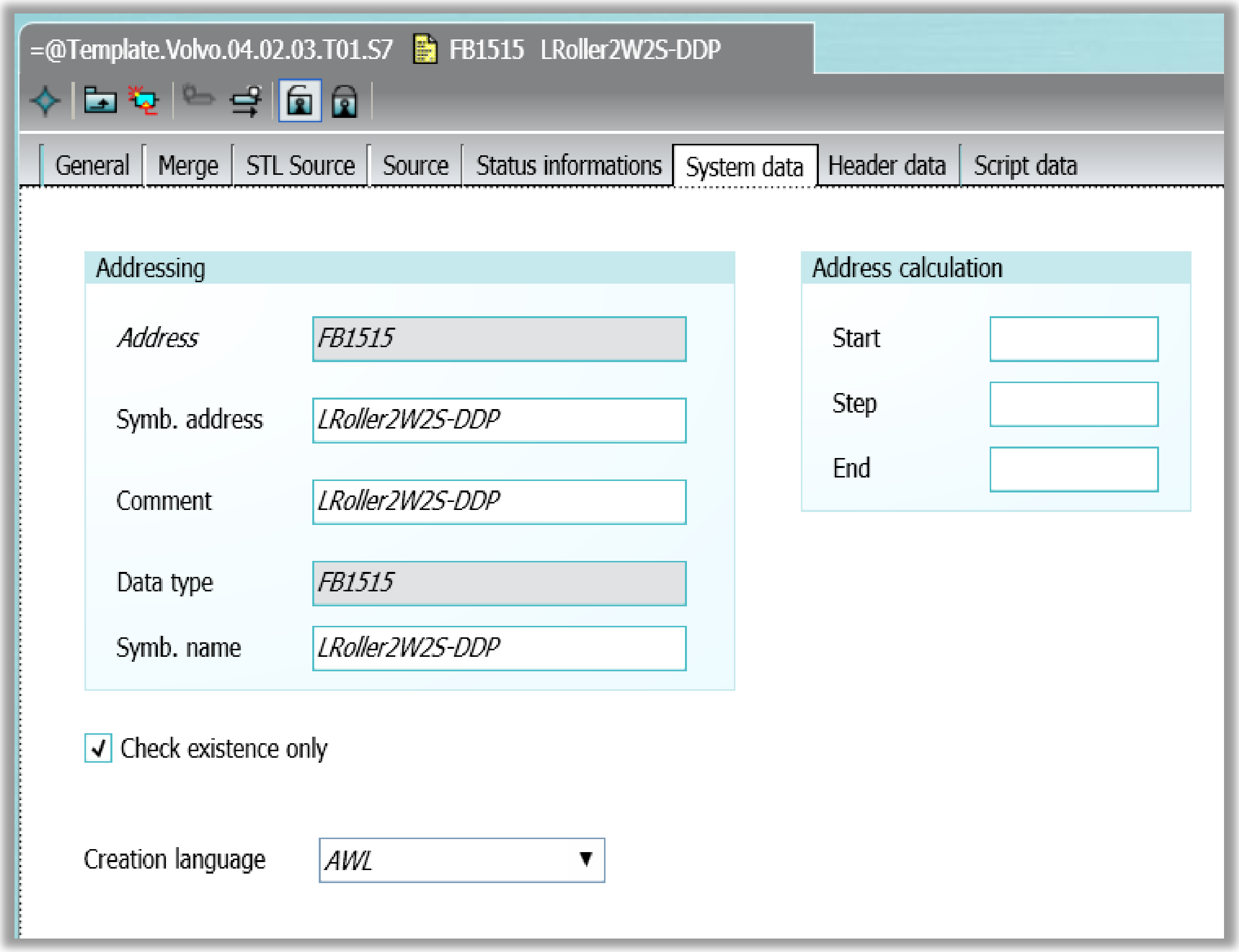


```
Volvo  Volvo templates
  01  Units
  02  Components
  04  S7 Programs
    01  Units
    02  Components
      01  Robots
      02  Turntables
      03  Conveyors
        T01  Roller2W2S-DDP S7 facet
          S7  S7-software
            FB1515  LRoller2W2S-DDP
              DB1515  Instance Data Block
              FB1100  LTrfIn
              FB1101  LTrfOut
              FB1306  LMotor2W2S-DDP
                DB1306  Instance Data Block
                FB1305  LDanfossDP
                FB1426  L1AlarmMultiIn
                FC1000  LBitToWord
                FC1005  LSimpleIAnim
              FB1400  LAlarmToWord
              FC1000  LBitToWord
              FC1001  LWordToBit
              FC1004  LComplexIAnim
              FC1005  LSimpleIAnim
              FC1011  LRollerConfig
          NT  Networks
            NT42  Machine Status - Parameters
            NT52  Flow equations
            NT73  Call for FB1515
            NT80  Alarm AB condition
            NT100  Symbolic names to calling DB
            NT101  Symbolic names declaration to calling DB
            NT102  Symbolic names to _INTLK
            NT103  Symbolic names declaration to _INTLK
            NT90SumA  Summary for level A alarms
            NT90SumB  Summary for level B alarms
            NT90SumC  Summary for level C alarms
            NT90SumD  Summary for level D alarms
          CFB  Assign caller FB
          C_INTLK  Assign DB20
```

Figure 16 - Content inside the S7-software template for the Conveyor

of the empty block must be set to the same name as the symbolic name of the block inside the STEP 7 library. Also the adress and data type needs to be the same as the block that should be copied from the STEP 7 library (see Figure 17). All this information can be set in the system data tab of the block inside Automation Designer where also the third thing can be set.

The check box “Check existence only” must be checked (see Figure 17). This tells Automation Designer that this block data should be collected from a library and no errors will be displayed bacause of an empty block. The reason for not getting the instance datablocks information from a library is that these does not have the same DB number each time.


=@Template.Volvo.04.02.03.T01.S7 FB1515 LRoller2W2S-DDP

General | Merge | STL Source | Source | Status informations | System data | Header data | Script data

Addressing

Address	FB1515
Symb. address	LRoller2W2S-DDP
Comment	LRoller2W2S-DDP
Data type	FB1515
Symb. name	LRoller2W2S-DDP

Address calculation

Start	
Step	
End	

✓ Check existence only

Creation language: AWL

Figure 17 - System data for FB1515 (function block for Conveyor) inside Automation Designer

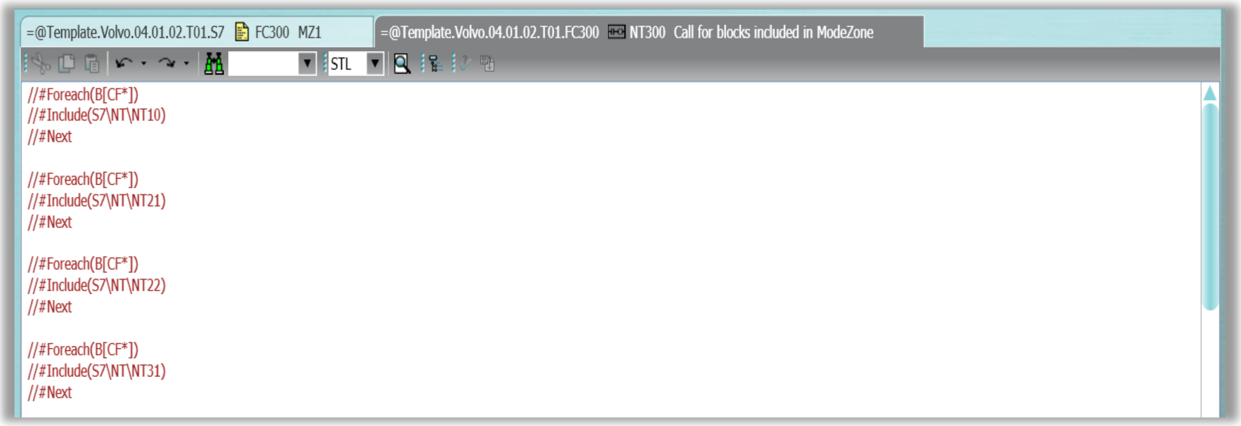
The networks displayed in Figure 16 are the ones creating the structure of the PLC code inside Automation Designer. Each network has got a special purpose that is described in Table 5.

Table 5 - Description of different Networks in Automation Designer

Network	Description
NT10	Code fitting into level 1 Inputs
NT21	Code fitting into level 2.1 Alarms - Enable Alarms
NT22	Code fitting into level 2.2 Alarms - Alarm generation
NT31	Code fitting into level 3.1 Communication – Other systems
NT32	Code fitting into level 3.2 Communication – Devices
NT41	Code fitting into level 4.1 Machine Status - Detected

NT42	Code fitting into level 4.2 Machine Status – Parameter
NT43	Code fitting into level 4.3 Machine Status – Operator selections
NT44	Code fitting into level 4.4 Machine Status – Calculated
NT45	Code fitting into level 4.5 Machine Status – Modes
NT46	Code fitting into level 4.6 Machine Status – Indications
NT51	Code fitting into level 5.1 Equations - Interlocks
NT52	Code fitting into level 5.2 Equations – Flow equations
NT60	Code fitting into level 6 Sequences
NT71	Code fitting into level 7.1 Sub modules – Non Machine
NT72	Code fitting into level 7.2 Sub modules – Command summary
NT73	Code fitting into level 7.3 Sub modules – Machine
NT80	Code fitting into level 8 Outputs
NT90SumA	Code fitting into level 9 Alarm Summary (only level A alarm summary)
NT90SumB	Code fitting into level 9 Alarm Summary (only level B alarm summary)
NT90SumC	Code fitting into level 9 Alarm Summary (only level C alarm summary)
NT90SumD	Code fitting into level 9 Alarm Summary (only level D alarm summary)
NT100	Symbolic Names with initial value that should be in the calling Data Block
NT101	Declaration of Symbolic Names that should be in the calling Data Block
NT102	Symbolic Names with initial value that should be in DB20 _INTLK
NT103	Declaration of Symbolic Names with initial value that should be in DB20 _INTLK

To get the VCC PLC structure, where the OB1 calls the ModeZone and the ModeZone calls the other objects, these networks are used inside Automation Designer. By using networks and inline macros (see Figure 18) it is possible to create a PLC structure where one block calls another and also to create FCs with internal code sorted as described in the VCC standard (see chapter 4.1.2). Each S7 template has got their own networks folder where networks with the information needed to call the blocks amongst other things are stored.

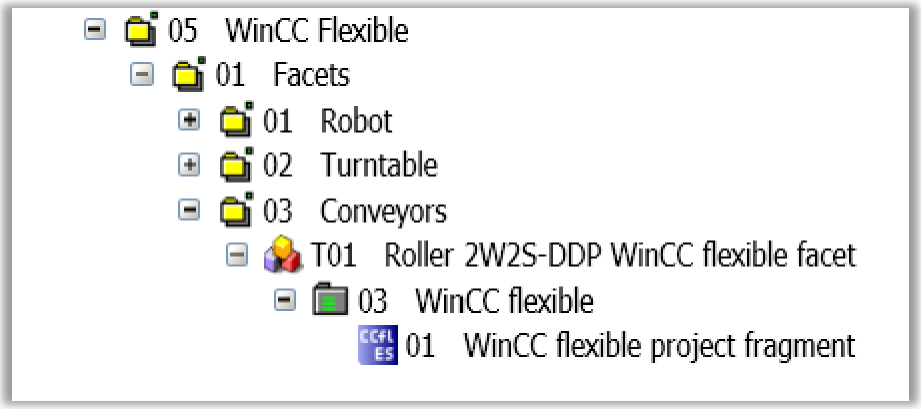

=@Template.Volvo.04.01.02.T01.S7 FC300 MZ1
=@Template.Volvo.04.01.02.T01.FC300 NT300 Call for blocks included in ModeZone
STL

```
//#Foreach(B[CF*])
//#Include(S7\NT\NT10)
//#Next

//#Foreach(B[CF*])
//#Include(S7\NT\NT21)
//#Next

//#Foreach(B[CF*])
//#Include(S7\NT\NT22)
//#Next

//#Foreach(B[CF*])
//#Include(S7\NT\NT31)
//#Next
```

Figure 18 - Inline macros to include networks

6.1.2.2 WinCC flexible templates

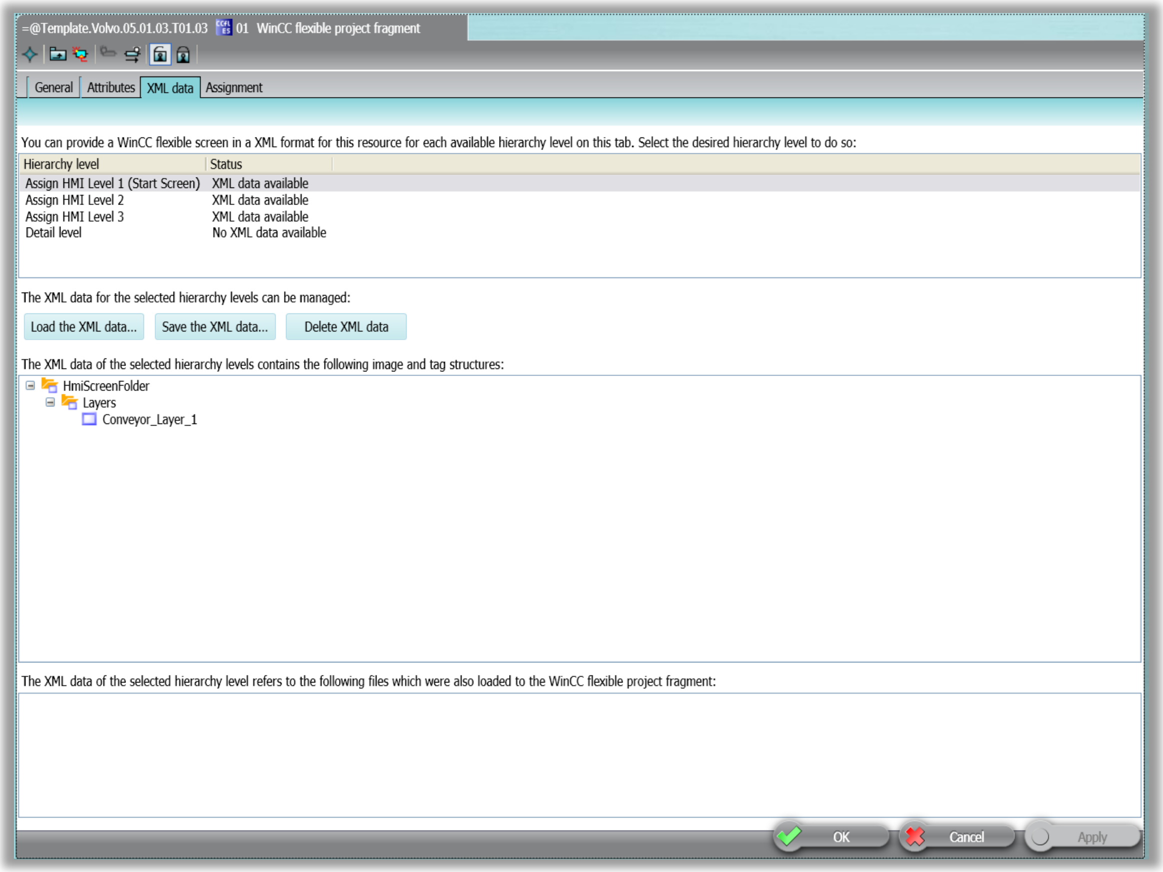
Templates for the HMI are only made for the objects that shall be represented inside the generated HMI screens. In this master thesis templates for the Robot, Turntable and Conveyor is created. The only thing placed inside the WinCC flexible template is a project fragment as shown in Figure 19.



```
05 WinCC Flexible
  01 Facets
    01 Robot
    02 Turntable
    03 Conveyors
      T01 Roller 2W2S-DDP WinCC flexible facet
        03 WinCC flexible
          01 WinCC flexible project fragment
```

Figure 19 - WinCC flexible template for Conveyor

By opening this fragment it is possible to assign XML data for the different levels of the HMI (more information about this in chapter 5.1.2). After loading of XML data to the different levels it is possible, by choosing the assignment tab, set the tags to be exported with the HMI screens. The tags is connected to the datablocks by inline makros.



=@Template.Volvo.05.01.03.T01.03 01 WinCC flexible project fragment

General | Attributes | XML data | Assignment

You can provide a WinCC flexible screen in a XML format for this resource for each available hierarchy level on this tab. Select the desired hierarchy level to do so:

Hierarchy level	Status
Assign HMI Level 1 (Start Screen)	XML data available
Assign HMI Level 2	XML data available
Assign HMI Level 3	XML data available
Detail level	No XML data available

The XML data for the selected hierarchy levels can be managed:

Load the XML data... | Save the XML data... | Delete XML data

The XML data of the selected hierarchy levels contains the following image and tag structures:

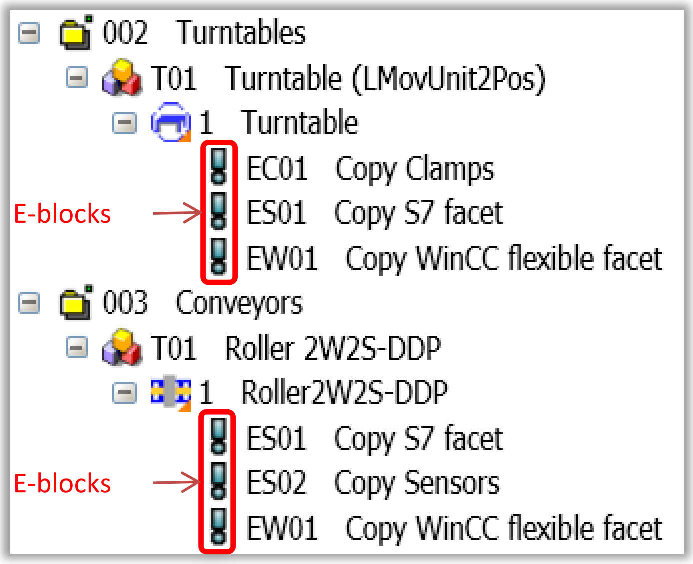
```
HmiScreenFolder
  Layers
    Conveyor_Layer_1
```

The XML data of the selected hierarchy level refers to the following files which were also loaded to the WinCC flexible project fragment:

Figure 20 - XML data for WinCC flexible template

6.1.2.3 eBlocks and Scripting

eBlocks are one of the core functions inside Automation Designer to make the connections between templates as well as the S7 code. eBlocks are placed inside the templates and can then be reached and executed inside an engineering project when the templates have been imported. The eBlocks has got an exclamation mark as a symbol in the navigation tree as seen in Figure 21. More information about how eBlocks work is described in Chapter 5.2.6.



002 Turntables
T01 Turntable (LMovUnit2Pos)
1 Turntable
EC01 Copy Clamps
ES01 Copy S7 facet
EW01 Copy WinCC flexible facet
003 Conveyors
T01 Roller 2W2S-DDP
1 Roller2W2S-DDP
ES01 Copy S7 facet
ES02 Copy Sensors
EW01 Copy WinCC flexible facet

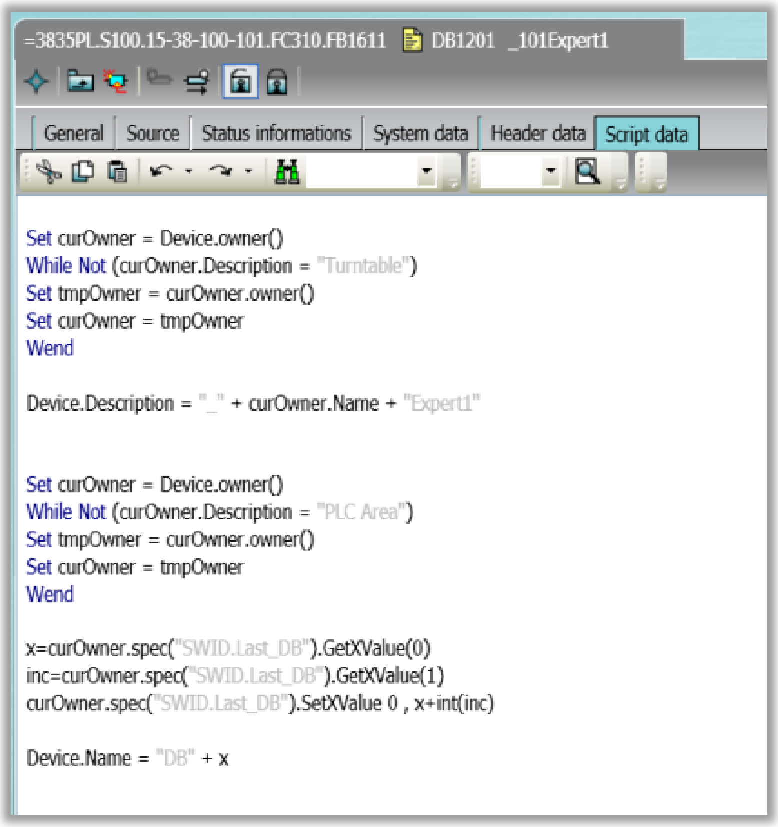
Figure 21 - eBlocks inside the object templates for Turntable and Conveyor

There are some standard eBlocks such as assign CPU and hardware interface. These are found in the database and can be put inside the templates and be used without any configuration. However for this master thesis some special eBlocks was needed to make the connection between the templates created (see Table 6). Most of them was found in already existed sample projects inside Automation Designer and these could be used by configurating them to the VCC templates.

Table 6 -Description of modified and created eBlocks for this master thesis

eBlock name	**Functionallity**
ES01 - Copy S7 facet	Imports the S7 code from the S7-software template
ES02 – Copy Sensors	Imports standard sensors available in the database to the Conveyor
EW01 – Copy WinCC flexible facet	Imports the screen and tag information from the HMI templates
EC01 – Copy Clamps	Imports the Clamps from the object template to the Turntable
CFB – Assign caller FB	Sets the connection which FC that calls which FB or FC
C_INTLK – Assign DB20	Sets the connection to the DB20 from the different objects

Scripting is another important factor when working with Automation Designer. By the help of scripting the engineer can create smart and dynamic templates which will decrease the time of work in the engineering project. Scripting can be used in a lot of places inside the software. For this project scripting is mainly used to set the right names and numbers on different things when importing them to the engineering project from the database. The best example is the script for setting the DB number of the instance datablocks. Each time the S7-software template for the Conveyor, Turntable, Clamp and ABB Robot gets imported inside the the engineering project the instance DBs must get a unique number and name according to the VCC standard (more information about the VCC standard in chapter 4). There can not be two instance DBs with the same number or name. To solve this, a script is placed inside the scriptdata tab of the data block as seen in Figure 22. The first part of the script is assigning the name of the DB by looking at the name of the object it lie beneath. Then the second part, which is assigning the DB number, looks at the attributes created on the object PLC Area. From the attribute the script can read which numer to assign the DB.

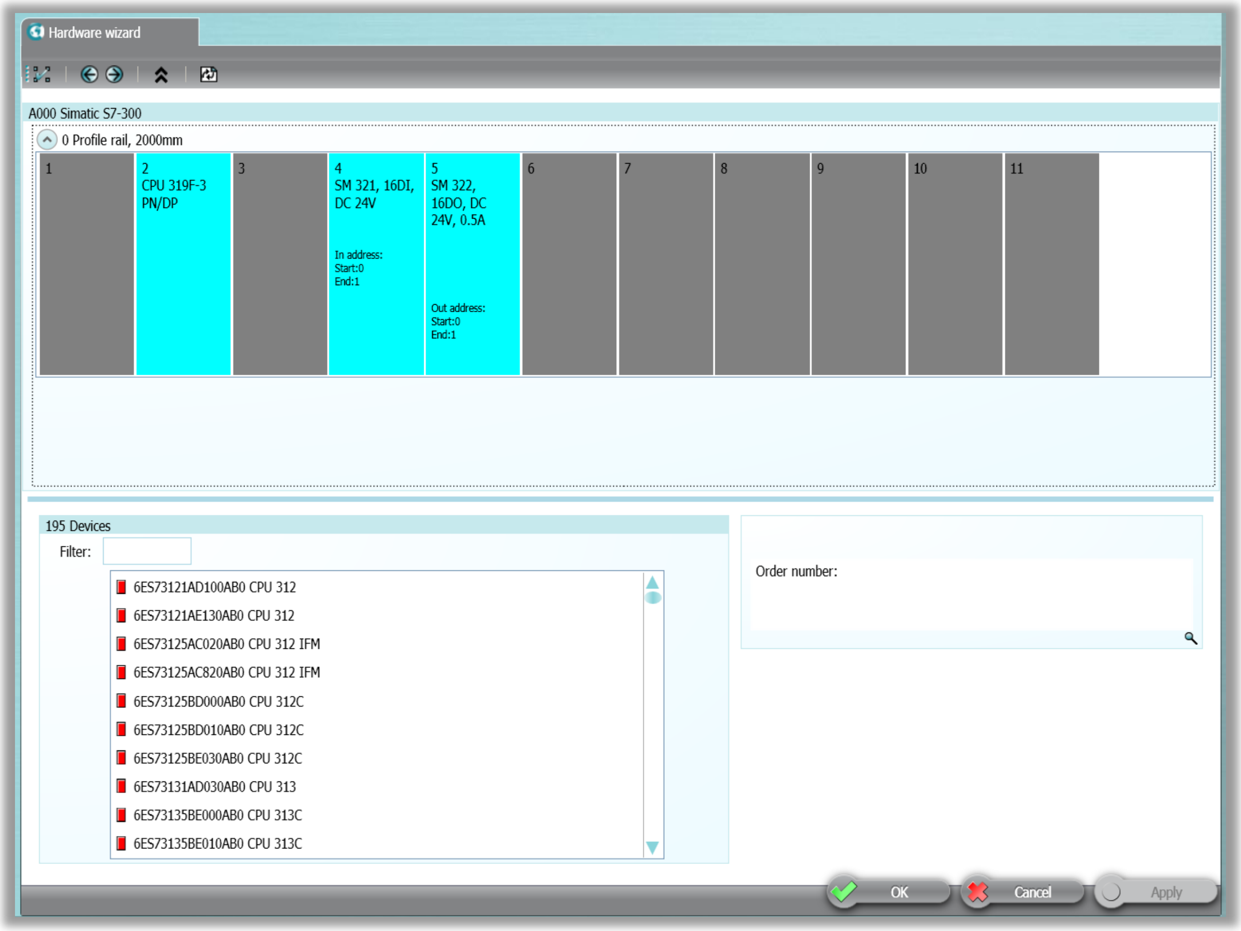


```
Set curOwner = Device.owner()
While Not (curOwner.Description = "Turntable")
Set tmpOwner = curOwner.owner()
Set curOwner = tmpOwner
Wend

Device.Description = "_" + curOwner.Name + "Expert1"


Set curOwner = Device.owner()
While Not (curOwner.Description = "PLC Area")
Set tmpOwner = curOwner.owner()
Set curOwner = tmpOwner
Wend

x=curOwner.spec("SWID.Last_DB").GetXValue(0)
inc=curOwner.spec("SWID.Last_DB").GetXValue(1)
curOwner.spec("SWID.Last_DB").SetXValue 0 , x+int(inc)

Device.Name = "DB" + x
```

Figure 22 - Script data for setting names and numbers on instance DBs

6.1.2.4 *Default settings*

If there is a standard of which hardware devices a component such as the PLC should have. Then there is a possibility to set the default values in the templates and get the right hardware each time a template is added to the engineering project. In this case study the PLC, HMI, BUS and Terminal box is configured with default equipment found in earlier created STEP 7 and WinCC flexible projects at VCC. In Figure 23 the default devices of the PLC cabinet is displayed. In the hardware wizard, displayed in the figure, it is possible to choose a slot and place a device in it. When selecting one of the slots a list of possible devices to connect to it will appear as shown in Figure 23. As Automation Designer is a Siemens software there is only the possibility to choose Siemens hardware devices at the moment. The same goes for the default settings of the HMI panels, which also can be set in the templates. The HMI is set to a default panel size and resolution in this case study.



6ES73121AD100AB0 CPU 312
6ES73121AE130AB0 CPU 312
6ES73125AC020AB0 CPU 312 IFM
6ES73125AC820AB0 CPU 312 IFM
6ES73125BD000AB0 CPU 312C
6ES73125BD010AB0 CPU 312C
6ES73125BE030AB0 CPU 312C
6ES73131AD030AB0 CPU 313
6ES73135BE000AB0 CPU 313C
6ES73135BE010AB0 CPU 313C

Figure 23 - Hardware wizard for PLC cabinet

6.2 Working with templates in the engineering project

When the base project is created, all the engineering will be performed in the engineering project of Automation Designer. The first thing to do in the engineering project is to create a structure in the navigation tree. There is two different ways to do this:

- Build up the structure with the right click of the mouse
- Build up the structure by importing an XML file from Process Simulate

Both methods is tested in this master thesis and briefly described in the following chapters.

6.2.1 Configuration of the engineering project with the right click of the mouse

In Automation Designer it is possible to choose which object to get into the navigation tree by right clicking on the different objects in it. To build up this feature some connections is needed in the database between the templates and base objects. Figure 24 shows how a selection of different objects to import is available when right clicking on an object in the navigation tree. By right click on different levels in the tree the user will get different choices. This will prevent the user from creating an invalid structure where for example a station is below a robot.

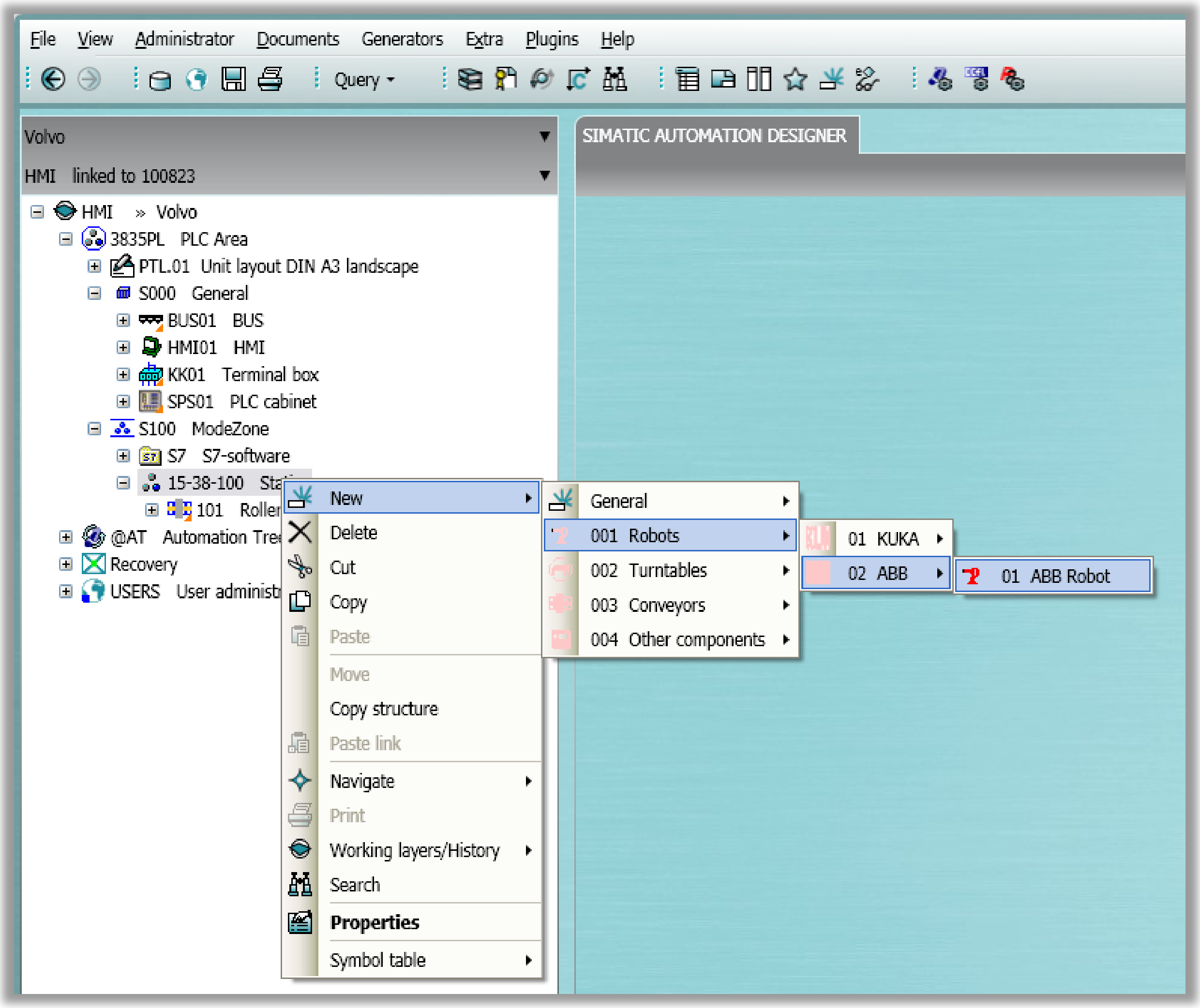


Figure 24 - Insert a new object in the navigation tree by the right click of the mouse

When the tree structure is created with the right click there are a couple of steps needed to be done before an export to either WinCC flexible or STEP 7 is possible. These are

1. Configure devices inside the General folder
2. Configure all objects below the stations
3. Configure the ModeZone
4. Place the objects on the 2D layout
5. Drag polygons around the objects on the 2D layout and calculate eBlocks
6. Configure hardware addresses and connect the sensors to the terminal box
7. Generate the symbol table

How to do these steps is described in Appendix A. The final structure of the three stations in this case study with the 2D layout is displayed in Figure 25.

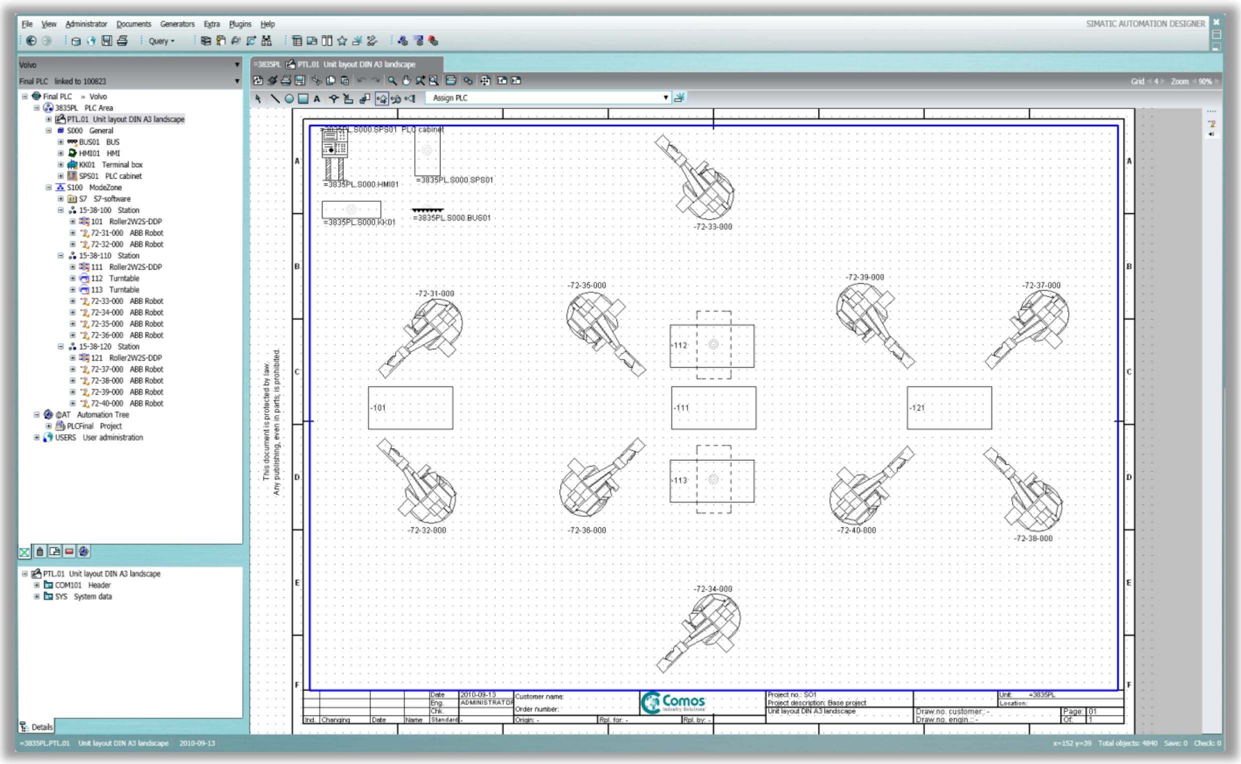


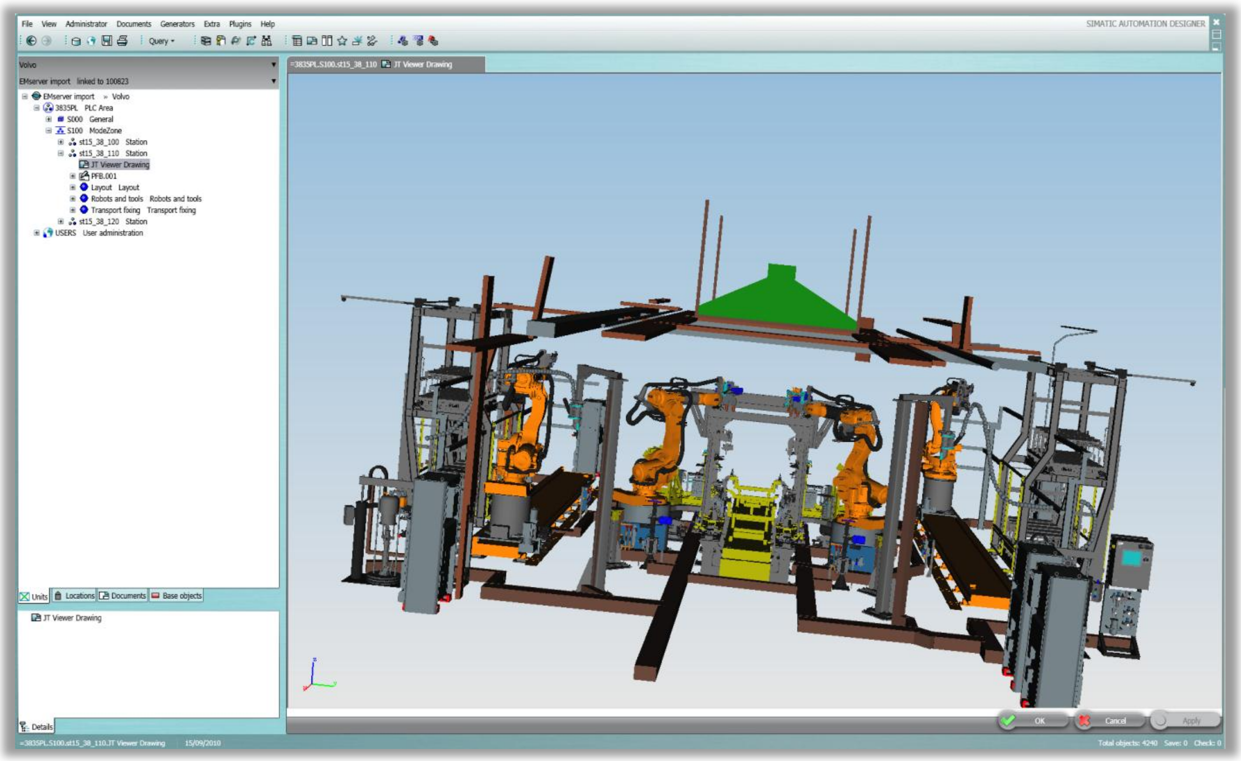
Figure 25 - Final tree structure with 2D-layout

6.2.2 Configuration of the project by importing an XML file from Process Simulate

It is possible to import an XML file from Process Simulate and get the tree structure from the simulation study. By importing a XML file will also give the user a 2D layout inside Automation Designer including the right objects. Another thing the user will get in his navigation tree is a JT drawing which can be opened inside Automation Designer. This drawing will show the 3D environment that has been built up in Process Simulate and Process Designer (see Figure 26). As there is a lot of different component such as fences that is needed in Process Simulate but not in Automation Designer, there is a possibility to choose what to not import from the XML file.

The configuration needed for this case study to make it possible to import an XML file is:

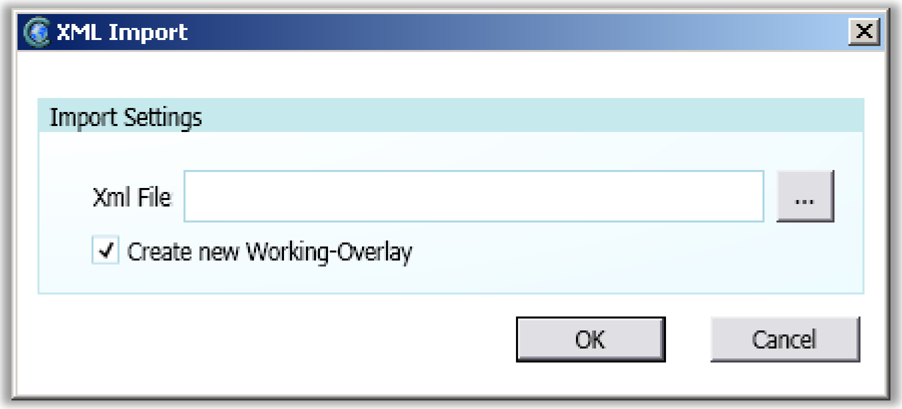
1. The object names in the XML file needs to be connected to the names of the templates inside Automation Designer.
2. A structure to the ModeZone level in the navigation tree needs to be built up with the right mouse click. This is because the XML file at VCC only includes structure from the Station level.
3. An XML configuration file needs to be edited. The file is available in the system root of the software.


File View Administrator Documents Generators Extra Plugins Help
Query
SIMATIC AUTOMATION DESIGNER
Volvo
EMserver import linked to 100823
=3835PL.S100.st15_38_110 JT Viewer Drawing
Units Locations Documents Base objects
Details
OK Cancel Apply

EMserver import » Volvo
3835PL PLC Area
S000 General
S100 ModeZone
st15_38_100 Station
st15_38_110 Station
JT Viewer Drawing
PFB.001
Layout Layout
Robots and tools Robots and tools
Transport fixing Transport fixing
st15_38_120 Station
USERS User administration

Figure 26 - JT-drawing of station 15-38-110 inside Automation Designer

When all this is performed the XML file can be imported by right clicking on the ModeZone level and choosing “EMserver Import”. A new window comes up and the path to the XML file needs to be chosen as seen in Figure 27. In the window there is also a possibility to choose if Automation Designer shall create a new working layer in the engineering project when importing the XML file or not. This box is always checked as default. The XML import feature is not yet a standard feature inside Automation Designer and the possibility to do this have been given to this case study exclusively.



Import Settings

Xml File

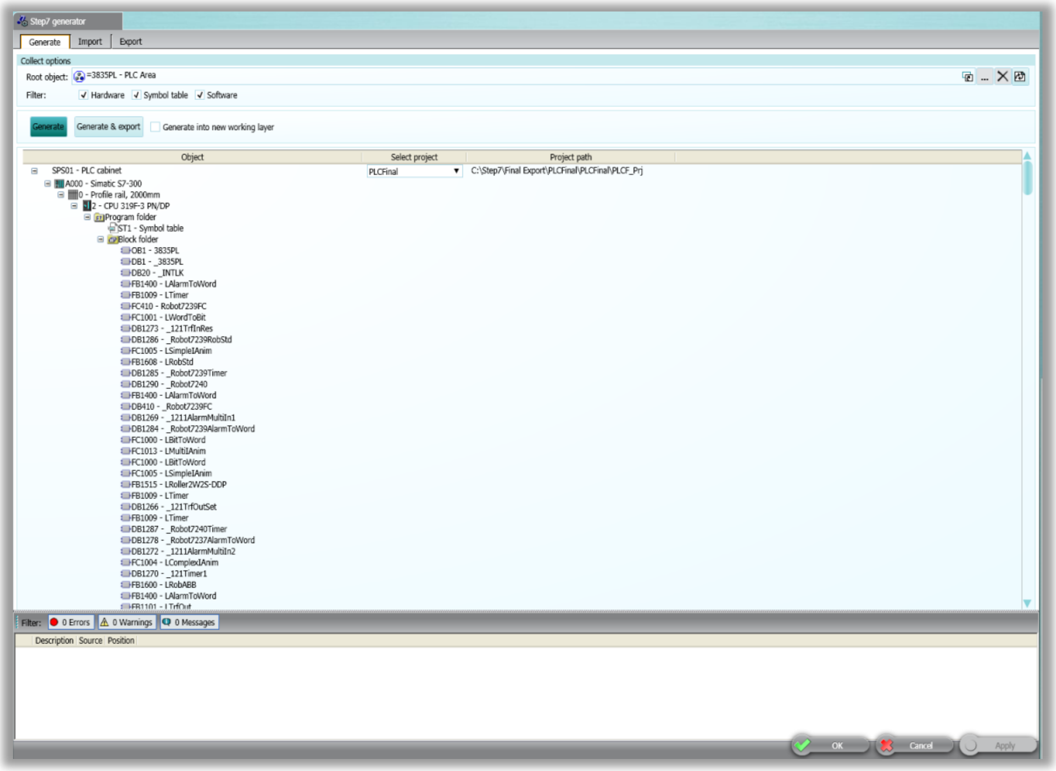
[✓] Create new Working-Overlay

Figure 27 - Pop-up window where an XML file needs to be chosen when importing information from EMserver

The final step is to import the XML file and get the structure with the objects included in the simulation study. How the structure will appear in Automation Designer is presented in chapter 7 Results.

6.2.3 Exporting the engineering project to STEP 7

To be able to generate PLC code to STEP 7 an engineering project must be built up and configured in the right way. When this is completed the export of the engineering project to STEP 7 is quite easy. Inside Automation Designer there is an export to STEP 7 icon in the tool bar of the software. This icon will open a window where the entire project can be dragged to as seen in Figure 28. Before an export is possible Automation Designer needs to generate the STEP 7 project. This will create the final blocks that will be exported as well as the calling structure of the PLC project. It is also in the generation part where the name of the STEP 7 project is set and the path where it should be stored.



Collect options
Root object: =3835PL - PLC Area
Filter: ✓ Hardware ✓ Symbol table ✓ Software
☐ Generate into new working layer

Object	Select project	Project path
SPS01 - PLC cabinet	PLCFinal	C:\Step7\Final Export\PLCFinal\PLCFinal\PLCF_Prj

```
SPS01 - PLC cabinet
  A000 - Simatic S7-300
    0 - Profile rail, 2000mm
      2 - CPU 319F-3 PN/DP
        Program folder
          ST1 - Symbol table
          Block folder
            OB1 - 3835PL
            DB1 - _3835PL
            DB20 - _INTLK
            FB1400 - LAlarmToWord
            FB1009 - LTimer
            FC410 - Robot7239FC
            FC1001 - LWordToBit
            DB1273 - _121TrfInRes
            DB1286 - _Robot7239RobStd
            FC1005 - LSimpleIAnim
            FB1608 - LRobStd
            DB1285 - _Robot7239Timer
            DB1290 - _Robot7240
            FB1400 - LAlarmToWord
            DB410 - _Robot7239FC
            DB1269 - _1211AlarmMultiIn1
            DB1284 - _Robot7239AlarmToWord
            FC1000 - LBitToWord
            FC1013 - LMultiIAnim
            FC1000 - LBitToWord
            FC1005 - LSimpleIAnim
            FB1515 - LRoller2W2S-DDP
            FB1009 - LTimer
            DB1266 - _121TrfOutSet
            FB1009 - LTimer
            DB1287 - _Robot7240Timer
            DB1278 - _Robot7237AlarmToWord
            DB1272 - _1211AlarmMultiIn2
            FC1004 - LComplexIAnim
            DB1270 - _121Timer1
            FB1600 - LRobABB
            FB1400 - LAlarmToWord
            FB1101 - LTrfOut
```

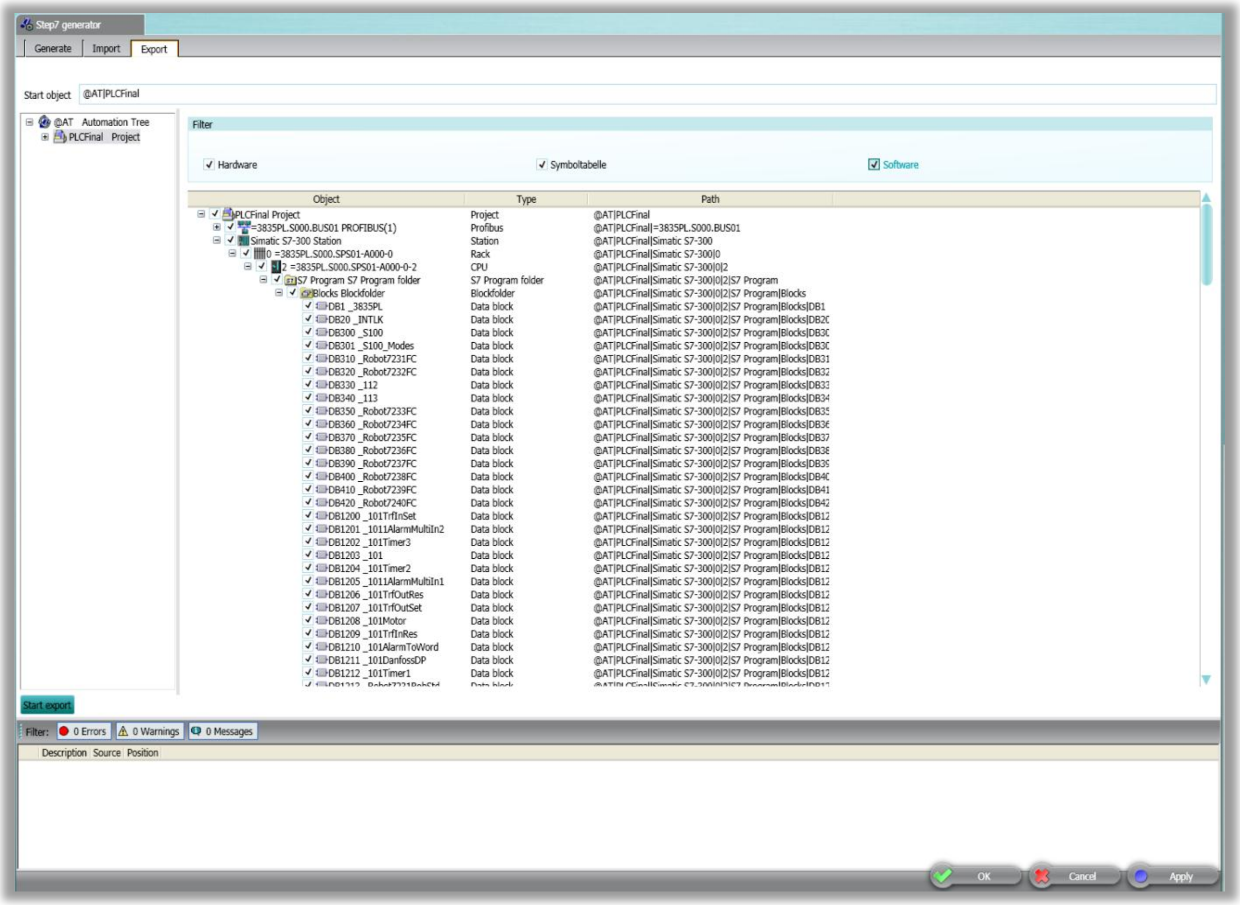
Filter: ● 0 Errors ⚠ 0 Warnings 0 Messages

Description	Source	Position

Figure 28 - STEP 7 generator

The generated project will be placed in the tab automation tree inside Automation Designer. In this tree it is possible to go into every block and see what code that has been generated. After the generation it is also possible to connect a STEP 7 function block library from where Automation Designer will get the library blocks needed. This is done in the properties of the generated project inside Automation Designer. For this case study a library named VolvoLib is connected to every generated project as default. The connection to a STEP 7 library is done by pointing at a search path from where the library is available.

The last step is to export the generated project to STEP 7. This is done by selecting the generated project and then selecting what to export from the project (software, hardware, symbol table) as seen in Figure 29. The final step is to click on the export button which will enable the export of the project and give errors if something is wrong with the PLC code. If nothing is wrong it is possible to go into SIMATIC Manager and open the project to see what is generated and continue working with the project if needed.



Start object @AT|PLCFinal

@AT Automation Tree
PLCFinal Project

Filter

✓ Hardware ✓ Symboltabelle ✓ Software

Object	Type	Path
✓ PLCFinal Project	Project	@AT|PLCFinal
✓ =3835PL.S000.BUS01 PROFIBUS(1)	Profibus	@AT|PLCFinal|=3835PL.S000.BUS01
✓ Simatic S7-300 Station	Station	@AT|PLCFinal|Simatic S7-300
✓ 0 =3835PL.S000.SPS01-A000-0	Rack	@AT|PLCFinal|Simatic S7-300|0
✓ 2 =3835PL.S000.SPS01-A000-0-2	CPU	@AT|PLCFinal|Simatic S7-300|0|2
✓ S7 Program S7 Program folder	S7 Program folder	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program
✓ Blocks Blockfolder	Blockfolder	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks
✓ DB1 _3835PL	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB1
✓ DB20 _INTLK	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB2
✓ DB300 _S100	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB3
✓ DB301 _S100_Modes	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB3
✓ DB310 _Robot7231FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB31
✓ DB320 _Robot7232FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB32
✓ DB330 _112	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB33
✓ DB340 _113	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB34
✓ DB350 _Robot7233FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB3
✓ DB360 _Robot7234FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB3
✓ DB370 _Robot7235FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB37
✓ DB380 _Robot7236FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB3
✓ DB390 _Robot7237FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB3
✓ DB400 _Robot7238FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB4
✓ DB410 _Robot7239FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB41
✓ DB420 _Robot7240FC	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB42
✓ DB1200 _101TrfInSet	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1201 _1011AlarmMultiIn2	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1202 _101Timer3	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1203 _101	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1204 _101Timer2	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1205 _1011AlarmMultiIn1	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1206 _101TrfOutRes	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1207 _101TrfOutSet	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1208 _101Motor	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1209 _101TrfInRes	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1210 _101AlarmToWord	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1211 _101DanfossDP	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12
✓ DB1212 _101Timer1	Data block	@AT|PLCFinal|Simatic S7-300|0|2|S7 Program|Blocks|DB12

Figure 29 - Export tab inside the STEP 7 generator

A detailed step by step description of how the STEP 7 generation and export is done for the case study project is described in Appendix A.

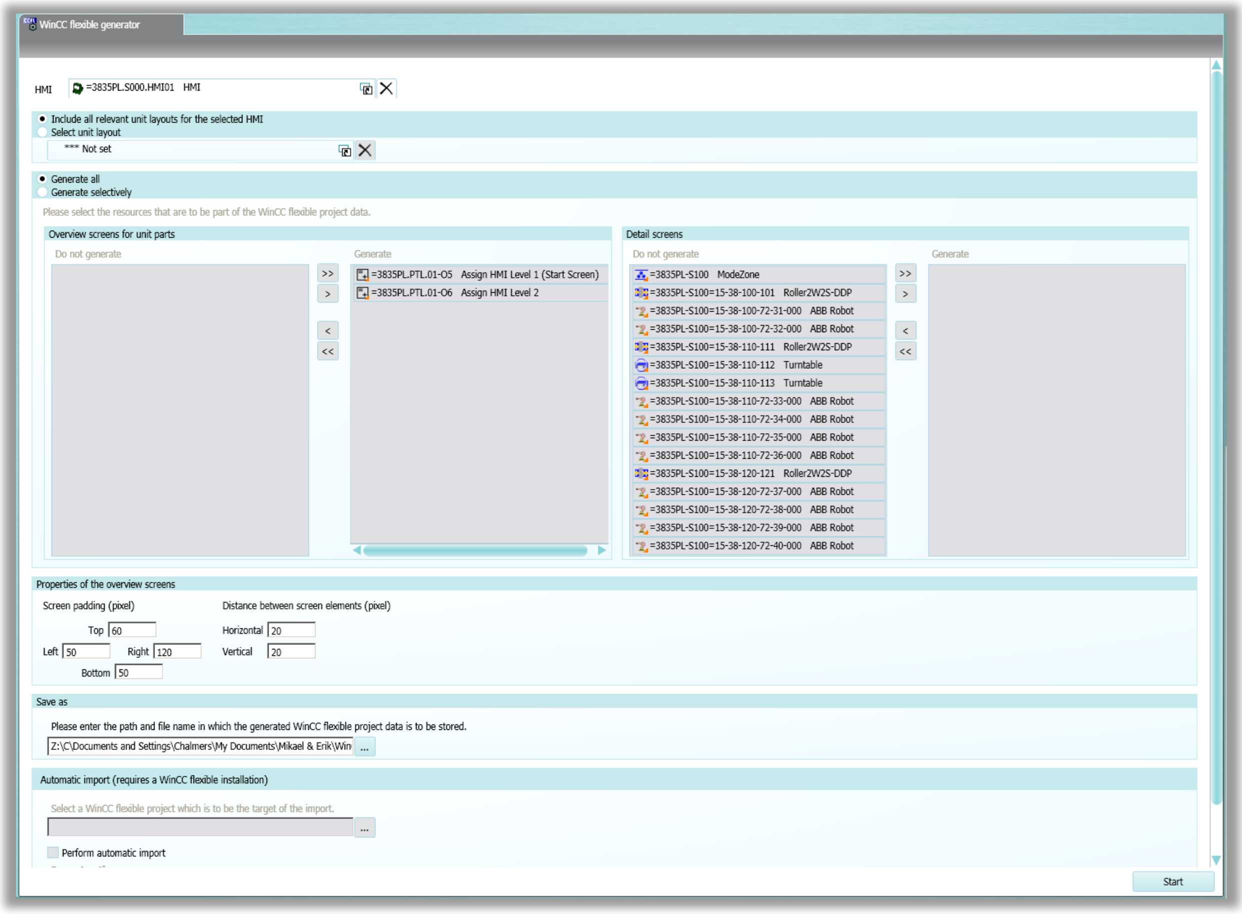
6.2.4 Exporting the engineering project to WinCC flexible

An object in Automation Designer can be represented in the HMI screens in four different levels; HMI level 1-3 and detail level. The HMI levels are designed to be a combination of several objects while the detail level only represents a single object. In the VCC standard the detail level is comparable with the library function blocks for e.g. the conveyor. For this project all levels except HMI level 3 are used. Level 1 represents OB1, level 2 the ModeZone and the detail screens are used for the ABB Robot, Turntable and Conveyor. To assign objects to a HMI level polygons are drawn on the 2D-layout around the desired objects. The design of each object icon in each level is done in WinCC flexible and screens are exported from there as XML files. One file per object and level is created and imported into the WinCC flexible templates where the files are assigned to its appropriate level. Each level can also contain tags that are used to connect the HMI to signals in the PLC program.

The VCC standard allows for two different types of HMI signals, H_Anim which contains the information for the currently viewed function block diagram (FBD) in a HMI and H_IAnim which contains information from an underlying FBD. These tags are included in the single exported XML files from WinCC flexible and have to be assigned an address in Automation Designer. To be able to assign addresses to these tags an object called DB variables must be added below the DB from where the tag should be connected to. For instance DBs this has been done in the templates where the DB variable is manually given the same address as the signal have got in the DB. Unless the library blocks are changed these DB variables will never have to be changed or updated.

The tags used to connect signals from the global DBs (H_IAnim signals) also need to be connected to a DB variable. However, the DB variable must be manually created inside the engineering project and cannot be predefined in the templates. The reason for this is Automation Designer which cannot create the address of DB variables and set the right names, depending on how many objects the user have got in his engineering project. Therefore this has to be done manually. However, a DB variable only needs to be created below the DB from where the WinCC flexible tag needs to be connected and be assigned a label which is the same as the signal it shall represent from the DB. After this is done the DB variable automatically calculates the signals address within the DB.

When the DB variables have been set and the HMI levels have been assigned, an export is possible. In the Automation Designer WinCC flexible generator there is the possibility to choose which screens to export, how far from the edges of the screen icons should be placed and the distances between each object (see Figure 30). The screens are exported as one single XML file which can be imported into a new WinCC flexible project and be applied with the final configurations.



HMI: =3835PL.S000.HMI01 HMI

[● Include all relevant unit layouts for the selected HMI]
○ Select unit layout

*** Not set

[● Generate all]
○ Generate selectively

Please select the resources that are to be part of the WinCC flexible project data.

Overview screens for unit parts

Do not generate	Generate
	=3835PL.PTL.01-O5 Assign HMI Level 1 (Start Screen)
	=3835PL.PTL.01-O6 Assign HMI Level 2

Detail screens

Do not generate	Generate
=3835PL-S100 ModeZone	
=3835PL-S100=15-38-100-101 Roller2W2S-DDP	
=3835PL-S100=15-38-100-72-31-000 ABB Robot	
=3835PL-S100=15-38-100-72-32-000 ABB Robot	
=3835PL-S100=15-38-110-111 Roller2W2S-DDP	
=3835PL-S100=15-38-110-112 Turntable	
=3835PL-S100=15-38-110-113 Turntable	
=3835PL-S100=15-38-110-72-33-000 ABB Robot	
=3835PL-S100=15-38-110-72-34-000 ABB Robot	
=3835PL-S100=15-38-110-72-35-000 ABB Robot	
=3835PL-S100=15-38-110-72-36-000 ABB Robot	
=3835PL-S100=15-38-120-121 Roller2W2S-DDP	
=3835PL-S100=15-38-120-72-37-000 ABB Robot	
=3835PL-S100=15-38-120-72-38-000 ABB Robot	
=3835PL-S100=15-38-120-72-39-000 ABB Robot	
=3835PL-S100=15-38-120-72-40-000 ABB Robot	

Properties of the overview screens

Screen padding (pixel)

Top: 60
Left: 50
Right: 120
Bottom: 50

Distance between screen elements (pixel)

Horizontal: 20
Vertical: 20

Save as

Please enter the path and file name in which the generated WinCC flexible project data is to be stored.

Z:\C\Documents and Settings\Chalmers\My Documents\Mikael & Erik\Win

Automatic import (requires a WinCC flexible installation)

Select a WinCC flexible project which is to be the target of the import.

☐ Perform automatic import

Figure 30 - WinCC flexible generator inside Automation Designer

7 Results

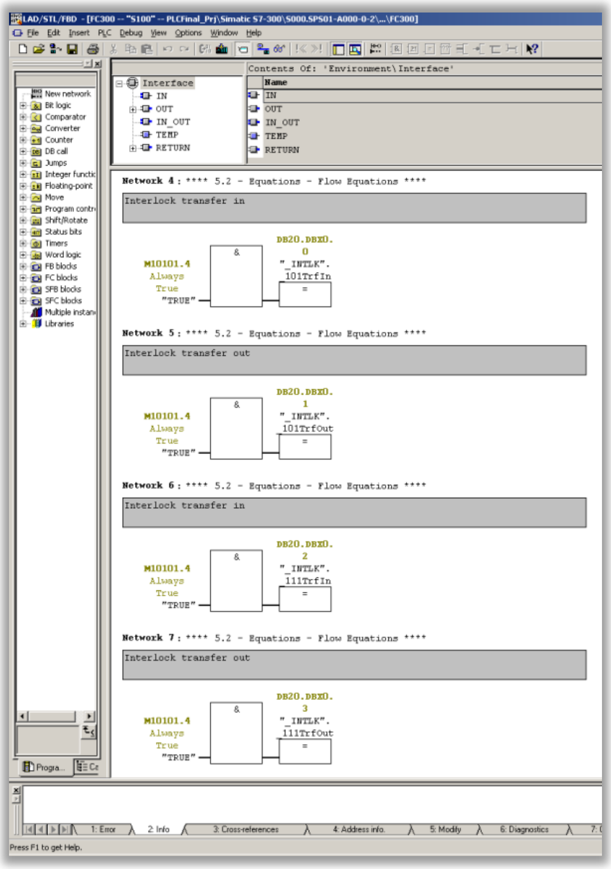
This chapter displays the results of the generated STEP 7 project and the generated WinCC flexible screens within this case study. The result also includes which information that has been transferred from Process Simulate into this project.

7.1 STEP 7 code generation

Automation Designer is able to generate and export the STEP 7 project according to the VCC standard. All the library FBs and FCs which were needed to be copied from the VCC function block library was successfully transferred by Automation Designer into the generated STEP 7 project. The generated STEP 7 project does not contain any function blocks or DBs other than the ones needed and used for the case study project. OB1 was also successfully generated and a modular program structure was achieved.

7.1.1 Generated FCs

In this master thesis there are FCs generated for three different objects; ModeZone, ABB Robot and Turntable. The generation of these FCs works all in the same way. All code is generated by the network structure described in the VCC standard (chapter 4.1). The generated FCs includes the calls for the underlying library FBs and FCs and variables or static values are connected to the inputs and outputs on the calling functions. In the FCs there are also summations of the existing alarms which will be sent up through the hierarchy according to the VCC standard in chapter 4.1.1. Placeholders have also been created with standard interlocks signals set to a default value. This is done to show the engineer which signals that requires interlock calculations (see Figure 31).

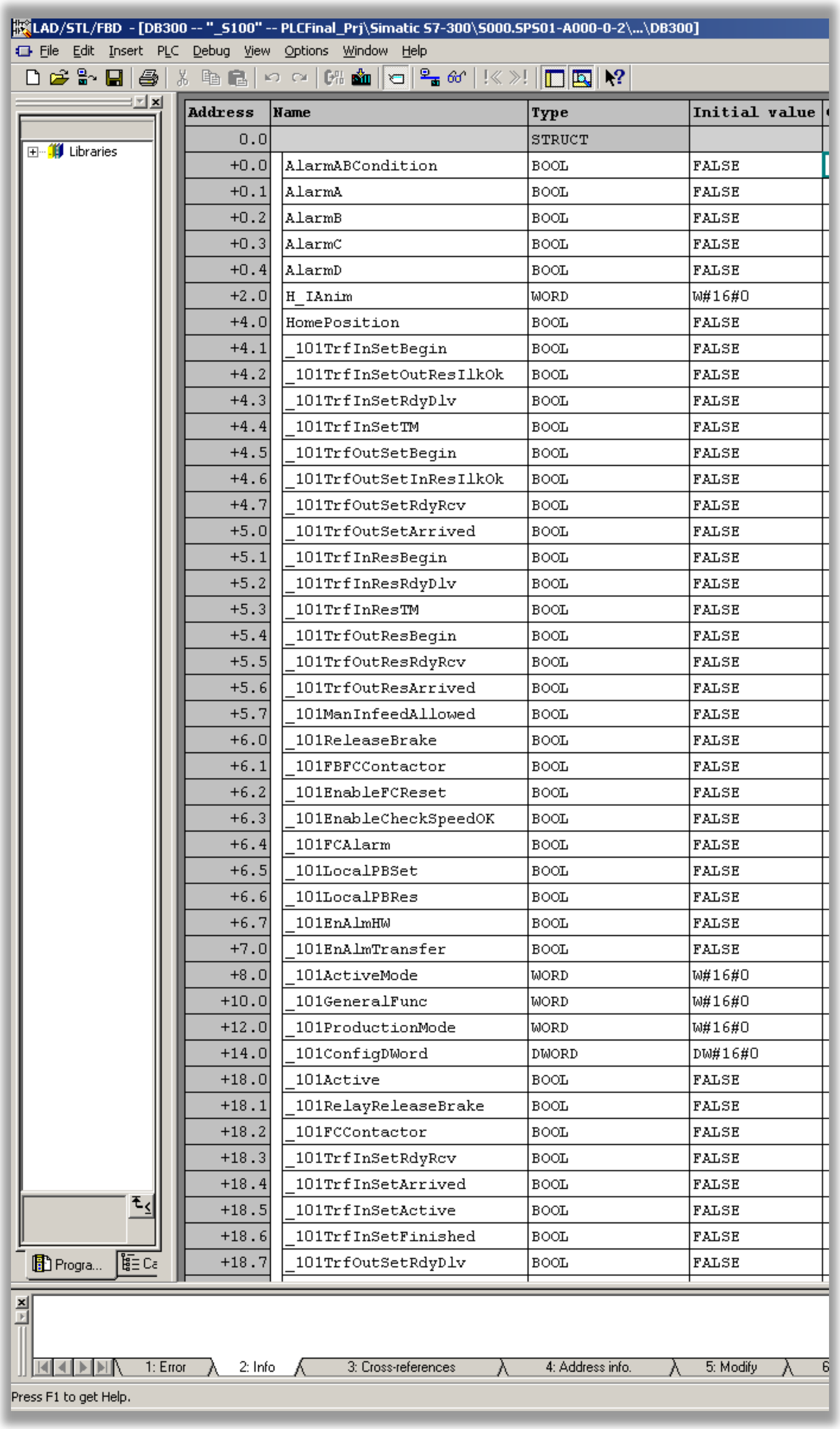


```
LAD/STL/FBD - [FC300 -- "S100" -- PLCFinal_Prj\Simatic S7-300\S000.SPS01-A000-0-2\...\FC300]

Contents Of: 'Environment\Interface'

Network 4 : **** 5.2 - Equations - Flow Equations ****
Interlock transfer in
M10101.4 Always True "TRUE" — & — DB20.DBX0.0 "_INTLK"._101TrfIn =

Network 5 : **** 5.2 - Equations - Flow Equations ****
Interlock transfer out
M10101.4 Always True "TRUE" — & — DB20.DBX0.1 "_INTLK"._101TrfOut =

Network 6 : **** 5.2 - Equations - Flow Equations ****
Interlock transfer in
M10101.4 Always True "TRUE" — & — DB20.DBX0.2 "_INTLK"._111TrfIn =

Network 7 : **** 5.2 - Equations - Flow Equations ****
Interlock transfer out
M10101.4 Always True "TRUE" — & — DB20.DBX0.3 "_INTLK"._111TrfOut =
```

Figure 31 - Placeholders for the interlock calculations

7.1.2 Generated DBs

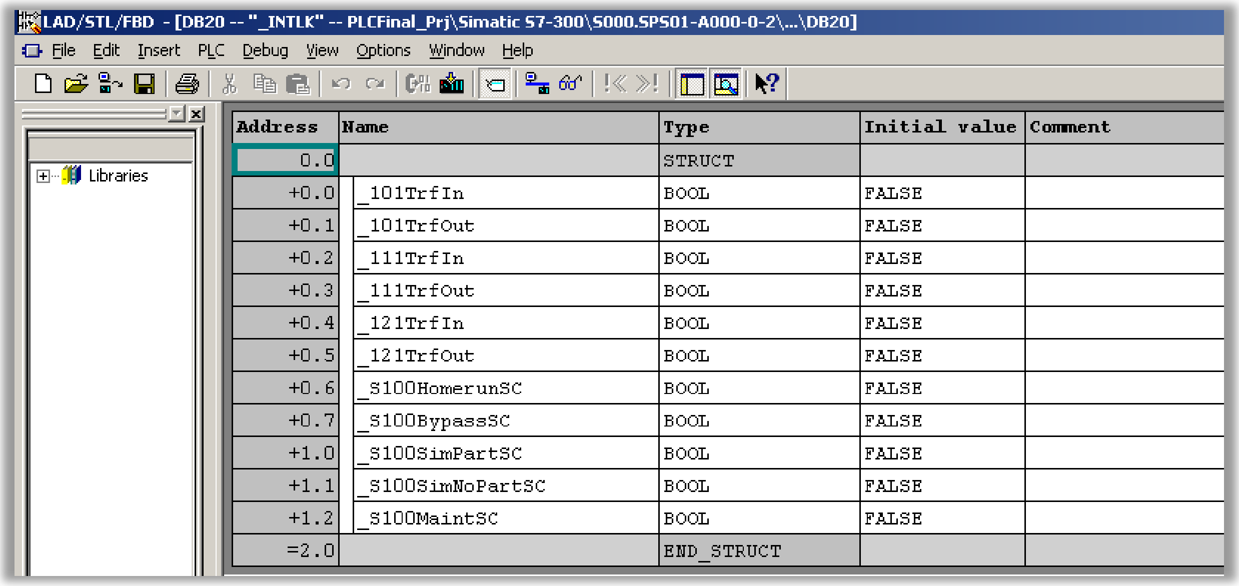
The generated global DBs contain variables needed for the FCs used in this case study project. All variables in the DBs have automatically been given a name that follows the VCC standard. The signals have also been given an initial value according to its data type (see Figure 32).


LAD/STL/FBD - [DB300 -- "_S100" -- PLCFinal_Prj\Simatic S7-300\S000.SPS01-A000-0-2\...\DB300]
File Edit Insert PLC Debug View Options Window Help
Libraries
Progra...
Ca
1: Error
2: Info
3: Cross-references
4: Address info.
5: Modify
Press F1 to get Help.

Address	Name	Type	Initial value
0.0		STRUCT	
+0.0	AlarmABCondition	BOOL	FALSE
+0.1	AlarmA	BOOL	FALSE
+0.2	AlarmB	BOOL	FALSE
+0.3	AlarmC	BOOL	FALSE
+0.4	AlarmD	BOOL	FALSE
+2.0	H_IAnim	WORD	W#16#0
+4.0	HomePosition	BOOL	FALSE
+4.1	_101TrfInSetBegin	BOOL	FALSE
+4.2	_101TrfInSetOutResIlkOk	BOOL	FALSE
+4.3	_101TrfInSetRdyDlv	BOOL	FALSE
+4.4	_101TrfInSetTM	BOOL	FALSE
+4.5	_101TrfOutSetBegin	BOOL	FALSE
+4.6	_101TrfOutSetInResIlkOk	BOOL	FALSE
+4.7	_101TrfOutSetRdyRcv	BOOL	FALSE
+5.0	_101TrfOutSetArrived	BOOL	FALSE
+5.1	_101TrfInResBegin	BOOL	FALSE
+5.2	_101TrfInResRdyDlv	BOOL	FALSE
+5.3	_101TrfInResTM	BOOL	FALSE
+5.4	_101TrfOutResBegin	BOOL	FALSE
+5.5	_101TrfOutResRdyRcv	BOOL	FALSE
+5.6	_101TrfOutResArrived	BOOL	FALSE
+5.7	_101ManInfeedAllowed	BOOL	FALSE
+6.0	_101ReleaseBrake	BOOL	FALSE
+6.1	_101FBFCContactor	BOOL	FALSE
+6.2	_101EnableFCReset	BOOL	FALSE
+6.3	_101EnableCheckSpeedOK	BOOL	FALSE
+6.4	_101FCAlarm	BOOL	FALSE
+6.5	_101LocalPBSet	BOOL	FALSE
+6.6	_101LocalPBRes	BOOL	FALSE
+6.7	_101EnAlmHW	BOOL	FALSE
+7.0	_101EnAlmTransfer	BOOL	FALSE
+8.0	_101ActiveMode	WORD	W#16#0
+10.0	_101GeneralFunc	WORD	W#16#0
+12.0	_101ProductionMode	WORD	W#16#0
+14.0	_101ConfigDWord	DWORD	DW#16#0
+18.0	_101Active	BOOL	FALSE
+18.1	_101RelayReleaseBrake	BOOL	FALSE
+18.2	_101FCContactor	BOOL	FALSE
+18.3	_101TrfInSetRdyRcv	BOOL	FALSE
+18.4	_101TrfInSetArrived	BOOL	FALSE
+18.5	_101TrfInSetActive	BOOL	FALSE
+18.6	_101TrfInSetFinished	BOOL	FALSE
+18.7	_101TrfOutSetRdyDlv	BOOL	FALSE

Figure 32 - The DB for the ModeZone

DB20 (Figure 33) which contains the variables used for interlock calculations is also generated and contain variables for all the added devices that might need it. The Conveyor and ModeZone are the only objects where default interlocks have been tested in this case study.

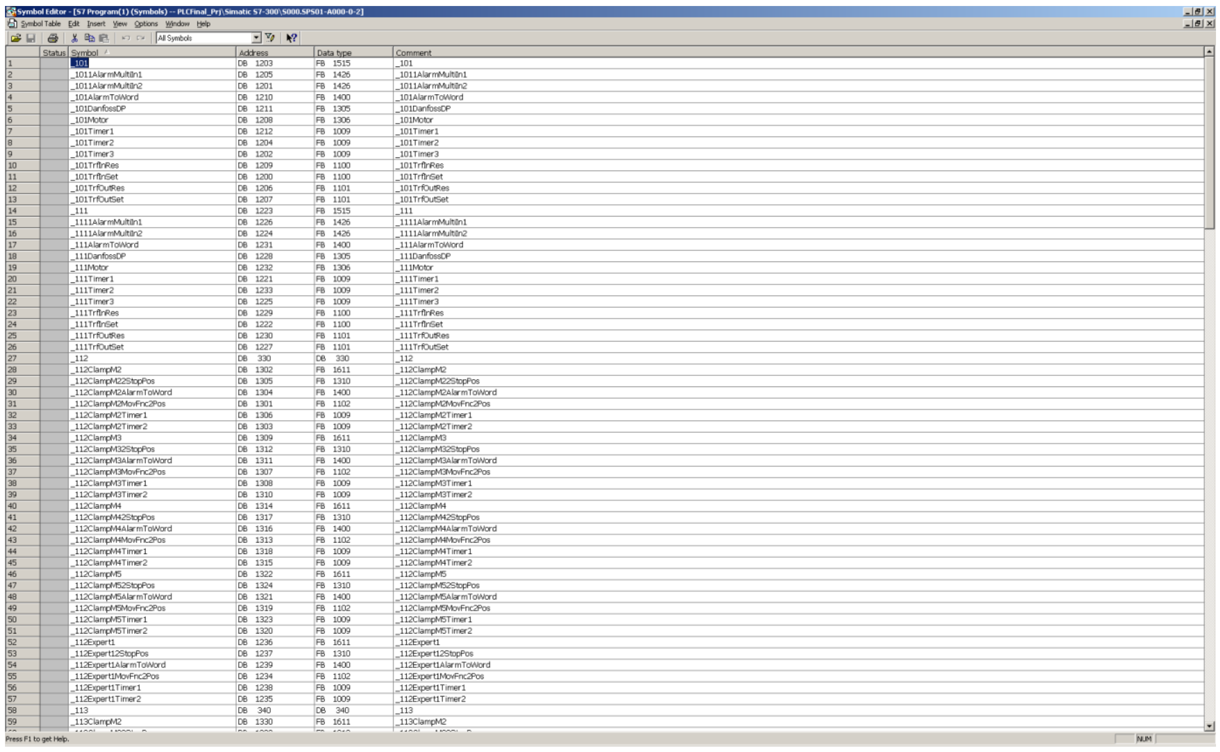

LAD/STL/FBD - [DB20 -- "_INTLK" -- PLCFinal_Prj\Simatic S7-300\S000.SPS01-A000-0-2\...\DB20]
File Edit Insert PLC Debug View Options Window Help
Libraries

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	_101TrfIn	BOOL	FALSE	
+0.1	_101TrfOut	BOOL	FALSE	
+0.2	_111TrfIn	BOOL	FALSE	
+0.3	_111TrfOut	BOOL	FALSE	
+0.4	_121TrfIn	BOOL	FALSE	
+0.5	_121TrfOut	BOOL	FALSE	
+0.6	_S100HomerunSC	BOOL	FALSE	
+0.7	_S100BypassSC	BOOL	FALSE	
+1.0	_S100SimPartSC	BOOL	FALSE	
+1.1	_S100SimNoPartSC	BOOL	FALSE	
+1.2	_S100MaintSC	BOOL	FALSE	
=2.0		END_STRUCT		

Figure 33 - Variables in DB20

7.1.3 Generated symbol table

The generated symbol table contains the addresses for all FCs, FBs and DBs (see Figure 34). Sensors connected to the conveyor are also generated with the right names and addresses. In the symbol table some static values such as TRUE and FALSE are declared. These values are commonly used by VCC in their function blocks to send in a true or false value to their calculations inside the networks. The reason for having TRUE and FALSE symbols for this is because of STEP 7 which does not allow the user to send in values directly into a network calculation.

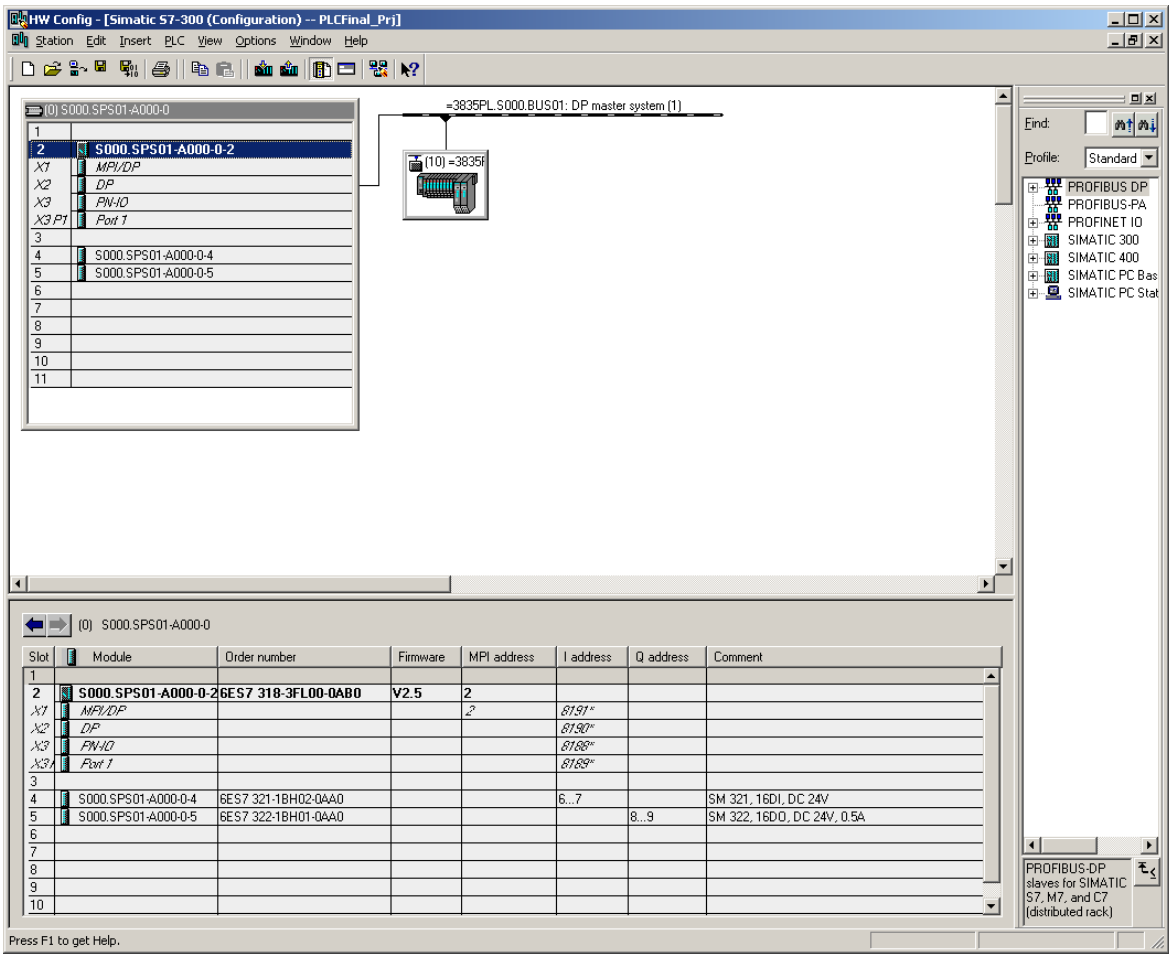

Symbol Editor - [S7 Program(1) (Symbols) -- PLCFinal_Prj\Simatic S7-300\S000.SPS01-A000-0-2]
Symbol Table Edit Insert View Options Window Help
All Symbols
Press F1 to get Help.
NUM

	Status	Symbol	Address	Data type	Comment
1		_101	DB 1203	FB 1515	_101
2		_1011AlarmMultiIn1	DB 1205	FB 1426	_1011AlarmMultiIn1
3		_1011AlarmMultiIn2	DB 1201	FB 1426	_1011AlarmMultiIn2
4		_101AlarmToWord	DB 1210	FB 1400	_101AlarmToWord
5		_101DanfossDP	DB 1211	FB 1305	_101DanfossDP
6		_101Motor	DB 1208	FB 1306	_101Motor
7		_101Timer1	DB 1212	FB 1009	_101Timer1
8		_101Timer2	DB 1204	FB 1009	_101Timer2
9		_101Timer3	DB 1202	FB 1009	_101Timer3
10		_101TrfInRes	DB 1209	FB 1100	_101TrfInRes
11		_101TrfInSet	DB 1200	FB 1100	_101TrfInSet
12		_101TrfOutRes	DB 1206	FB 1101	_101TrfOutRes
13		_101TrfOutSet	DB 1207	FB 1101	_101TrfOutSet
14		_111	DB 1223	FB 1515	_111
15		_1111AlarmMultiIn1	DB 1226	FB 1426	_1111AlarmMultiIn1
16		_1111AlarmMultiIn2	DB 1224	FB 1426	_1111AlarmMultiIn2
17		_111AlarmToWord	DB 1231	FB 1400	_111AlarmToWord
18		_111DanfossDP	DB 1228	FB 1305	_111DanfossDP
19		_111Motor	DB 1232	FB 1306	_111Motor
20		_111Timer1	DB 1221	FB 1009	_111Timer1
21		_111Timer2	DB 1233	FB 1009	_111Timer2
22		_111Timer3	DB 1225	FB 1009	_111Timer3
23		_111TrfInRes	DB 1229	FB 1100	_111TrfInRes
24		_111TrfInSet	DB 1222	FB 1100	_111TrfInSet
25		_111TrfOutRes	DB 1230	FB 1101	_111TrfOutRes
26		_111TrfOutSet	DB 1227	FB 1101	_111TrfOutSet
27		_112	DB 330	DB 330	_112
28		_112ClampM2	DB 1302	FB 1611	_112ClampM2
29		_112ClampM22StopPos	DB 1305	FB 1310	_112ClampM22StopPos
30		_112ClampM2AlarmToWord	DB 1304	FB 1400	_112ClampM2AlarmToWord
31		_112ClampM2MovFnc2Pos	DB 1301	FB 1102	_112ClampM2MovFnc2Pos
32		_112ClampM2Timer1	DB 1306	FB 1009	_112ClampM2Timer1
33		_112ClampM2Timer2	DB 1303	FB 1009	_112ClampM2Timer2
34		_112ClampM3	DB 1309	FB 1611	_112ClampM3
35		_112ClampM32StopPos	DB 1312	FB 1310	_112ClampM32StopPos
36		_112ClampM3AlarmToWord	DB 1311	FB 1400	_112ClampM3AlarmToWord
37		_112ClampM3MovFnc2Pos	DB 1307	FB 1102	_112ClampM3MovFnc2Pos
38		_112ClampM3Timer1	DB 1308	FB 1009	_112ClampM3Timer1
39		_112ClampM3Timer2	DB 1310	FB 1009	_112ClampM3Timer2
40		_112ClampM4	DB 1314	FB 1611	_112ClampM4
41		_112ClampM42StopPos	DB 1317	FB 1310	_112ClampM42StopPos
42		_112ClampM4AlarmToWord	DB 1316	FB 1400	_112ClampM4AlarmToWord
43		_112ClampM4MovFnc2Pos	DB 1313	FB 1102	_112ClampM4MovFnc2Pos
44		_112ClampM4Timer1	DB 1318	FB 1009	_112ClampM4Timer1
45		_112ClampM4Timer2	DB 1315	FB 1009	_112ClampM4Timer2
46		_112ClampM5	DB 1322	FB 1611	_112ClampM5
47		_112ClampM52StopPos	DB 1324	FB 1310	_112ClampM52StopPos
48		_112ClampM5AlarmToWord	DB 1321	FB 1400	_112ClampM5AlarmToWord
49		_112ClampM5MovFnc2Pos	DB 1319	FB 1102	_112ClampM5MovFnc2Pos
50		_112ClampM5Timer1	DB 1323	FB 1009	_112ClampM5Timer1
51		_112ClampM5Timer2	DB 1320	FB 1009	_112ClampM5Timer2
52		_112Expert1	DB 1236	FB 1611	_112Expert1
53		_112Expert12StopPos	DB 1237	FB 1310	_112Expert12StopPos
54		_112Expert1AlarmToWord	DB 1239	FB 1400	_112Expert1AlarmToWord
55		_112Expert1MovFnc2Pos	DB 1234	FB 1102	_112Expert1MovFnc2Pos
56		_112Expert1Timer1	DB 1238	FB 1009	_112Expert1Timer1
57		_112Expert1Timer2	DB 1235	FB 1009	_112Expert1Timer2
58		_113	DB 340	DB 340	_113
59		_113ClampM2	DB 1330	FB 1611	_113ClampM2

Figure 34 - Generated symbol table

7.1.4 Generated Hardware

The exported S7 project contains the right hardware configuration of the PLC and one terminal box. The names for hardware are based on the objects search path in Automation Designer and therefore not correctly assigned according to VCCs standards (see Figure 35). However the sensors on the conveyor are connected to the terminal box and are assigned a valid address. The BUS type and its addresses are also set to the right values in the exported project.


HW Config - [Simatic S7-300 (Configuration) -- PLCFinal_Prj]
Station Edit Insert PLC View Options Window Help
(0) S000.SPS01-A000-0
=3835PL.S000.BUS01: DP master system (1)
(10) =3835P
Find:
Profile: Standard
PROFIBUS DP
PROFIBUS-PA
PROFINET IO
SIMATIC 300
SIMATIC 400
SIMATIC PC Bas
SIMATIC PC Stat
PROFIBUS-DP slaves for SIMATIC S7, M7, and C7 (distributed rack)
Press F1 to get Help.

(0) S000.SPS01-A000-0

1	
2	S000.SPS01-A000-0-2
X1	MPI/DP
X2	DP
X3	PN-IO
X3 P1	Port 1
3	
4	S000.SPS01-A000-0-4
5	S000.SPS01-A000-0-5
6	
7	
8	
9	
10	
11	

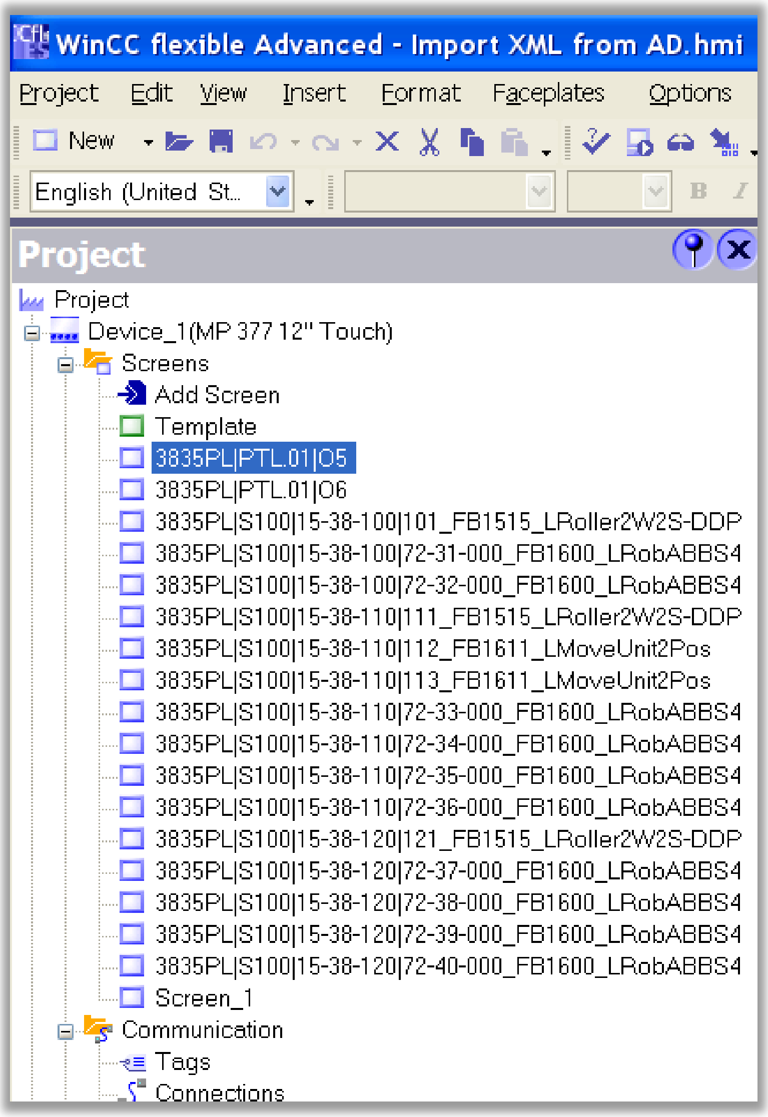
(0) S000.SPS01-A000-0

Slot	Module	Order number	Firmware	MPI address	I address	Q address	Comment
1							
2	S000.SPS01-A000-0-2	6ES7 318-3FL00-0AB0	V2.5	2			
X1	MPI/DP			2	8191*		
X2	DP				8190*		
X3	PN-IO				8188*		
X3	Port 1				8189*		
3							
4	S000.SPS01-A000-0-4	6ES7 321-1BH02-0AA0			6...7		SM 321, 16DI, DC 24V
5	S000.SPS01-A000-0-5	6ES7 322-1BH01-0AA0				8...9	SM 322, 16DO, DC 24V, 0.5A
6							
7							
8							
9							
10							

Figure 35 - Hardware configuration for the generated project.

7.2 WinCC flexible screen generation

The export from Automation Designer includes screens for the highest level (OB1), the mode zone and detail screens for each included object such as robots, conveyors and turntables. A tag list containing all tags used in the screens is also included in the export. The screen names (see Figure 36) are generated by Automation Designer and are based on the search paths in Automation Designer for each object. The screen names do not correspond to the VCC standard.


WinCC flexible Advanced - Import XML from AD.hmi

Project
- Device_1(MP 377 12'' Touch)
  - Screens
    - Add Screen
    - Template
    - 3835PL|PTL.01|O5
    - 3835PL|PTL.01|O6
    - 3835PL|S100|15-38-100|101_FB1515_LRoller2W2S-DDP
    - 3835PL|S100|15-38-100|72-31-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-100|72-32-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-110|111_FB1515_LRoller2W2S-DDP
    - 3835PL|S100|15-38-110|112_FB1611_LMoveUnit2Pos
    - 3835PL|S100|15-38-110|113_FB1611_LMoveUnit2Pos
    - 3835PL|S100|15-38-110|72-33-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-110|72-34-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-110|72-35-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-110|72-36-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-120|121_FB1515_LRoller2W2S-DDP
    - 3835PL|S100|15-38-120|72-37-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-120|72-38-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-120|72-39-000_FB1600_LRobABBS4
    - 3835PL|S100|15-38-120|72-40-000_FB1600_LRobABBS4
    - Screen_1
  - Communication
    - Tags
    - Connections

Figure 36 - List of screens in WinCC flexible generated from Automation Designer

The screens for OB1 and the mode zone are based on the 2D layout defined in Automation Designer (see Figure 25). The OB1 screen (Figure 37) includes graphical representations of all objects and an icon for the mode zone. The mode zone screen contains icons for all objects (Figure 38). All icons have H_IAnim tags connected to them pointing to correct addresses in the DBs. All icons are created according to the VCC standard in chapter 4.1.3.

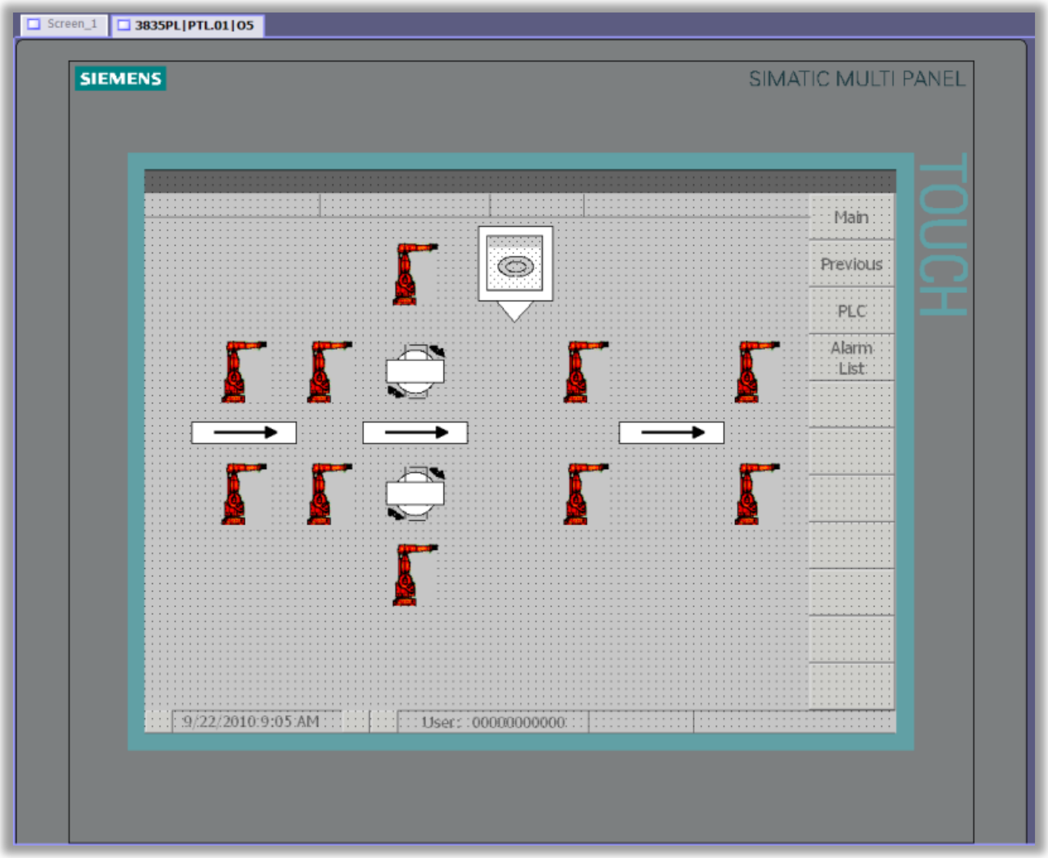


Figure 37 - Generated screen forOB1

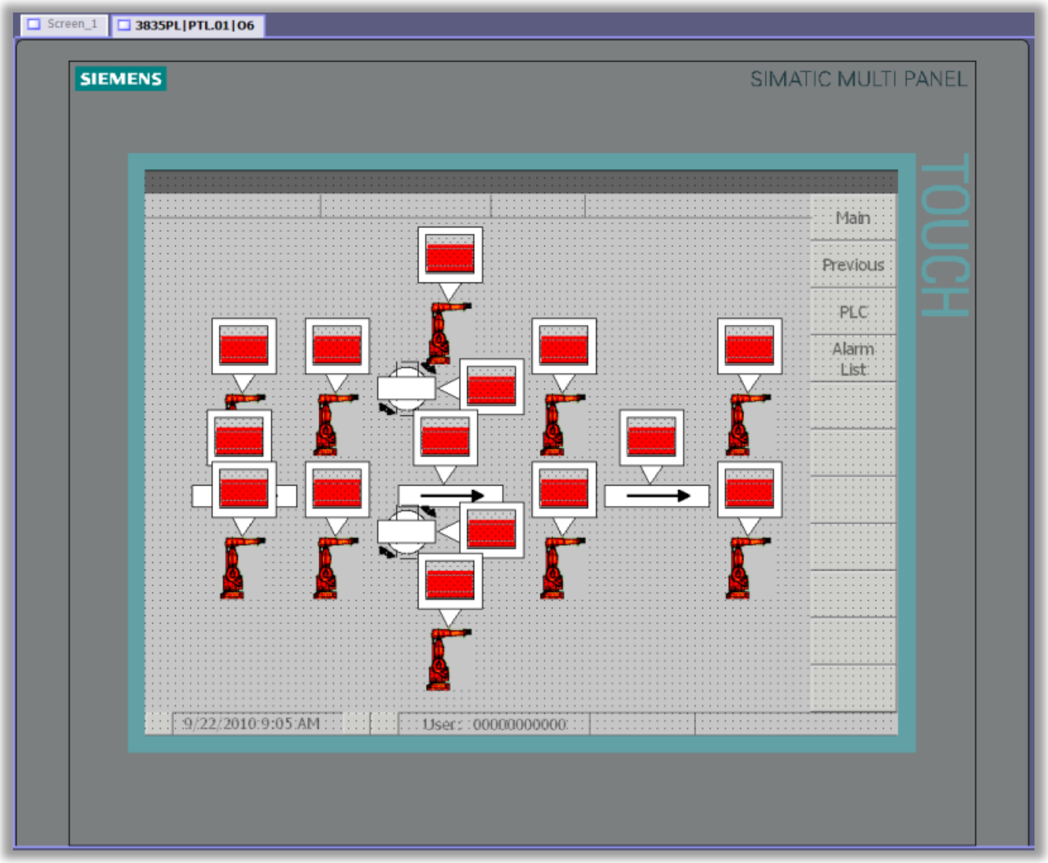


Figure 38 - Generated screen for mode zone

The detail screens (Figure 39) has got H_Anim tags connected to them as specified in the VCC standard. Detail screens are generated in the same way as they were imported to Automation Designer from WinCC flexible, as described in chapter 6.1.2.2. The information generated by Automation Designer concerning the detail screens is the tags connected to the screens.

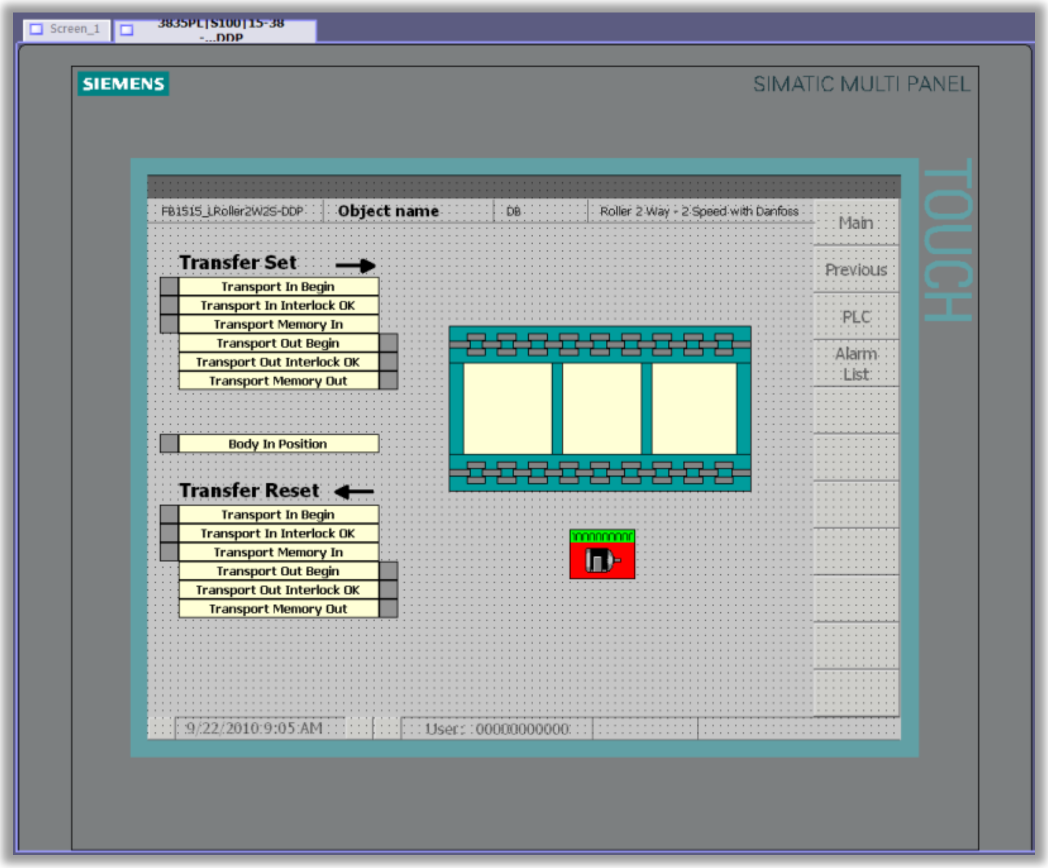
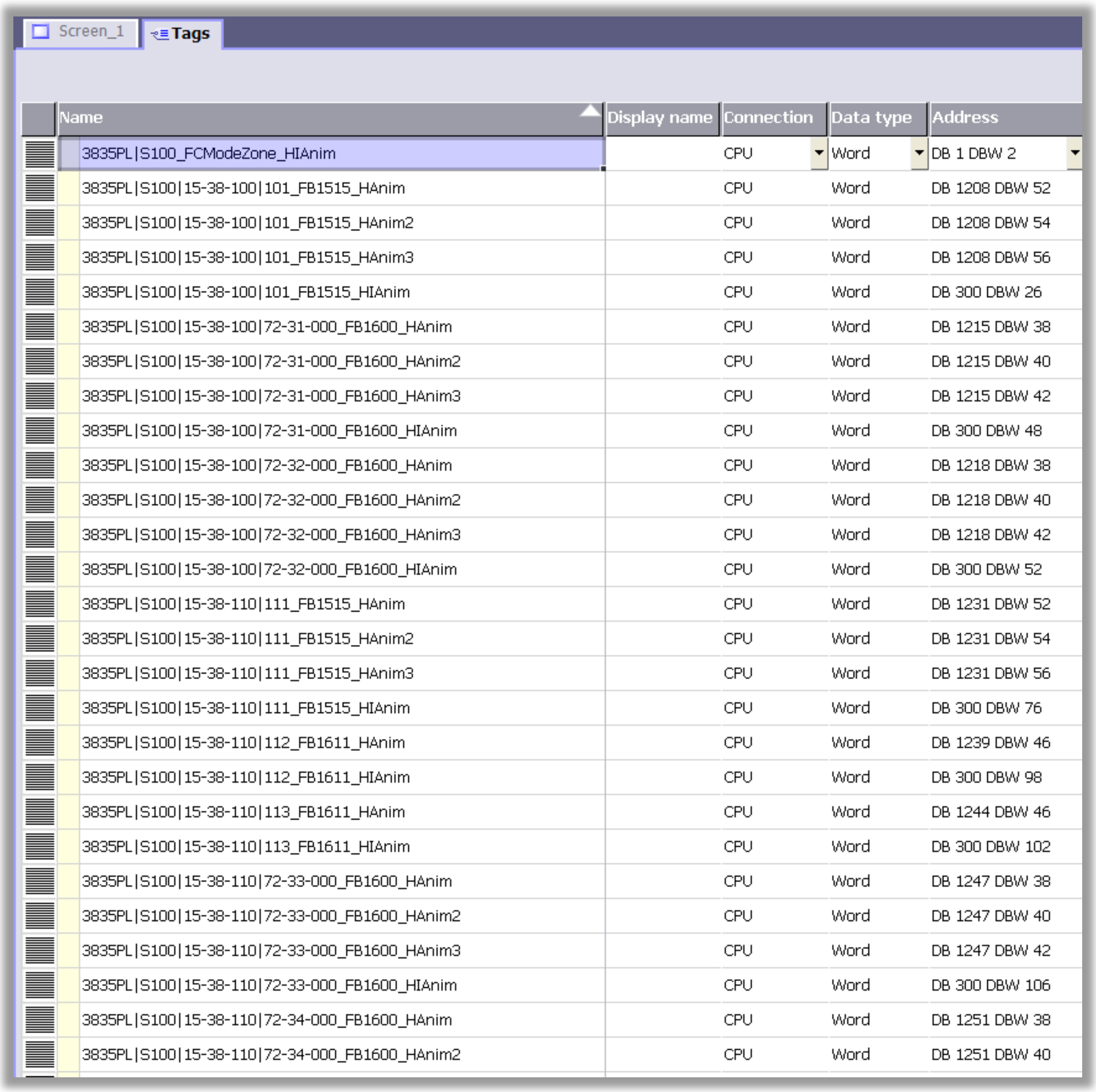


Figure 39 - Detail screen for conveyer FB1515

The generated tag list (Figure 40) contains tags with correct addresses to the already generated PLC project. The names for these tags are generated by Automation Designer and do not follow the VCC standard. It is not possible to change the names in Automation Designer before the export to WinCC flexible is done.

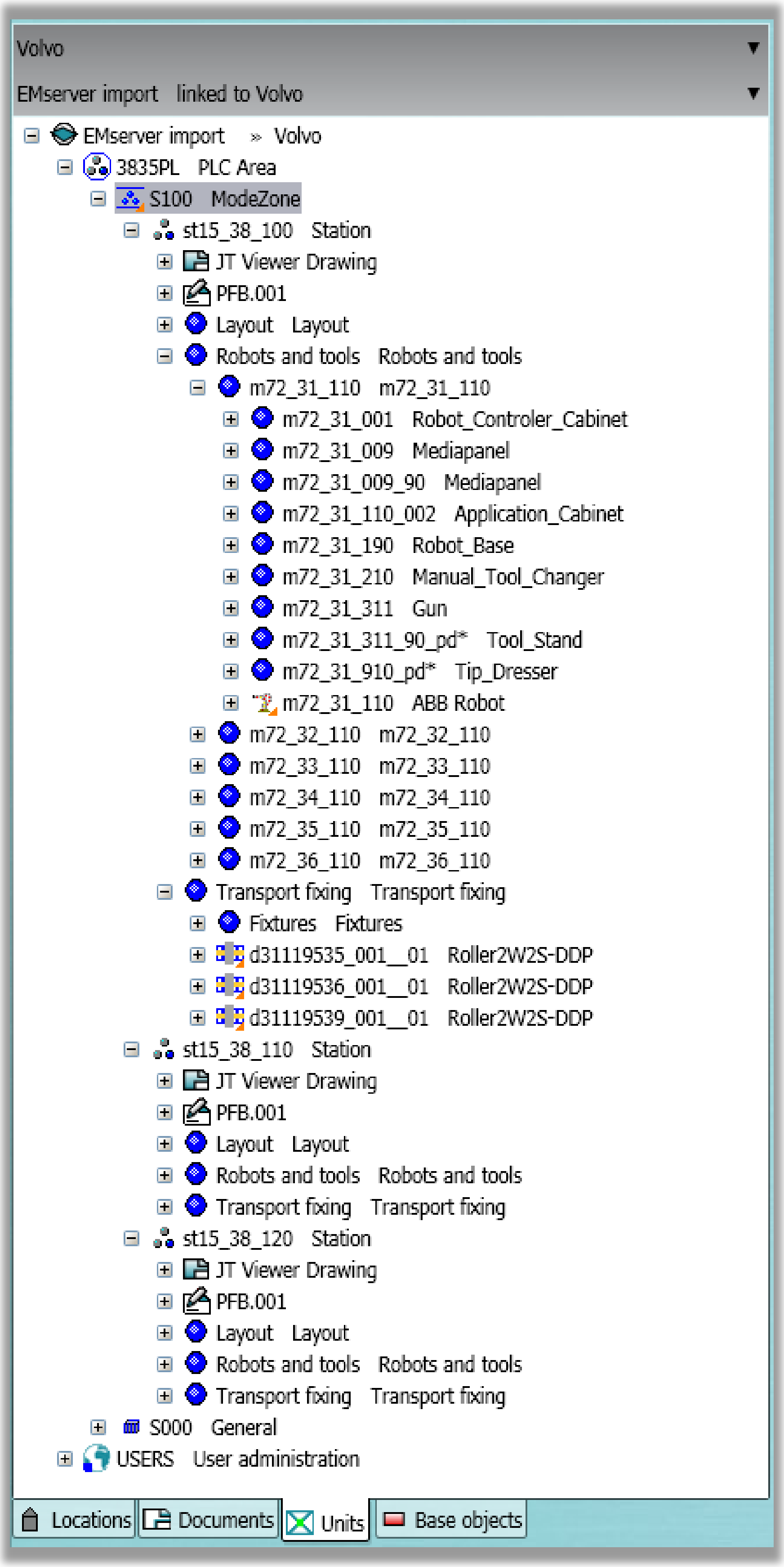


Name	Display name	Connection	Data type	Address
3835PL|S100_FCModeZone_HIAnim		CPU	Word	DB 1 DBW 2
3835PL|S100|15-38-100|101_FB1515_HAnim		CPU	Word	DB 1208 DBW 52
3835PL|S100|15-38-100|101_FB1515_HAnim2		CPU	Word	DB 1208 DBW 54
3835PL|S100|15-38-100|101_FB1515_HAnim3		CPU	Word	DB 1208 DBW 56
3835PL|S100|15-38-100|101_FB1515_HIAnim		CPU	Word	DB 300 DBW 26
3835PL|S100|15-38-100|72-31-000_FB1600_HAnim		CPU	Word	DB 1215 DBW 38
3835PL|S100|15-38-100|72-31-000_FB1600_HAnim2		CPU	Word	DB 1215 DBW 40
3835PL|S100|15-38-100|72-31-000_FB1600_HAnim3		CPU	Word	DB 1215 DBW 42
3835PL|S100|15-38-100|72-31-000_FB1600_HIAnim		CPU	Word	DB 300 DBW 48
3835PL|S100|15-38-100|72-32-000_FB1600_HAnim		CPU	Word	DB 1218 DBW 38
3835PL|S100|15-38-100|72-32-000_FB1600_HAnim2		CPU	Word	DB 1218 DBW 40
3835PL|S100|15-38-100|72-32-000_FB1600_HAnim3		CPU	Word	DB 1218 DBW 42
3835PL|S100|15-38-100|72-32-000_FB1600_HIAnim		CPU	Word	DB 300 DBW 52
3835PL|S100|15-38-110|111_FB1515_HAnim		CPU	Word	DB 1231 DBW 52
3835PL|S100|15-38-110|111_FB1515_HAnim2		CPU	Word	DB 1231 DBW 54
3835PL|S100|15-38-110|111_FB1515_HAnim3		CPU	Word	DB 1231 DBW 56
3835PL|S100|15-38-110|111_FB1515_HIAnim		CPU	Word	DB 300 DBW 76
3835PL|S100|15-38-110|112_FB1611_HAnim		CPU	Word	DB 1239 DBW 46
3835PL|S100|15-38-110|112_FB1611_HIAnim		CPU	Word	DB 300 DBW 98
3835PL|S100|15-38-110|113_FB1611_HAnim		CPU	Word	DB 1244 DBW 46
3835PL|S100|15-38-110|113_FB1611_HIAnim		CPU	Word	DB 300 DBW 102
3835PL|S100|15-38-110|72-33-000_FB1600_HAnim		CPU	Word	DB 1247 DBW 38
3835PL|S100|15-38-110|72-33-000_FB1600_HAnim2		CPU	Word	DB 1247 DBW 40
3835PL|S100|15-38-110|72-33-000_FB1600_HAnim3		CPU	Word	DB 1247 DBW 42
3835PL|S100|15-38-110|72-33-000_FB1600_HIAnim		CPU	Word	DB 300 DBW 106
3835PL|S100|15-38-110|72-34-000_FB1600_HAnim		CPU	Word	DB 1251 DBW 38
3835PL|S100|15-38-110|72-34-000_FB1600_HAnim2		CPU	Word	DB 1251 DBW 40

Figure 40 - Tags with addresses generated from Automation Designer

7.3 EMserver import

Figure 41 shows the imported structure from Process Simulate. The highest levels of the structure are the stations; therefore the importation is done from the ModeZone object. The imported structure contains many objects that are not defined in Automation Designer. These objects are set as a default object and have got blue dots as icons (see Figure 41). The robots and conveyors are set as their respective template. The naming scripts associated with the templates are not executed instead the robots and conveyors keeps their names from Process Simulate. The structure from Process Simulate provided for this case study includes several extra layers and several instances of the same conveyors. This makes it incompatible with the structure defined in the base project of Automation Designer.



```
Volvo
EMserver import   linked to Volvo

⊟ EMserver import  » Volvo
  ⊟ 3835PL   PLC Area
    ⊟ S100   ModeZone
      ⊟ st15_38_100   Station
        ⊞ JT Viewer Drawing
        ⊞ PFB.001
        ⊞ Layout   Layout
        ⊟ Robots and tools   Robots and tools
          ⊟ m72_31_110   m72_31_110
            ⊞ m72_31_001   Robot_Controler_Cabinet
            ⊞ m72_31_009   Mediapanel
            ⊞ m72_31_009_90   Mediapanel
            ⊞ m72_31_110_002   Application_Cabinet
            ⊞ m72_31_190   Robot_Base
            ⊞ m72_31_210   Manual_Tool_Changer
            ⊞ m72_31_311   Gun
            ⊞ m72_31_311_90_pd*   Tool_Stand
            ⊞ m72_31_910_pd*   Tip_Dresser
            ⊞ m72_31_110   ABB Robot
          ⊞ m72_32_110   m72_32_110
          ⊞ m72_33_110   m72_33_110
          ⊞ m72_34_110   m72_34_110
          ⊞ m72_35_110   m72_35_110
          ⊞ m72_36_110   m72_36_110
        ⊟ Transport fixing   Transport fixing
          ⊞ Fixtures   Fixtures
          ⊞ d31119535_001__01   Roller2W2S-DDP
          ⊞ d31119536_001__01   Roller2W2S-DDP
          ⊞ d31119539_001__01   Roller2W2S-DDP
      ⊟ st15_38_110   Station
        ⊞ JT Viewer Drawing
        ⊞ PFB.001
        ⊞ Layout   Layout
        ⊞ Robots and tools   Robots and tools
        ⊞ Transport fixing   Transport fixing
      ⊟ st15_38_120   Station
        ⊞ JT Viewer Drawing
        ⊞ PFB.001
        ⊞ Layout   Layout
        ⊞ Robots and tools   Robots and tools
        ⊞ Transport fixing   Transport fixing
    ⊞ S000   General
  ⊞ USERS   User administration

Locations | Documents | Units | Base objects
```

Figure 41 - Tree structure from XML file imported from Process Simulate

8 Discussion

The main goal with this master thesis was to test how Automation Designer can be used to generate PLC code according to the VCC standard. The three stations 15-38-100, 110, 120 at VCC in Torslanda were chosen to be used as a case study. The three stations are controlled by the same PLC and contained a variety of different equipment which made them suitable for the study. The first task was to build base objects and templates for every object in the stations that is controlled by the PLC; however the time limit did not make this possible. Only the main equipment such as the robots, conveyors and turntables was created. The reason for this was that Automation Designer has been in the development phase during the entire time of this case study and a lot of work has been to deal with bugs and other problems in the software. Nevertheless the created objects were enough to test the possibility in Automation Designer to create STEP 7 code in accordance with the VCC standard.

The results of the generated PLC code show that the VCC PLC structure is possible to generate by the help of Automation Designer, but the code still have to be complemented with logical calculations such as interlocks afterwards in Simatic Manager. It is hard to validate and verify the generated PLC code because the stations are using Mitsubishi PLCs instead of Siemens today. These are programmed in a different way which makes it hard to compare the results. The standards for programming the different types of PLCs are also different within VCC.

The generated PLC project can work as a better base project than the one used by VCC today. Advantages with the generated PLC project in comparison with the start PLC project VCC uses today is that the generated program only includes the function and data blocks used in the project. No unnecessary information is stored in the project. Another advantage is that the majority of the needed variables in the global DBs are defined and every DB and FC has been set with the right name and number. The most important advantage is that the VCC PLC structure is set in the generated PLC project which decreases the risk of getting an inaccurate structure in the finished PLC program.

The VCC function block library was successfully integrated into Automation Designer. The library function blocks used in the generated PLC project is collected from the library during the export of the PLC code to STEP 7. This makes is possible to have only one function block library on one place from where Automation Designer can collect the required blocks each time it generates a PLC project.

A secondary goal after generating the PLC code was to generate HMI screens for the stations. The feature to do this inside Automation Designer was not fully functional until late in the case study. Therefore only some of the H_Anim and H_IAnim variables from the data blocks are connected in the used screens. The assignment for the HMI generation was the same as for the PLC code; to test if the result concedes with the VCC standard. Screens generated for OB1 and the mode zone shows that it is possible to get the layout desired by VCC; however the names on the tags connected to the screens are not named according to the Standard. The same goes for the names on the screens which have a search path from Automation Designer as a name. It is possible to change the names manually in WinCC flexible and still maintain the address to the data blocks. This problem is known by Siemens and they are going to take a look at it and hopefully change it in a future service pack.

As this master thesis is about how to reuse information one of the tasks was to have a look upon how information from earlier simulation studies, in this case from Process Simulate, could be reused. As seen in the results it is possible to import an XML file from Process Simulate into Automation

Designer and get the tree structure with the same names on the objects as they had in Process Simulate. To make this possible a connection between the objects in Process Simulate and the templates in Automation Designer must be set up. The result in this case study was a tree structure which did not fit very well with the way of working in Automation Designer. One of the reasons for the bad structure was because the simulation study was old and performed in the software Robcad. Another reason for this was that the XML import feature was presented late in the project and the focus was to build up a structure with the VCC PLC standards as reference.

Working with Automation Designer has been tricky sometimes because, as written earlier, the software was under development during this case study. A lot of time when a problem have occurred the reason has been software related instead of errors from the user. This has taken much time that could have been spent on developing and optimizing templates and base objects. The final version of Automation Designer 9.1 is due for release at the same time as this report is published. Now there are a lot of smart functions included in the software which we would have had use for when creating the base project inside the software. An example of a new feature is the possibility to quickly import source code from function blocks by the push of a button. In our case we have had to do a couple of steps inside SIMATIC Manager and then copy and paste the information needed. Nevertheless the software is easier to work with since the last service pack (service pack 100) where a lot of the bugs are solved.

9 Conclusion

This master thesis has shown that it is possible to use Automation Designer for generating PLC code according to the VCC standards. The generated PLC program will not be completely functional and has to be completed before it can be used in a real production line. However, the workload that has to be performed on the generated STEP 7 project from Automation Designer in comparison to the standard VCC start project is decreased significantly. Generating PLC projects with Automation Designer will also ensure they follow the VCC PLC structure. A function block library can also be integrated with Automation Designer from which the software can collect the necessary library blocks and no unnecessary blocks will be included in the generated PLC project.

HMI screens for WinCC flexible are possible to create with the help of Automation Designer and tags can be connected to a PLC project and get the right address. However, the names of the generated tags and screens cannot be generated according to the VCC standard.

It is possible to transfer information from Process Simulate into Automation Designer by the help of an XML file. To have any advantage of importing a XML file into the software the naming standard of objects must be the same in both software's. Otherwise it is hard to reuse the information in a useful way and the benefits of earning working hours will be lost on changing names inside Automation Designer.

Our conclusion about Automation Designer is that it is developed to decrease the commissioning hours when creating new lines. For updating lines already implemented in the factory and where only smaller changes has to be made, there is no benefits to use Automation Designer. The largest earnings of using the software would be in the long run when creating completely new lines.

10 Future recommendations

The first recommendation is that VCC needs to come up with a common naming standard of objects between the simulation and PLC department. This would be a first step to solve the issue of reusing information within the company.

The second recommendation is that VCC should initiate a new master thesis which could continue the work on Automation Designer and investigate the possibility of generating electrical drawings.

11 References

Ashton, A. (2007, July 1). *PLC programming standards*. Retrieved Mars 3, 2010, from SA instrumentation & Control: http://instrumentation.co.za/article.aspx?pklArticleId=4564&pklCategoryId=67

Frey, G., & Litz, L. (2000). Formal methods in PLC programming. *IEEE Conference on System Man and Cybernetics*, (pp. 2431-2436). Nashville.

Güttel, K., Weber, P., & Fay, A. (2008). Automatic generation of PLC code beyond the nominal sequence. *Emerging Technologies and Factory Automation*, (pp. 1277 - 1284). Hamburg.

PLCopen. (2008, December 4). *PLCopen*. Retrieved March 3, 2010, from plcopen.org: http://www.plcopen.org/pages/tc1_standards/

Selander, J. (2010). Control System Expert VCC. (M. Andersson, & E. Helander, Interviewers) Göteborg.

Siemens AG Automation and Drives. (2006, May 2). *Operating Manual for Programming with STEP 7 V5.4.* Retrieved March 5, 2010, from Siemens.com: http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objID=10805383&subtype=133300

Siemens AG Industry Sector. (2008, October 20). *Operating Manual for WinCC flexible 2008 Compact/ Standard/ Advanced.* Retrieved March 2, 2010, from Siemens.com: http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&lang=en&objID=16502685&subtype=133300

Siemens AG Industry Sector. (2010, 06 29). Manual SIMATIC Automation Designer based on Comos. Nürnberg, Germany.

Siemens AG Industry Sector. (2010, February 18). *SIMATIC WinCC flexible Brochure March 2010.* Retrieved March 2, 2010, from Siemens: http://www.automation.siemens.com/salesmaterial-as/brochure_simatic-wincc-flexible_en.pdf

Siemens PLM Software. (2008, January 25). *Process Designer fact sheet.* Retrieved March 4, 2010, from Siemens.com: http://www.plm.automation.siemens.com/en_us/Images/tx%20process%20designer%20fs%20W%205_tcm1023-4941.pdf

Siemens PLM Software. (2008, January 22). *Process Simulate fact sheet.* Retrieved March 3, 2010, from Siemens.com: http://www.plm.automation.siemens.com/en_us/Images/7457_tcm1023-80351.pdf

Smith, F. O. (2009, October). When eBOMs and mBOMs converge. *Control Engineering*, pp. 41-44.

Sundbäck, M. (2010). Simulation Expert VCC. (M. Andersson, & E. Helander, Interviewers) Göteborg.

Torsten Heverhagen, R. H. (n.d.). *Function Blocks*. Retrieved Mars 4, 2010, from Function Blocks: http//www.functionblocks.org/Introduction.html

Vinnova. (2010, January 25). *FFI - Strategic Vehicle Research and Innivation*. Retrieved February 11, 2010, from Vinnova.se: http://www.vinnova.se/en/Activities/Transportation/FFI-English/

Volvo Car Corporation. (2008, November). Programming instructions PLC equipment Simatic S7.

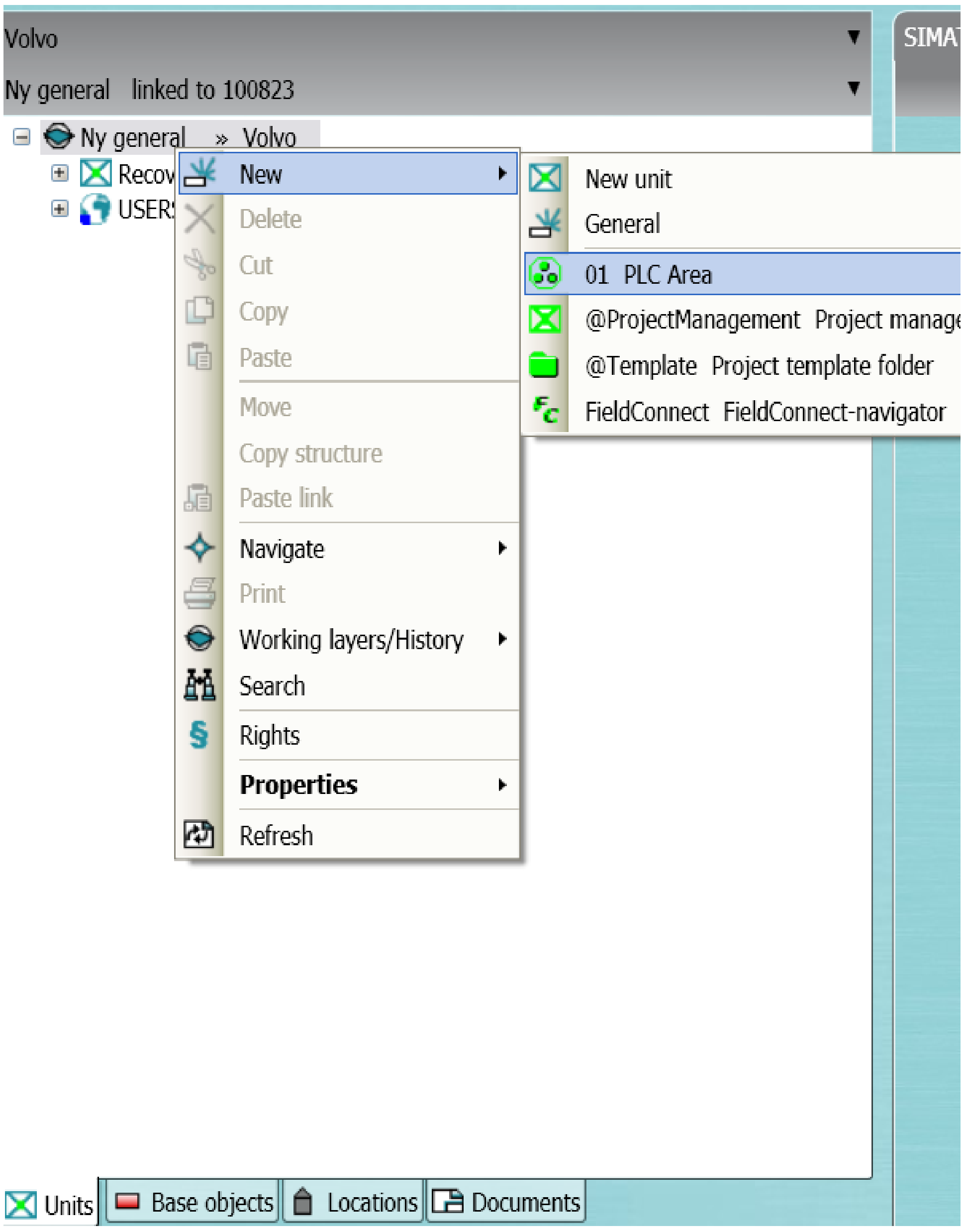
Volvo Car Corporation. (2008, November). Programming instructions PLC systems.

Appendix A

This appendix will describe how to create and configure an engineering project from the database created in this master thesis.

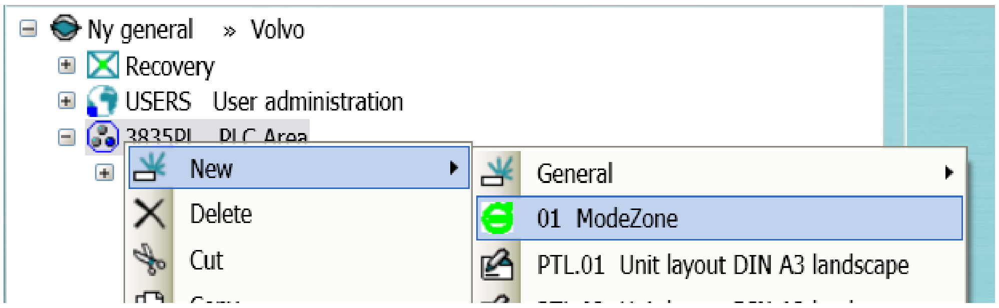
A.1 Adding objects in the navigation tree

1. Right click on the project object, the topmost item in the units tab of the navigation tree, and select New -> PLC Area. All objects controlled by a single PLC will be added under the PLC area later on.

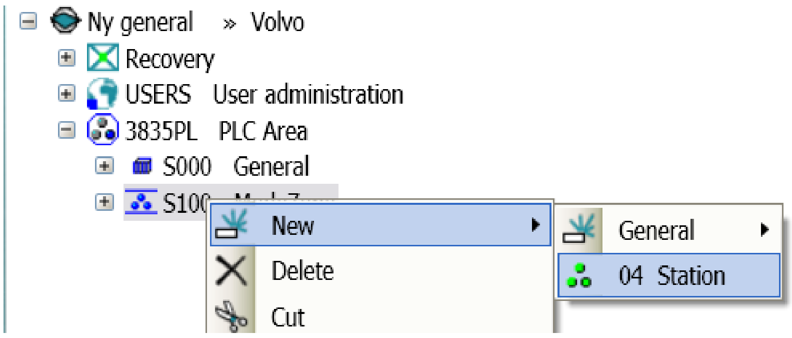


2. Now right click on the PLC Area object and select New-> ModeZone. The mode zone is a crucial part of the VCC PLC structure and it is within the FC for the mode zone most of the

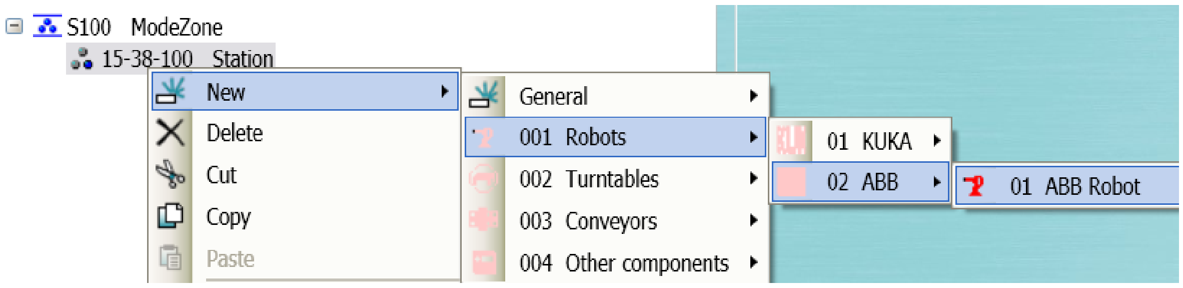
interlock calculations for the equipment will occur. A PLC can control several mode zones.



3. Right click on the ModeZone object and select New -> Station. The stations do not have any impact on the PLC code but the name of some of the objects that will be created below the Station depends on the name of it.



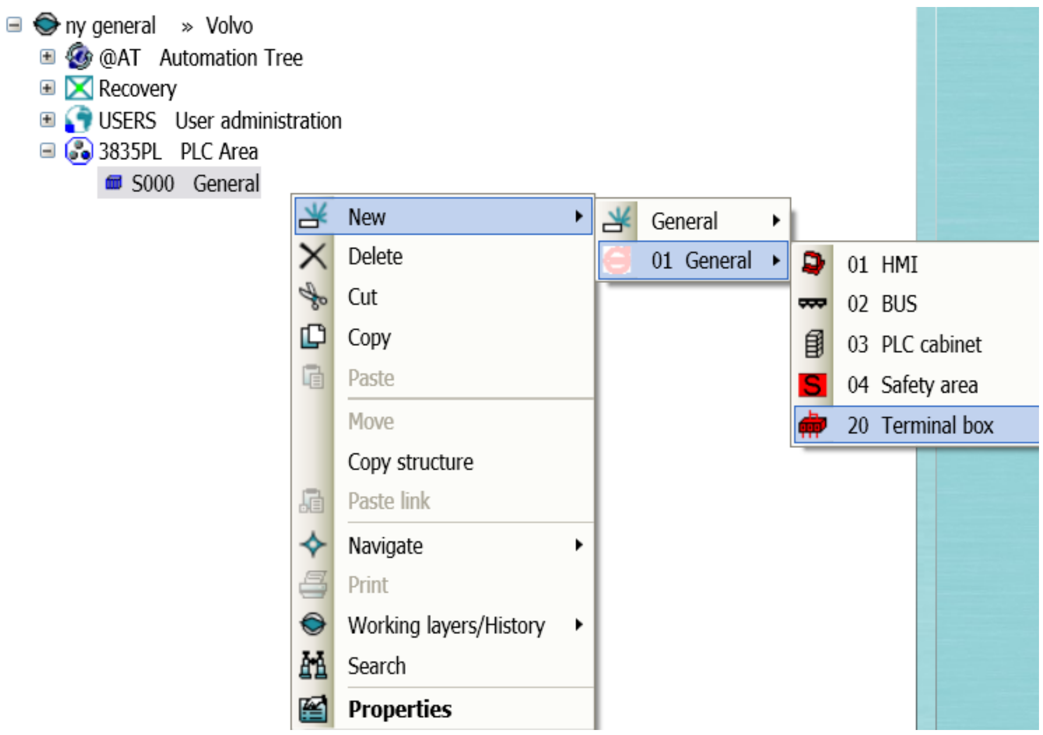
4. The last step is to add the objects that build up the stations. This database contains three different objects that can be added:
   I. First there is an ABB robot. There is a folder for KUKA robots also though it is empty and just there for exemplifying how the structure in a complete database could look like.



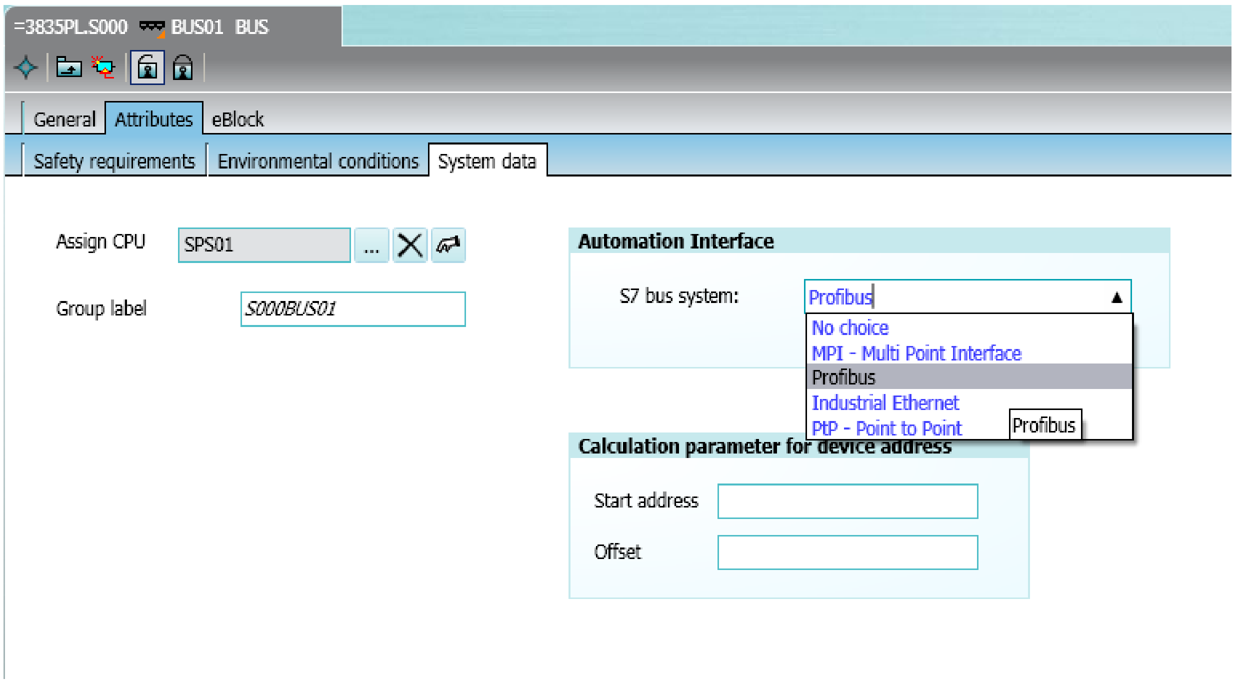
   II. Next is the Turntable. The turntable object consists of a two position turntable plus a variable set of clamps (see chapter A.4).
   III. The last device is a two way two speed roller with Danfoss equipment controlled through Profibus.
5. Add the preferred objects by right clicking on the Station object and select New and then the desired object.

A.2 Configuration of the General folder

1. Right click on the General folder, choose New -> General and add the following items:
   - HMI
   - BUS
   - PLC cabinet
   - Terminal Box



2. Double click on the BUS in the navigation tree to open its properties in the configuration area and select System data in the attributes tab.
3. Select Profibus as S7 bus system and assign a start address (e.g. 1) and a offset (e.g. 10).


=3835PL.S000 BUS01 BUS

General | Attributes | eBlock

Safety requirements | Environmental conditions | System data

Assign CPU: SPS01

Group label: S000BUS01

Automation Interface

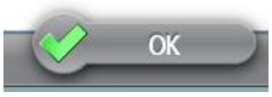
S7 bus system: Profibus

Calculation parameter for device address

Start address:

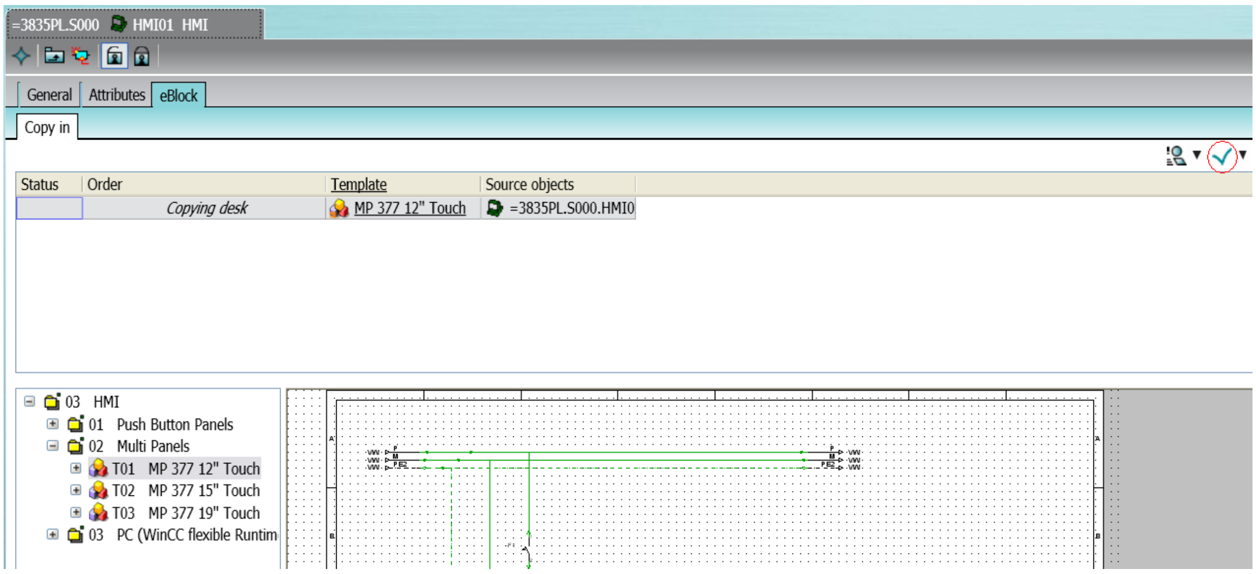
Offset:

4. Press OK to save and close the BUS property window.



5. Double click on the HMI in the navigation tree to open its properties in the configuration area and select the eBlock tab.
6. In the copy in tab for the HMI object there is one eBlock for copying in the hardware specification of the hmi. The default template that is chosen is a 12” Multi Panel. It is

possible to choose a larger multi panel or even a PC though the HMI screen templates in this project only works with multi panels. To execute the eBlock click on the check icon highlighted in the picture below.


=3835PL.S000 HMI01 HMI

General | Attributes | eBlock

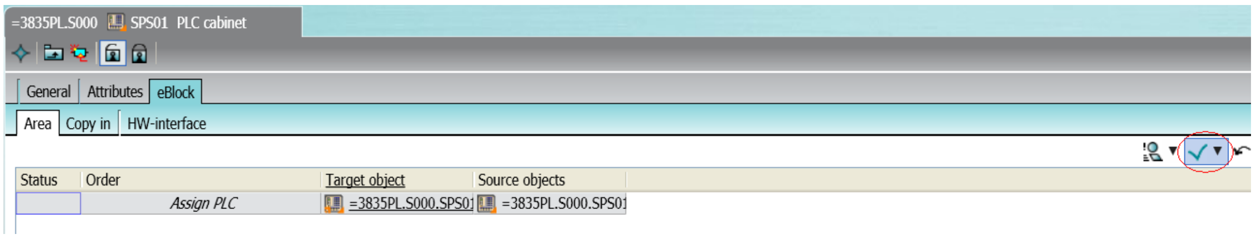
Copy in

Status	Order	Template	Source objects
	Copying desk	MP 377 12" Touch	=3835PL.S000.HMI0

03 HMI
- 01 Push Button Panels
- 02 Multi Panels
  - T01 MP 377 12" Touch
  - T02 MP 377 15" Touch
  - T03 MP 377 19" Touch
- 03 PC (WinCC flexible Runtim

7. Press OK to save and close the HMI property window.

8. Double click on the PLC cabinet in the navigation tree to open its properties in the configuration area and select the eBlock tab.
9. First the “Assign PLC” eBlock needs to be executed in order for some of the Siemens base objects to work properly. Do this by clicking on the execute icon highlighted in the picture below.


=3835PL.S000 SPS01 PLC cabinet

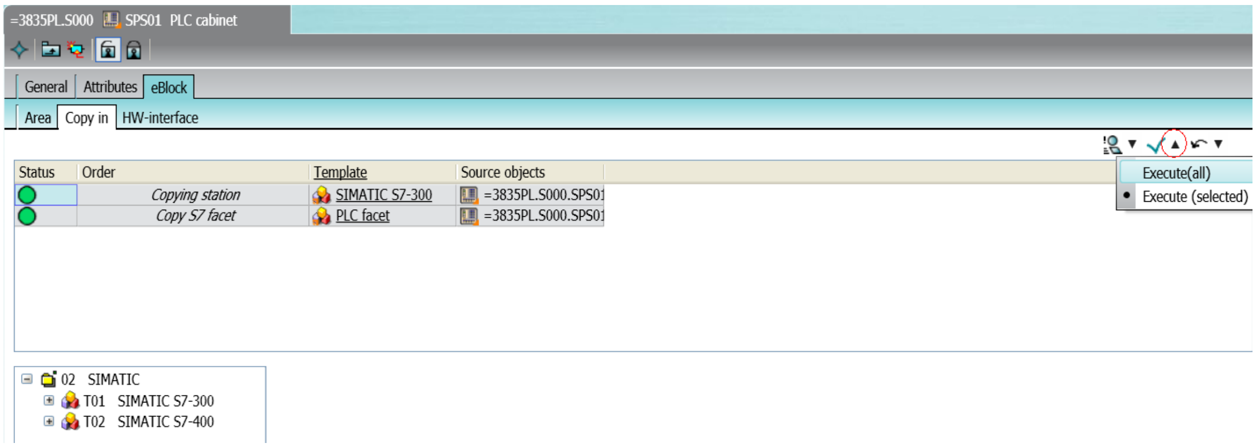
General | Attributes | eBlock

Area | Copy in | HW-interface

Status	Order	Target object	Source objects
	Assign PLC	=3835PL.S000.SPS01	=3835PL.S000.SPS01

10. Switch to the Copy in tab. Here we need to copy in the hardware configuration for the PLC and the STEP 7 source code that is used in the top level of the PLC program e.g. OB1, DB20. For the hardware configuration use the SIMATIC S7-300. To execute all eBlocks press the

arrow next to the checker icon (red circle) and select Execute(all).



=3835PL.S000 SPS01 PLC cabinet

General | Attributes | eBlock

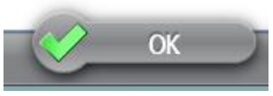
Area | Copy in | HW-interface

Status	Order	Template	Source objects
●	Copying station	SIMATIC S7-300	=3835PL.S000.SPS01
●	Copy S7 facet	PLC facet	=3835PL.S000.SPS01

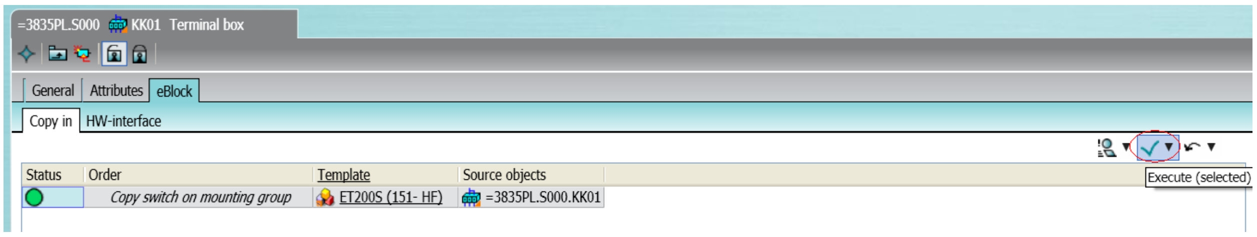
Execute(all)
• Execute (selected)

02 SIMATIC
T01 SIMATIC S7-300
T02 SIMATIC S7-400

11. The HW-interface tab in the eBlocks requires that all the devices in the project e.g. Robots, Roller has been added and configured before they can be executed. Theses will be dealt with later. Press OK to save and close the PLC cabinet property window.



12. Double click on the Terminal box in the navigation tree to open its properties in the configuration area and select the eBlock tab.
13. Press the execute icon to copy in the hardware configuration for the Terminal box. As for the PLC cabinet the HW-interface can't be configured until later.



=3835PL.S000 KK01 Terminal box

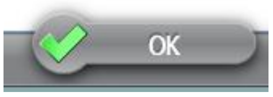
General | Attributes | eBlock

Copy in | HW-interface

Status	Order	Template	Source objects
●	Copy switch on mounting group	ET200S (151- HF)	=3835PL.S000.KK01

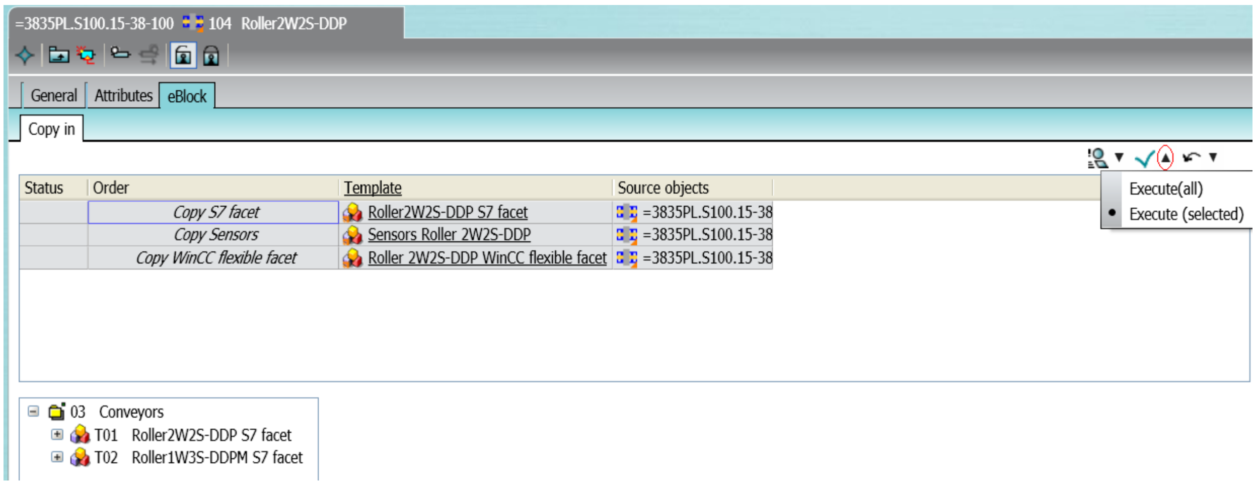
Execute (selected)

14. Press OK to save and close the Terminal box property window.



A.3 Configuration of Roller 2W2S-DDP

1. Open the properties for the Roller you wish to configure and select the eBlock tab.
2. At first there are three different eBlocks under the copy in tab. The eBlock "Copy S7 facet" copies in all FB:s, FC:s and DB:s that the roller is using. "Copy Sensors" copies in six predefined sensors that the roller needs. "Copy WinCC flexible facet" supplies the base objects needed for generating HMI screens for the roller. To execute all eBlocks at the same time press the arrow next to the execute icon (red circle) and select Execute(all).


=3835PL.S100.15-38-100 104 Roller2W2S-DDP

General | Attributes | eBlock

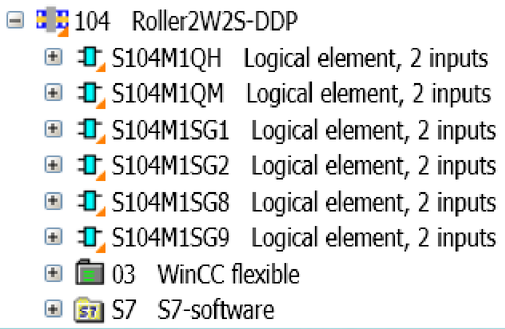
Copy in

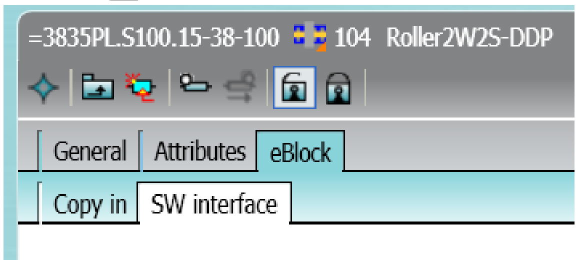
Execute(all)
• Execute (selected)

Status	Order	Template	Source objects
	Copy S7 facet	Roller2W2S-DDP S7 facet	=3835PL.S100.15-38
	Copy Sensors	Sensors Roller 2W2S-DDP	=3835PL.S100.15-38
	Copy WinCC flexible facet	Roller 2W2S-DDP WinCC flexible facet	=3835PL.S100.15-38

03 Conveyors
T01 Roller2W2S-DDP S7 facet
T02 Roller1W3S-DDPM S7 facet

3. You should now have some new objects under your roller in the navigation tree as well as a tab called “SW interface” in the configuration display of the roller.


104 Roller2W2S-DDP
S104M1QH Logical element, 2 inputs
S104M1QM Logical element, 2 inputs
S104M1SG1 Logical element, 2 inputs
S104M1SG2 Logical element, 2 inputs
S104M1SG8 Logical element, 2 inputs
S104M1SG9 Logical element, 2 inputs
03 WinCC flexible
S7 S7-software

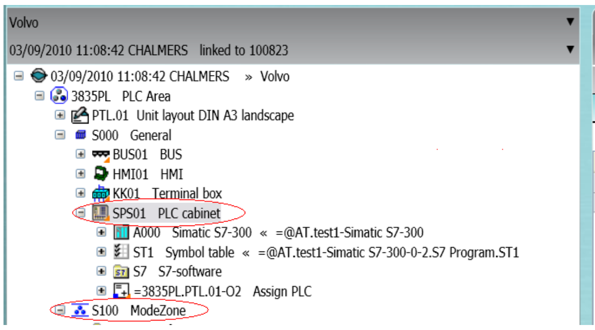

=3835PL.S100.15-38-100 104 Roller2W2S-DDP

General | Attributes | eBlock

Copy in | SW interface

4. In the software interface the hierarchal connection between the function blocks is made. The software interface is used by the inline macros in the STEP 7 source code objects. In this case we want the mode zone to be the one calling the roller. Drag the ModeZone object from the navigation tree to the target object cell for “Assign caller FB”. VCC also uses a data block called DB20 to store all the variables that are used for interlocks. In this structure the DB20 is placed in the S7-software folder for the PLC cabinet. To be able to automatically create these interlock variables for the roller we need to assign a connection between the roller and the PLC cabinet. This is done by dragging the PLC cabinet object from the navigation tree to the

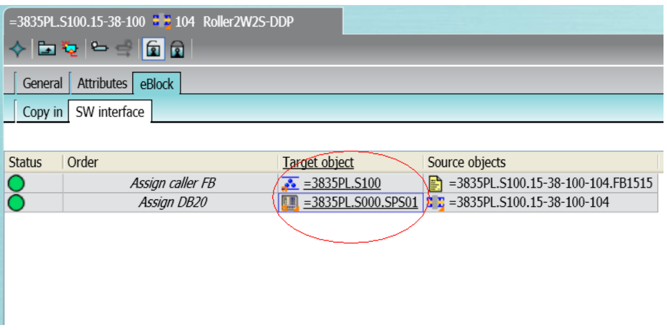
target object cell for “Assign DB20”.



Volvo

03/09/2010 11:08:42 CHALMERS linked to 100823

```
⊟ 03/09/2010 11:08:42 CHALMERS » Volvo
  ⊟ 3835PL PLC Area
    ⊞ PTL.01 Unit layout DIN A3 landscape
    ⊟ S000 General
      ⊞ BUS01 BUS
      ⊞ HMI01 HMI
      ⊞ KK01 Terminal box
      ⊟ SPS01 PLC cabinet
        ⊞ A000 Simatic S7-300 « =@AT.test1-Simatic S7-300
        ⊞ ST1 Symbol table « =@AT.test1-Simatic S7-300-0-2.S7 Program.ST1
        ⊞ S7 S7-software
        ⊞ =3835PL.PTL.01-O2 Assign PLC
    ⊟ S100 ModeZone
```



=3835PL.S100.15-38-100 104 Roller2W2S-DDP

General | Attributes | eBlock

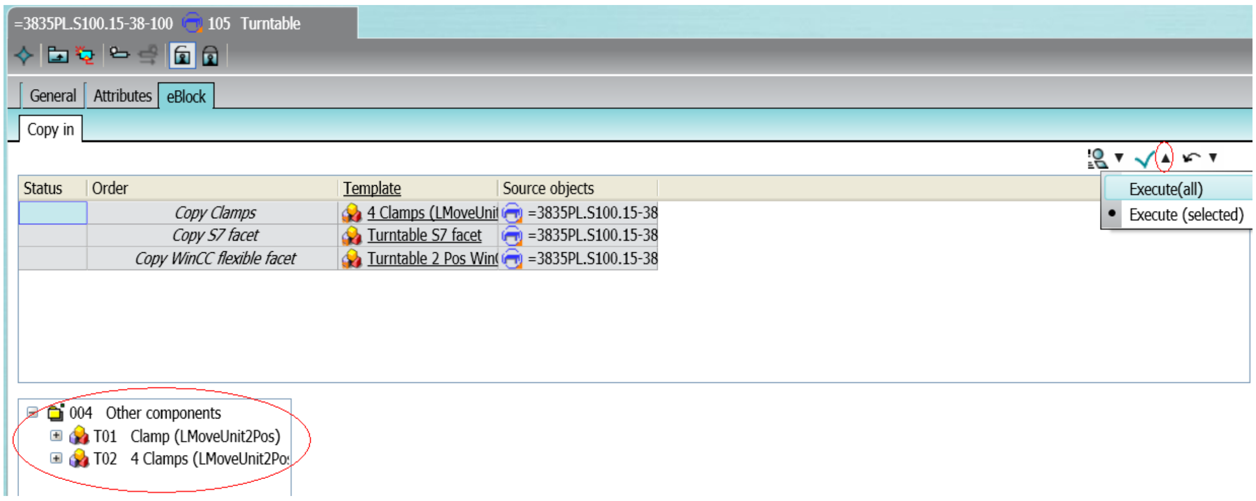
Copy in | SW interface

Status	Order	Target object	Source objects
●	*Assign caller FB*	=3835PL.S100	=3835PL.S100.15-38-100-104.FB1515
●	*Assign DB20*	=3835PL.S000.SPS01	=3835PL.S100.15-38-100-104

5. Execute the eBlocks as in step 2.
6. The roller should now be configured. Press OK.

A.4 Configuration of Turntable

1. Open the properties for the turntable you wish to configure and select the eBlock tab.
2. At first there are three different eBlocks under the copy in tab. The eBlock “Copy S7 facet” copies in all FB:s, FC:s and DB:s that the roller is using. “Copy WinCC flexible facet” supplies the base objects needed for generating HMI screens for the roller. “Copy Clamps” adds four new clamps to the turntable as default. It is possible to only copy in one clamp. To do this select the copy clamps eBlock and double click on the Clamp (LmoveUnit2Pos) template in the box below. To execute all eBlocks at the same time press the arrow next to the checker icon and select Execute(all).



=3835PL.S100.15-38-100 105 Turntable

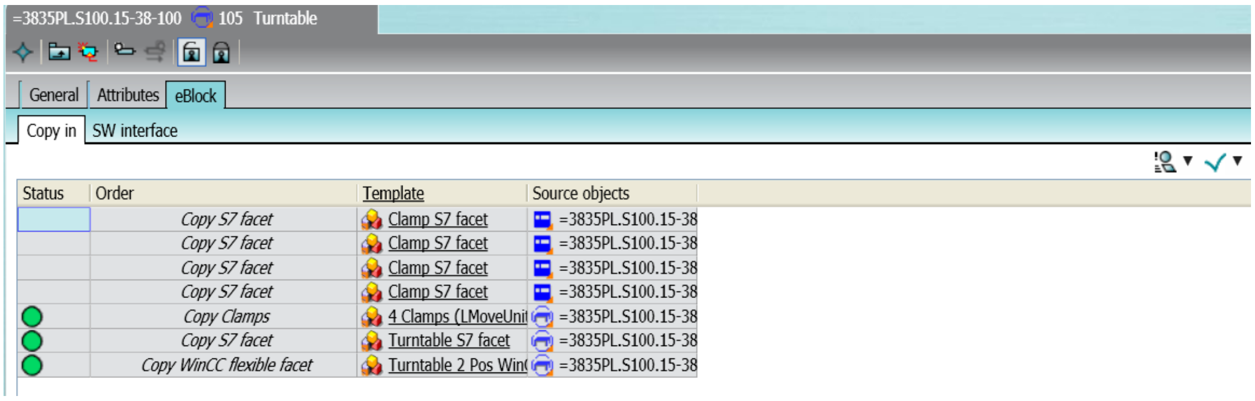
General | Attributes | eBlock

Copy in

Status	Order	Template	Source objects
	Copy Clamps	4 Clamps (LMoveUnit	=3835PL.S100.15-38
	Copy S7 facet	Turntable S7 facet	=3835PL.S100.15-38
	Copy WinCC flexible facet	Turntable 2 Pos Win(	=3835PL.S100.15-38

```
⊟ 004 Other components
  ⊞ T01 Clamp (LMoveUnit2Pos)
  ⊞ T02 4 Clamps (LMoveUnit2Po
```

3. After executing the eBlocks four new will appear. These are the for the clamps STEP 7 code. Press Execute(all) again to copy in the STEP 7 code blocks for the clamps.

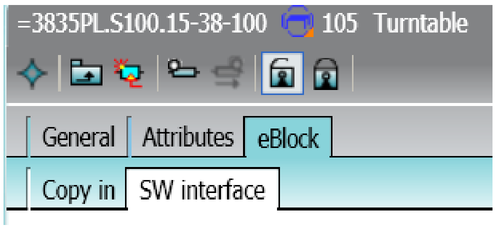

=3835PL.S100.15-38-100 105 Turntable

General | Attributes | eBlock

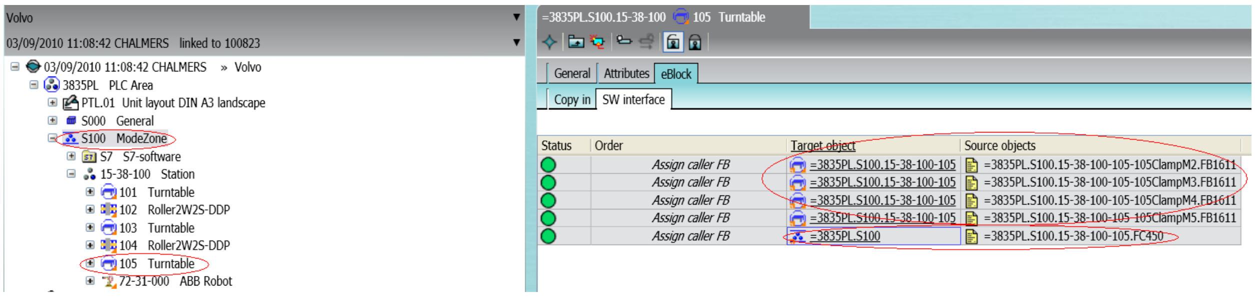
Copy in | SW interface

Status	Order	Template	Source objects
	Copy S7 facet	Clamp S7 facet	=3835PL.S100.15-38
	Copy S7 facet	Clamp S7 facet	=3835PL.S100.15-38
	Copy S7 facet	Clamp S7 facet	=3835PL.S100.15-38
	Copy S7 facet	Clamp S7 facet	=3835PL.S100.15-38
●	*Copy Clamps*	4 Clamps (LMoveUnit	=3835PL.S100.15-38
●	*Copy S7 facet*	Turntable S7 facet	=3835PL.S100.15-38
●	*Copy WinCC flexible facet*	Turntable 2 Pos Win(	=3835PL.S100.15-38

4. Switch to the SW interface tab.



5. In the software interface the hierarchal connection between the function blocks is made. The software interface is used by the inline macros in the S7 source code objects. In this case we want the mode zone to be the one calling the turntable. Drag the ModeZone object from the navigation tree to the target object cell for the "Assign caller FB" eBlock which has a source object that ends with FC***. The four other "Assign caller FB" eBlocks are there to make the connection between the clamps and the turntables FC. Here we need to assign the turntable which the clamps belong to the target object for the "Assign caller FB" eBlocks. Just drag the turntable from the navigation tree to the target object cells.


Volvo

03/09/2010 11:08:42 CHALMERS linked to 100823

- 03/09/2010 11:08:42 CHALMERS » Volvo
  - 3835PL PLC Area
    - PTL.01 Unit layout DIN A3 landscape
    - S000 General
    - S100 ModeZone
      - S7 S7-software
      - 15-38-100 Station
        - 101 Turntable
        - 102 Roller2W2S-DDP
        - 103 Turntable
        - 104 Roller2W2S-DDP
        - 105 Turntable
        - 72-31-000 ABB Robot

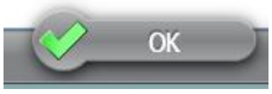
=3835PL.S100.15-38-100 105 Turntable

General | Attributes | eBlock

Copy in | SW interface

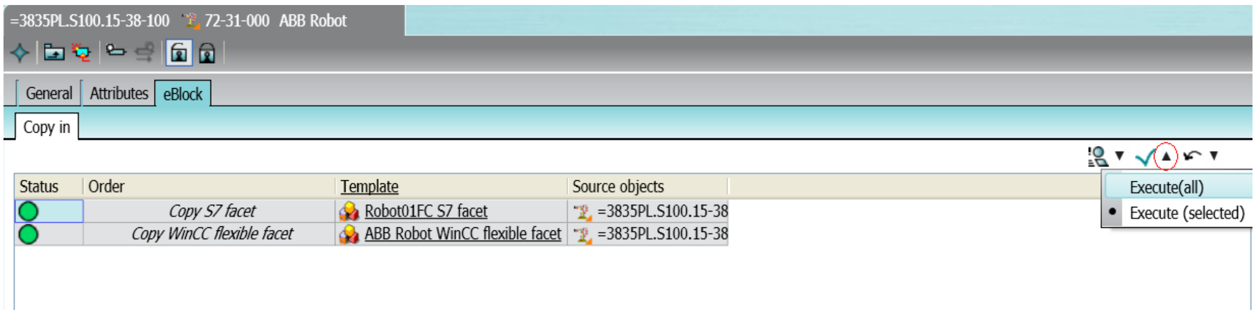
Status	Order	Target object	Source objects
●	*Assign caller FB*	=3835PL.S100.15-38-100-105	=3835PL.S100.15-38-100-105-105ClampM2.FB1611
●	*Assign caller FB*	=3835PL.S100.15-38-100-105	=3835PL.S100.15-38-100-105-105ClampM3.FB1611
●	*Assign caller FB*	=3835PL.S100.15-38-100-105	=3835PL.S100.15-38-100-105-105ClampM4.FB1611
●	*Assign caller FB*	=3835PL.S100.15-38-100-105	=3835PL.S100.15-38-100-105-105ClampM5.FB1611
●	*Assign caller FB*	=3835PL.S100	=3835PL.S100.15-38-100-105.FC450

6. Execute the eBlocks as in step 2.
7. The turntable should now be configured. Press OK.



A.5 Configuration of ABB Robot

1. Open the properties for the turntable you wish to configure and select the eBlock tab.

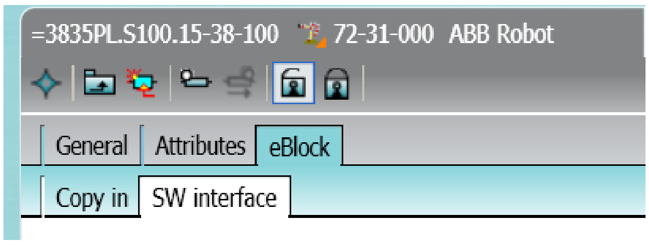

=3835PL.S100.15-38-100 72-31-000 ABB Robot

General | Attributes | eBlock

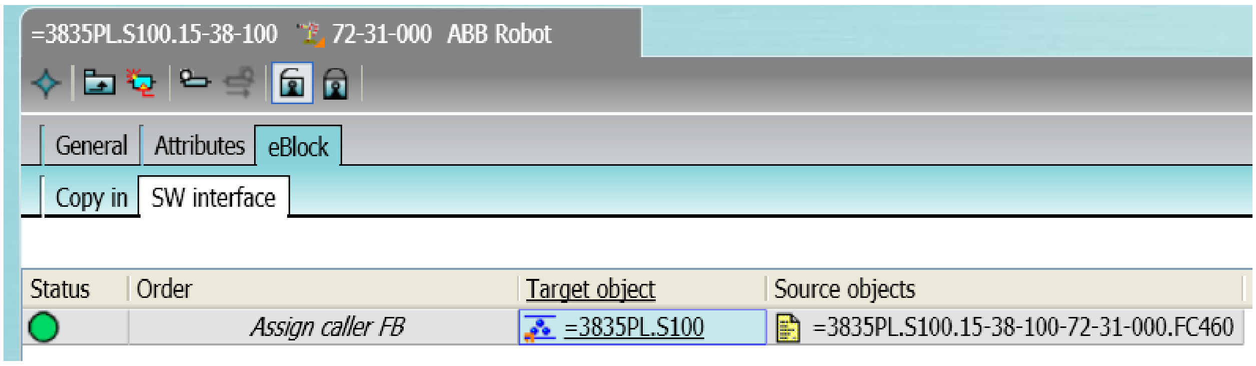
Copy in

Status	Order	Template	Source objects
🟢	*Copy S7 facet*	Robot01FC S7 facet	=3835PL.S100.15-38
🟢	*Copy WinCC flexible facet*	ABB Robot WinCC flexible facet	=3835PL.S100.15-38

2. At first there are two different eBlocks under the copy in tab. The eBlock “Copy S7 facet” copies in all FB:s, FC:s and DB:s that the roller is using. “Copy WinCC flexible facet” supplies the base objects needed for generating HMI screens for the roller. To execute all eBlocks at the same time press the arrow next to the checker icon and select Execute(all).
3. Switch to the SW interface tab.



4. In the software interface the hierarchal connection between the function blocks is made. The software interface is used by the inline macros in the S7 source code objects. In this case we want the mode zone to be the one calling the robot. Drag the ModeZone object from the navigation tree to the target object cell for “Assign caller FB”.

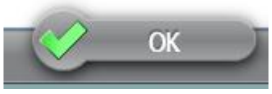

=3835PL.S100.15-38-100 72-31-000 ABB Robot

General | Attributes | eBlock

Copy in | SW interface

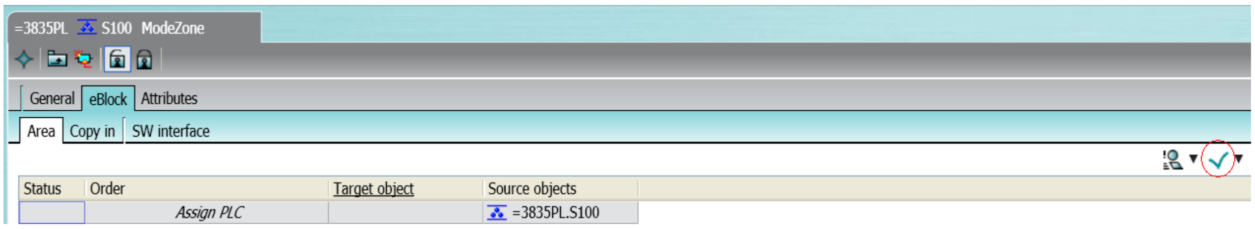
Status	Order	Target object	Source objects
🟢	*Assign caller FB*	=3835PL.S100	=3835PL.S100.15-38-100-72-31-000.FC460

5. Execute the eBlocks as in step 2.
6. The robot should now be configured. Press OK.



A.6 Configuration of ModeZone

1. Open the properties for the ModeZone you wish to configure and select the eBlock tab.
2. In the Area tab we need to assign a PLC to the ModeZone. Drag the PLC cabinet from the general folder to the target object. Press the checker icon to execute the eBlock


=3835PL S100 ModeZone

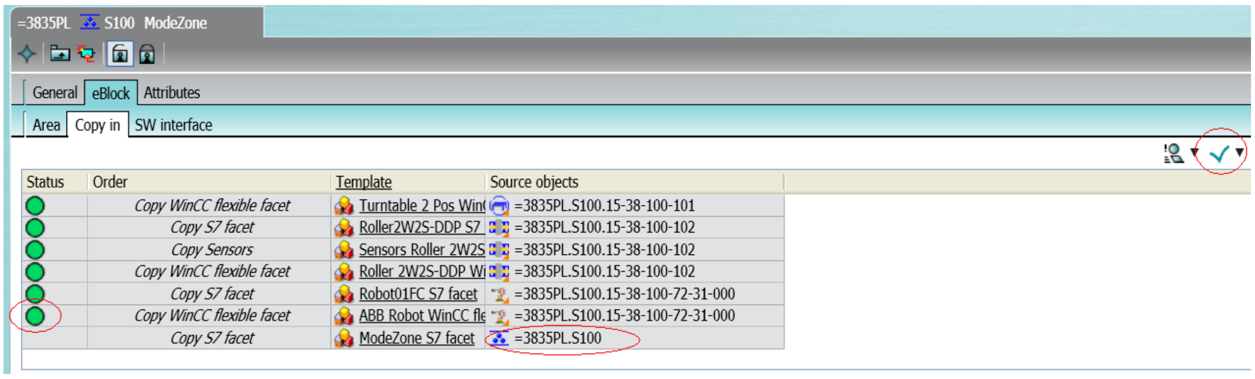
General | eBlock | Attributes

Area | Copy in | SW interface

Status	Order	Target object	Source objects
	Assign PLC		=3835PL.S100

.

3. Switch to the copy in tab
4. In the eBlocks tab for the ModeZone we can see the eBlocks for all devices that resides under it in the navigation tree e.g. conveyors and robots. These eBlocks can be executed from here so we don't need to open each object separately to execute its eBlocks as we did in chapter A.3, A.4 and A.5. All eBlocks that has been already executed should have a green circle in its status cell. Beware though that the green circle can be bugged sometimes and will show an incorrect status so to be on the safe side press the arrow next to the checker icon and select execute (all). The only unique eBlock for the ModeZone in the copy in tab is for copying in the STEP 7 code. Notice in the source object column that eBlock is linked to the ModeZone.

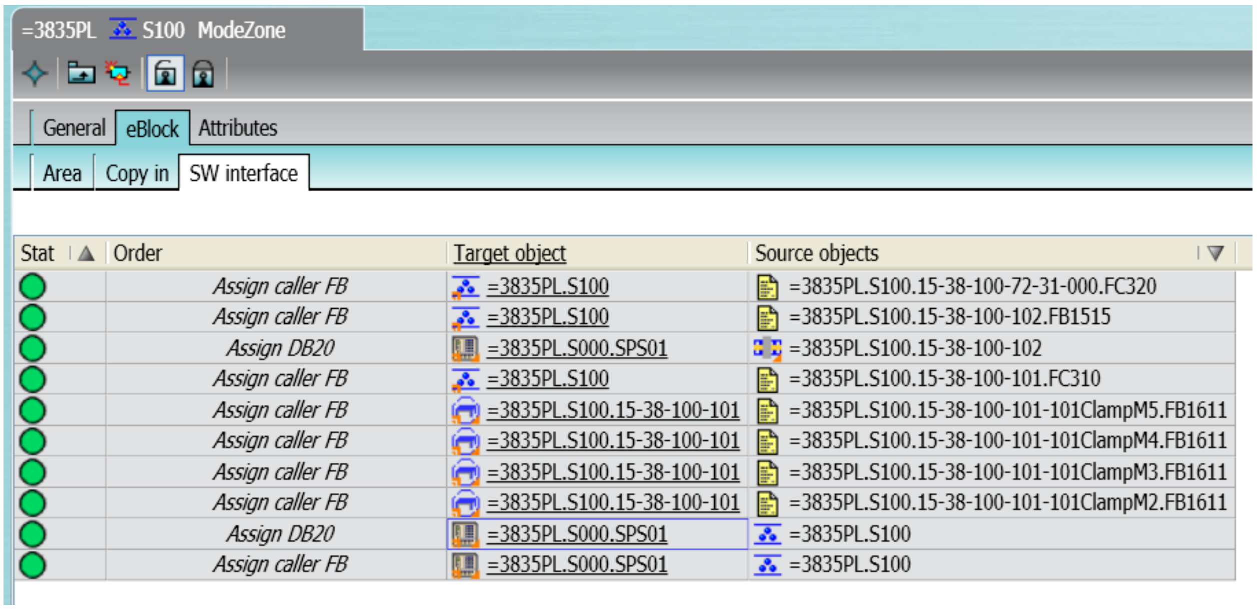

=3835PL S100 ModeZone

General | eBlock | Attributes

Area | Copy in | SW interface

Status	Order	Template	Source objects
●	*Copy WinCC flexible facet*	Turntable 2 Pos Win(	=3835PL.S100.15-38-100-101
●	*Copy S7 facet*	Roller2W2S-DDP S7	=3835PL.S100.15-38-100-102
●	*Copy Sensors*	Sensors Roller 2W2S	=3835PL.S100.15-38-100-102
●	*Copy WinCC flexible facet*	Roller 2W2S-DDP Wi	=3835PL.S100.15-38-100-102
●	*Copy S7 facet*	Robot01FC S7 facet	=3835PL.S100.15-38-100-72-31-000
●	*Copy WinCC flexible facet*	ABB Robot WinCC fle	=3835PL.S100.15-38-100-72-31-000
	Copy S7 facet	ModeZone S7 facet	=3835PL.S100

5. Switch to the SW interface tab. If you choose to work with the eBlocks from all devices here in the property window for the ModeZone you need to be careful of which source object you assign your target objects to. The two eblock connected to the ModeZone needs to have the PLC cabinet from the general folder as target object.


=3835PL S100 ModeZone

General | eBlock | Attributes

Area | Copy in | SW interface

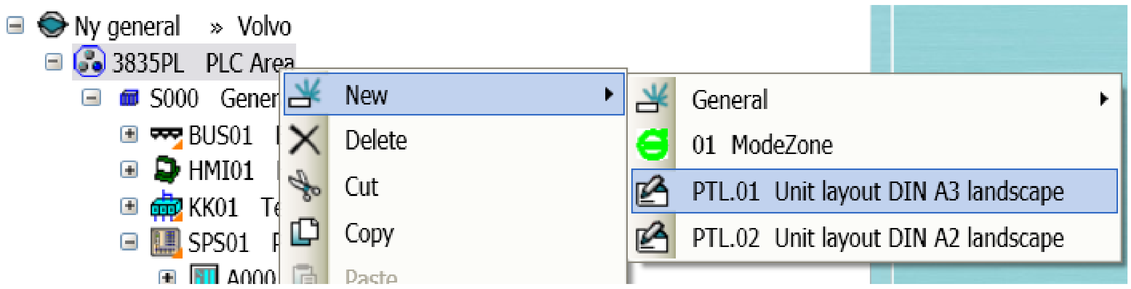
Stat	Order	Target object	Source objects
●	*Assign caller FB*	=3835PL.S100	=3835PL.S100.15-38-100-72-31-000.FC320
●	*Assign caller FB*	=3835PL.S100	=3835PL.S100.15-38-100-102.FB1515
●	*Assign DB20*	=3835PL.S000.SPS01	=3835PL.S100.15-38-100-102
●	*Assign caller FB*	=3835PL.S100	=3835PL.S100.15-38-100-101.FC310
●	*Assign caller FB*	=3835PL.S100.15-38-100-101	=3835PL.S100.15-38-100-101-101ClampM5.FB1611
●	*Assign caller FB*	=3835PL.S100.15-38-100-101	=3835PL.S100.15-38-100-101-101ClampM4.FB1611
●	*Assign caller FB*	=3835PL.S100.15-38-100-101	=3835PL.S100.15-38-100-101-101ClampM3.FB1611
●	*Assign caller FB*	=3835PL.S100.15-38-100-101	=3835PL.S100.15-38-100-101-101ClampM2.FB1611
●	*Assign DB20*	=3835PL.S000.SPS01	=3835PL.S100
●	*Assign caller FB*	=3835PL.S000.SPS01	=3835PL.S100

6. After assigning the target objects press the arrow next to the checker icon and select execute(all).
7. The robot should now be configured. Press OK.

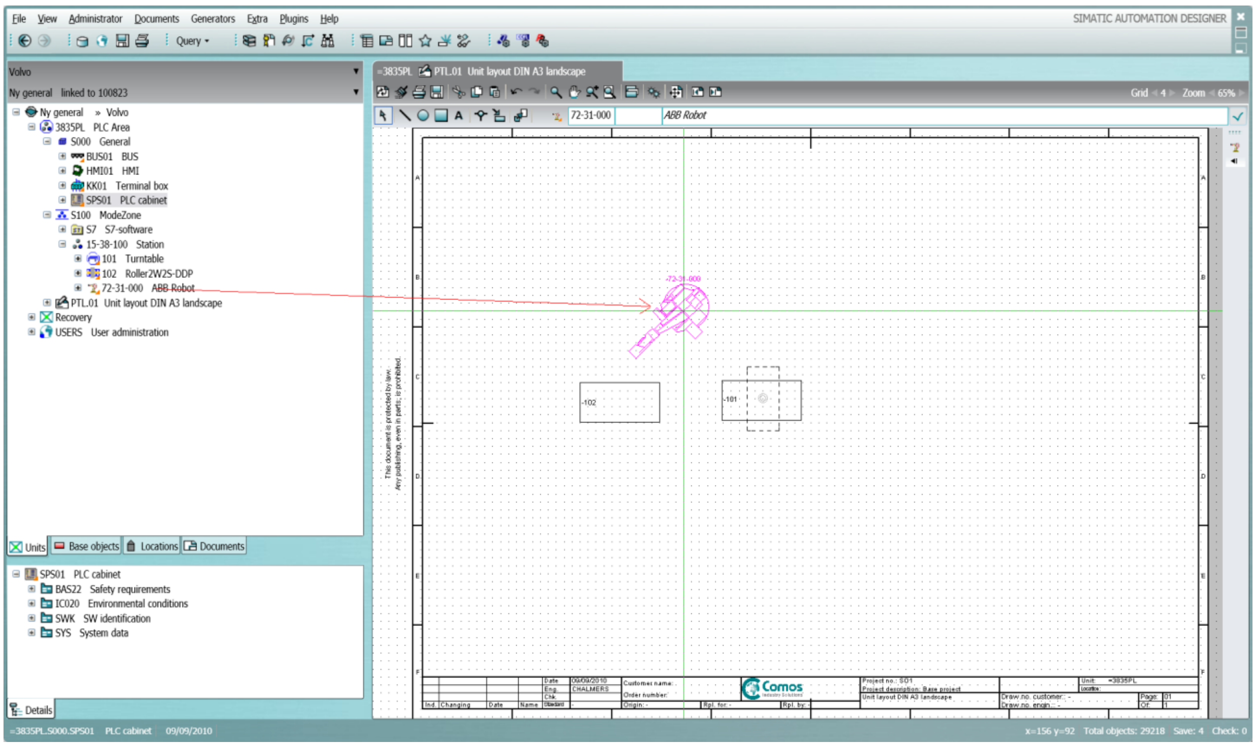
A.7 Unit layout and polygons

The unit layout and polygons are used to quickly assign specific eBlocks with the same target object to multiple source objects on a layout drawing.

1. First add a new unit layout object by right clicking on the PLC Area and select New -> Unit layout DIN A3 landscape.



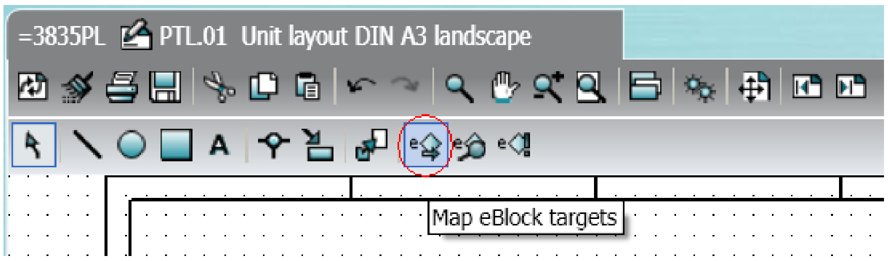
2. Open the unit layout by double clicking on it in the navigation tree.
3. To add objects to the unit layout just drag them from the navigation tree and drop them on the layout.


File View Administrator Documents Generators Extra Plugins Help
SIMATIC AUTOMATION DESIGNER
Query
Volvo
Ny general linked to 100823
Ny general » Volvo
3835PL PLC Area
S000 General
BUS01 BUS
HMI01 HMI
KK01 Terminal box
SPS01 PLC cabinet
S100 ModeZone
S7 S7-software
15-38-100 Station
101 Turntable
102 Roller2W2S-DDP
72-31-000 ABB-Robot
PTL.01 Unit layout DIN A3 landscape
Recovery
USERS User administration
Units Base objects Locations Documents
SPS01 PLC cabinet
BAS22 Safety requirements
IC020 Environmental conditions
SWK SW identification
SYS System data
Details
=3835PL PTL.01 Unit layout DIN A3 landscape
72-31-000
ABB Robot
Grid 4 Zoom 65%
-102
-101
Comos
=3835PL.S000.SPS01 PLC cabinet 09/09/2010
x=156 y=92 Total objects: 29218 Save: 4 Check: 0

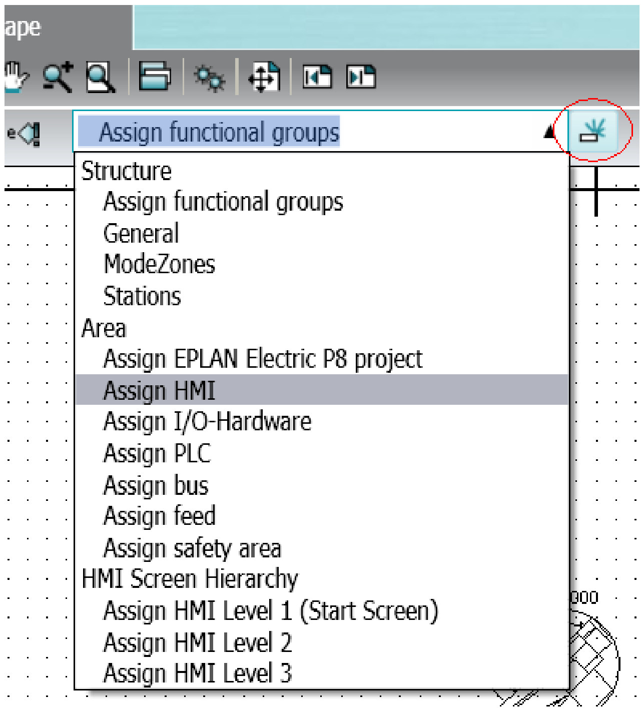
The objects that needs to be added are the following:

- All robots, rollers and turntables.
- The ModeZone
- HMI, BUS , Terminal box and PLC cabinet from the general folder.

4. To start mapping eBlocks press the Map eBlock targets button (red circle).

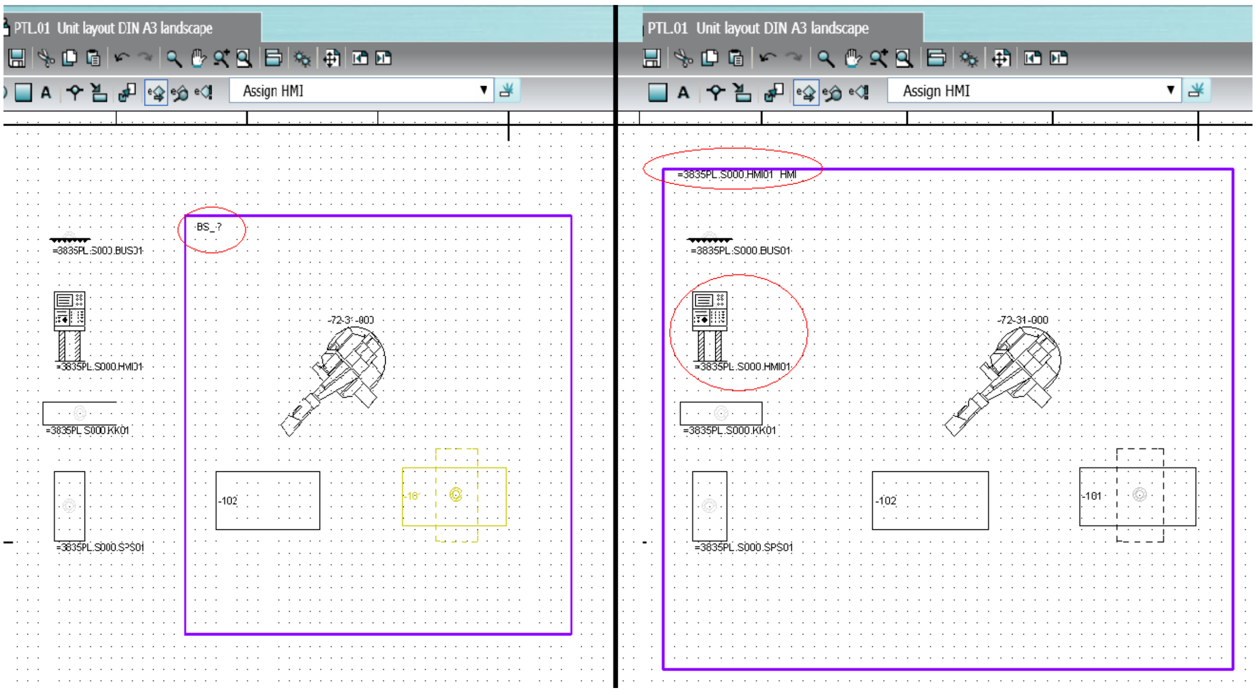


5. In the drop down list that appears are the different eBlocks that is possible to map with polygons. Start with selecting the “Assign HMI” and press the button next to the list.

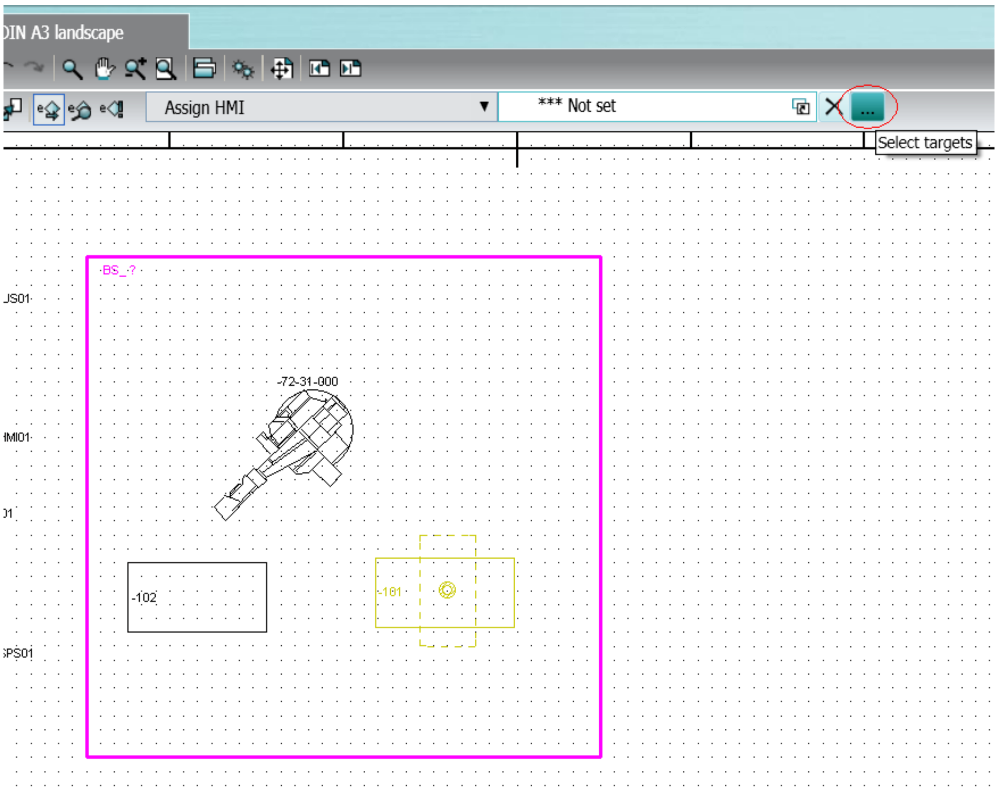


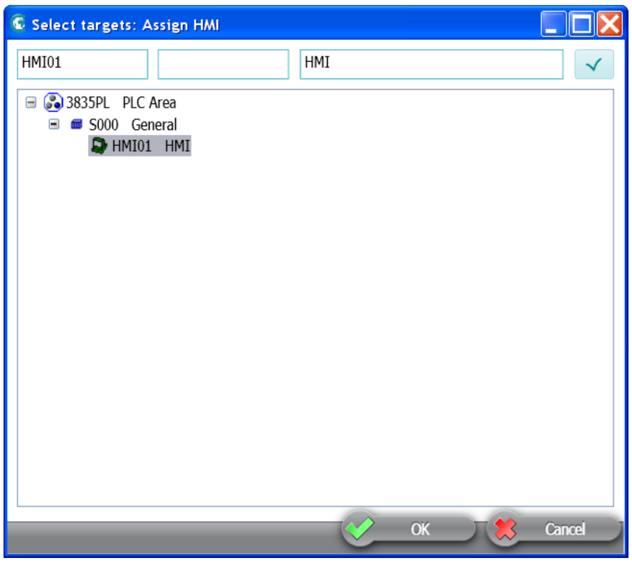
6. A polygon could either be dragged by dragging lines until it forms a closed loop or only a rectangular shape is needed by placing a line that forms the diagonal of the rectangle and press the right mouse button.
7. When creating a polygon around your objects Automation Designer will automatically select a target object if a viable one is placed within the polygon. You can see the name of the

target object in the upper left corner of the polygon.



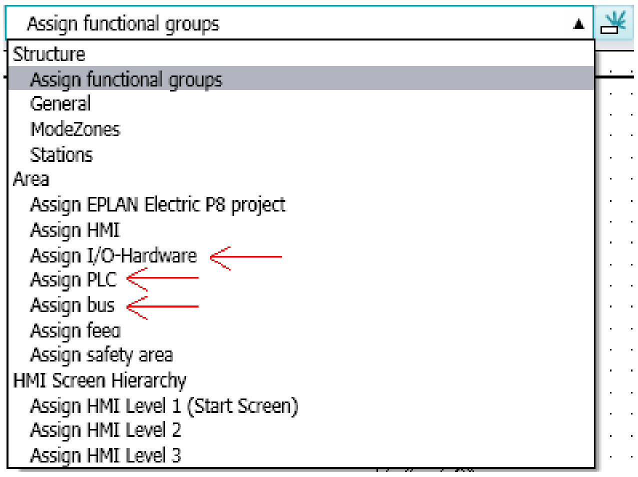
8. If the target object is not automatically assigned or the target object is wrong you can assign it manually. To do this click on the polygon. Press the button with three dots. Select the appropriate object in the list that appears and press OK to confirm.



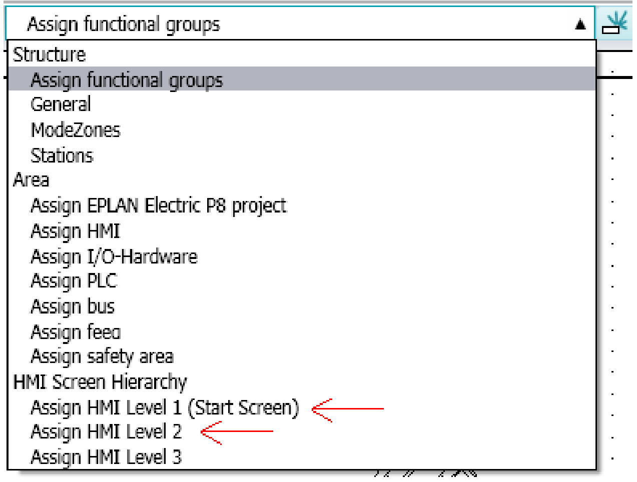


9. Repeat step 5 – 8 for the I/O-Hardware, the PLC and for the bus and assign the target objects described in the table below.

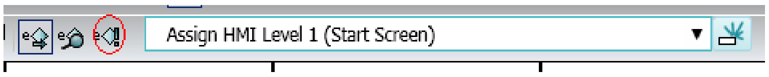
eBlock	Target object
Assign HMI	General -> HMI
Assign I/O-Hardware	General -> Terminal box
Assign PLC	General -> PLC cabinet
Assign bus	General -> BUS



10. The layouts for the generated HMI screens are set with polygons in the 2D layout. This is done with the "Assign HMI level 1-3" eBlocks. In this project only level 1 and 2 are used. Level 1 will represent the top level controlled by OB1.The polygon for this level should be dragged so it covers the entire 2D layout. Level 2 represents the mode zones. These should be dragged so only the objects included in the specific mode zone are covered by the polygon. Several level 2 polygons can be made depending on the number of mod zones. Assign HMI level shall have the General -> HMI object as target object.

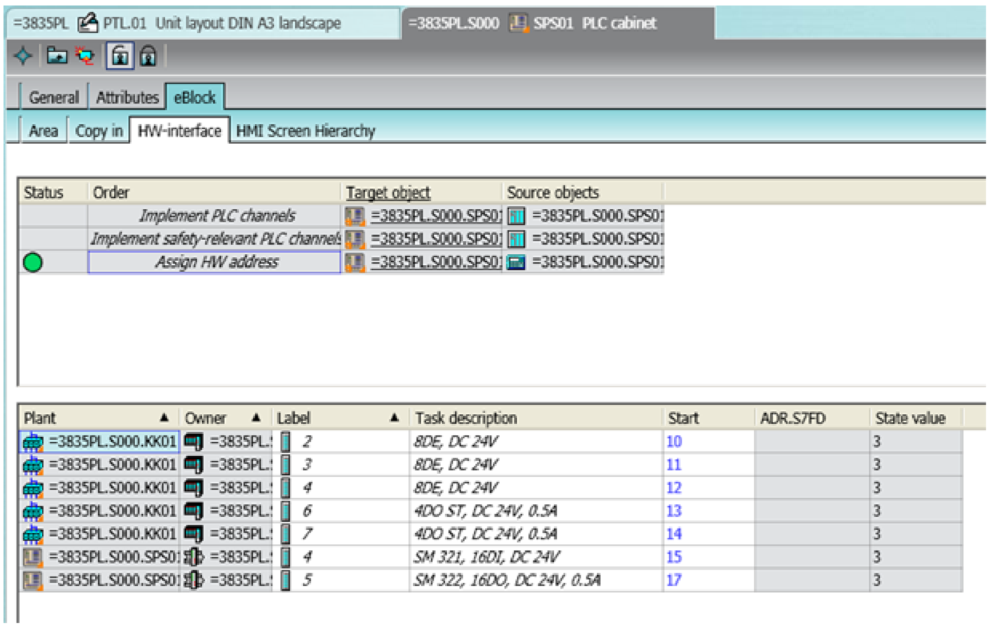


11. To execute all eBlocks press the "Calculate eBlock target" button



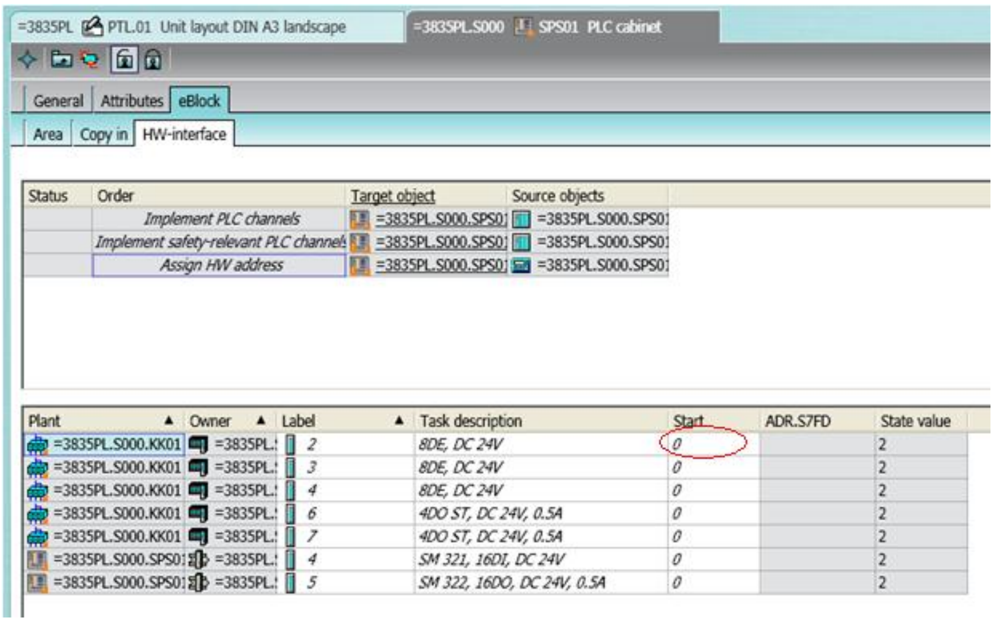
A.8 Configure the hardware interface

1. Open the PLC cabinet object and select the eBlock -> HW-interface tab. Select Assign HW address. In the start column assign a bus address for the first object and press execute. This will give bus addresses to all other objects as well.


=3835PL PTL.01 Unit layout DIN A3 landscape
=3835PL.S000 SPS01 PLC cabinet
General
Attributes
eBlock
Area
Copy in
HW-interface
HMI Screen Hierarchy

Status	Order	Target object	Source objects
	Implement PLC channels	=3835PL.S000.SPS01	=3835PL.S000.SPS01
	Implement safety-relevant PLC channels	=3835PL.S000.SPS01	=3835PL.S000.SPS01
●	*Assign HW address*	=3835PL.S000.SPS01	=3835PL.S000.SPS01

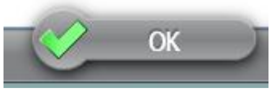
Plant	Owner	Label	Task description	Start	ADR.S7FD	State value
=3835PL.S000.KK01	=3835PL.	*2*	*8DE, DC 24V*	10		3
=3835PL.S000.KK01	=3835PL.	*3*	*8DE, DC 24V*	11		3
=3835PL.S000.KK01	=3835PL.	*4*	*8DE, DC 24V*	12		3
=3835PL.S000.KK01	=3835PL.	*6*	*4DO ST, DC 24V, 0.5A*	13		3
=3835PL.S000.KK01	=3835PL.	*7*	*4DO ST, DC 24V, 0.5A*	14		3
=3835PL.S000.SPS01	=3835PL.	*4*	*SM 321, 16DI, DC 24V*	15		3
=3835PL.S000.SPS01	=3835PL.	*5*	*SM 322, 16DO, DC 24V, 0.5A*	17		3


=3835PL PTL.01 Unit layout DIN A3 landscape
=3835PL.S000 SPS01 PLC cabinet
General
Attributes
eBlock
Area
Copy in
HW-interface

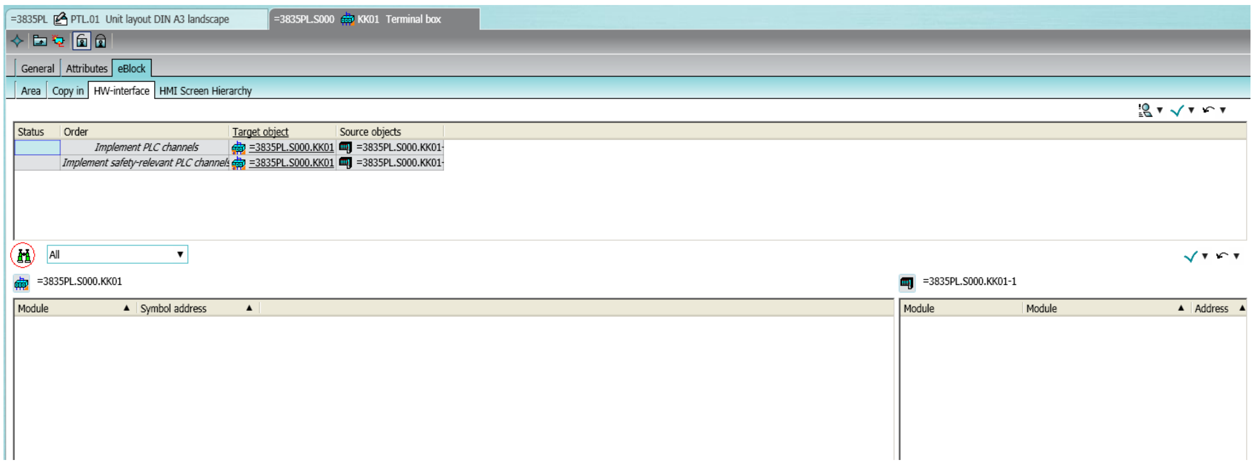
Status	Order	Target object	Source objects
	Implement PLC channels	=3835PL.S000.SPS01	=3835PL.S000.SPS01
	Implement safety-relevant PLC channels	=3835PL.S000.SPS01	=3835PL.S000.SPS01
	Assign HW address	=3835PL.S000.SPS01	=3835PL.S000.SPS01

Plant	Owner	Label	Task description	Start	ADR.S7FD	State value
=3835PL.S000.KK01	=3835PL.	*2*	*8DE, DC 24V*	*0*		2
=3835PL.S000.KK01	=3835PL.	*3*	*8DE, DC 24V*	*0*		2
=3835PL.S000.KK01	=3835PL.	*4*	*8DE, DC 24V*	*0*		2
=3835PL.S000.KK01	=3835PL.	*6*	*4DO ST, DC 24V, 0.5A*	*0*		2
=3835PL.S000.KK01	=3835PL.	*7*	*4DO ST, DC 24V, 0.5A*	*0*		2
=3835PL.S000.SPS01	=3835PL.	*4*	*SM 321, 16DI, DC 24V*	*0*		2
=3835PL.S000.SPS01	=3835PL.	*5*	*SM 322, 16DO, DC 24V, 0.5A*	*0*		2

2. Press OK.



3. Open the terminal box object and select the HW interface tab. Press the binocular icon.


=3835PL PTL.01 Unit layout DIN A3 landscape
=3835PL.S000 KK01 Terminal box
General
Attributes
eBlock
Area
Copy in
HW-interface
HMI Screen Hierarchy
All

Status	Order	Target object	Source objects
	Implement PLC channels	=3835PL.S000.KK01	=3835PL.S000.KK01-
	Implement safety-relevant PLC channels	=3835PL.S000.KK01	=3835PL.S000.KK01-

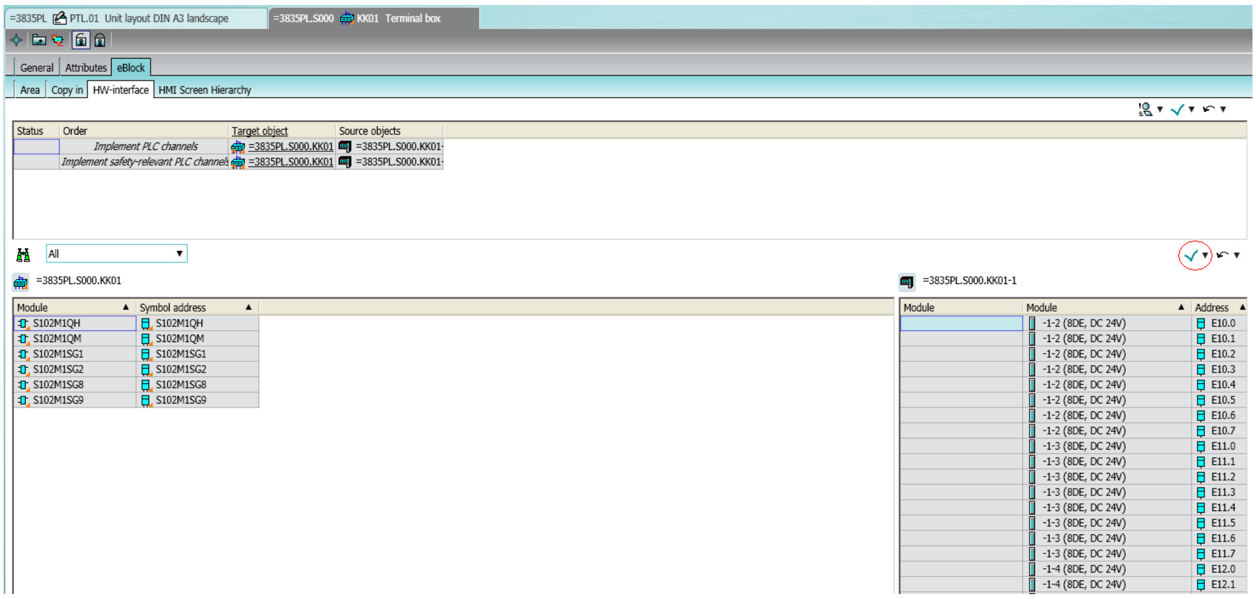
=3835PL.S000.KK01

Module	Symbol address

=3835PL.S000.KK01-1

Module	Module	Address

4. The sensors will appear in the lower left window. Press the arrow next to the lower checker icon and choose Implement (all) to automatically assign the sensors to hardware addresses.


=3835PL PTL.01 Unit layout DIN A3 landscape
=3835PL.S000 KK01 Terminal box
General
Attributes
eBlock
Area
Copy in
HW-interface
HMI Screen Hierarchy
All

Status	Order	Target object	Source objects
	Implement PLC channels	=3835PL.S000.KK01	=3835PL.S000.KK01-
	Implement safety-relevant PLC channels	=3835PL.S000.KK01	=3835PL.S000.KK01-

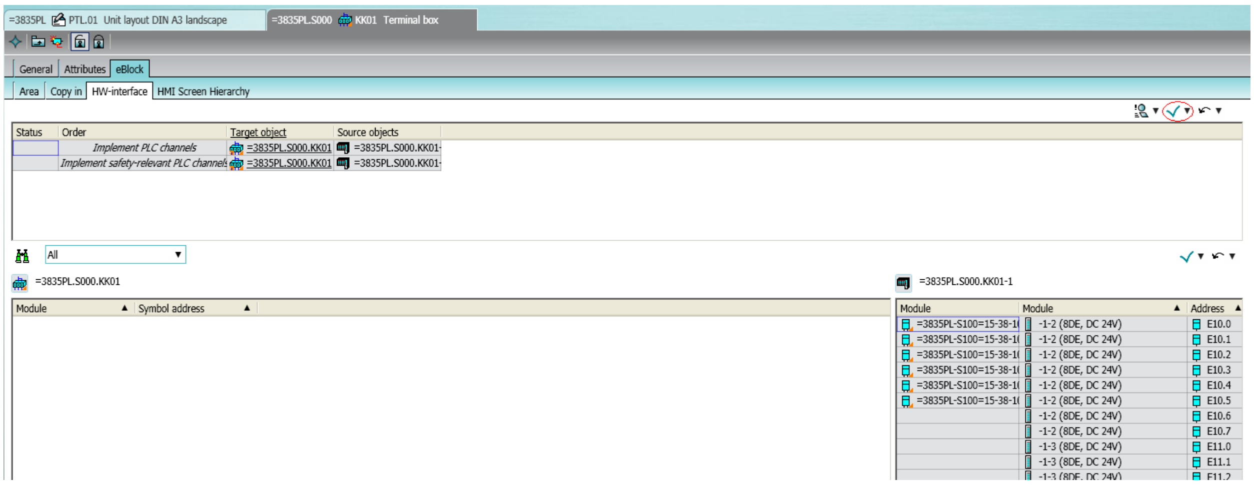
=3835PL.S000.KK01

Module	Symbol address
S102M1QH	S102M1QH
S102M1QM	S102M1QM
S102M1SG1	S102M1SG1
S102M1SG2	S102M1SG2
S102M1SG8	S102M1SG8
S102M1SG9	S102M1SG9

=3835PL.S000.KK01-1

Module	Module	Address
	-1-2 (8DE, DC 24V)	E10.0
	-1-2 (8DE, DC 24V)	E10.1
	-1-2 (8DE, DC 24V)	E10.2
	-1-2 (8DE, DC 24V)	E10.3
	-1-2 (8DE, DC 24V)	E10.4
	-1-2 (8DE, DC 24V)	E10.5
	-1-2 (8DE, DC 24V)	E10.6
	-1-2 (8DE, DC 24V)	E10.7
	-1-3 (8DE, DC 24V)	E11.0
	-1-3 (8DE, DC 24V)	E11.1
	-1-3 (8DE, DC 24V)	E11.2
	-1-3 (8DE, DC 24V)	E11.3
	-1-3 (8DE, DC 24V)	E11.4
	-1-3 (8DE, DC 24V)	E11.5
	-1-3 (8DE, DC 24V)	E11.6
	-1-3 (8DE, DC 24V)	E11.7
	-1-4 (8DE, DC 24V)	E12.0
	-1-4 (8DE, DC 24V)	E12.1

5. Press the upper checker icon to execute the eBlock.


=3835PL PTL.01 Unit layout DIN A3 landscape
=3835PL.S000 KK01 Terminal box
General
Attributes
eBlock
Area
Copy in
HW-interface
HMI Screen Hierarchy
All

Status	Order	Target object	Source objects
	Implement PLC channels	=3835PL.S000.KK01	=3835PL.S000.KK01-
	Implement safety-relevant PLC channels	=3835PL.S000.KK01	=3835PL.S000.KK01-

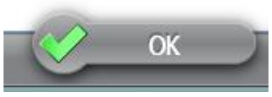
=3835PL.S000.KK01

Module	Symbol address

=3835PL.S000.KK01-1

Module	Module	Address
=3835PL-S100=15-38-1(	-1-2 (8DE, DC 24V)	E10.0
=3835PL-S100=15-38-1(	-1-2 (8DE, DC 24V)	E10.1
=3835PL-S100=15-38-1(	-1-2 (8DE, DC 24V)	E10.2
=3835PL-S100=15-38-1(	-1-2 (8DE, DC 24V)	E10.3
=3835PL-S100=15-38-1(	-1-2 (8DE, DC 24V)	E10.4
=3835PL-S100=15-38-1(	-1-2 (8DE, DC 24V)	E10.5
	-1-2 (8DE, DC 24V)	E10.6
	-1-2 (8DE, DC 24V)	E10.7
	-1-3 (8DE, DC 24V)	E11.0
	-1-3 (8DE, DC 24V)	E11.1
	-1-3 (8DE, DC 24V)	E11.2

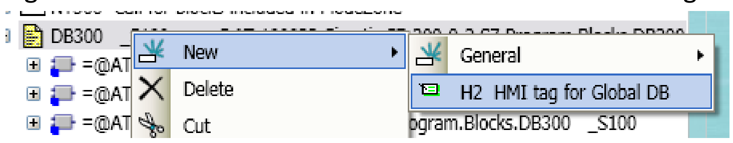
6. Press OK to save.



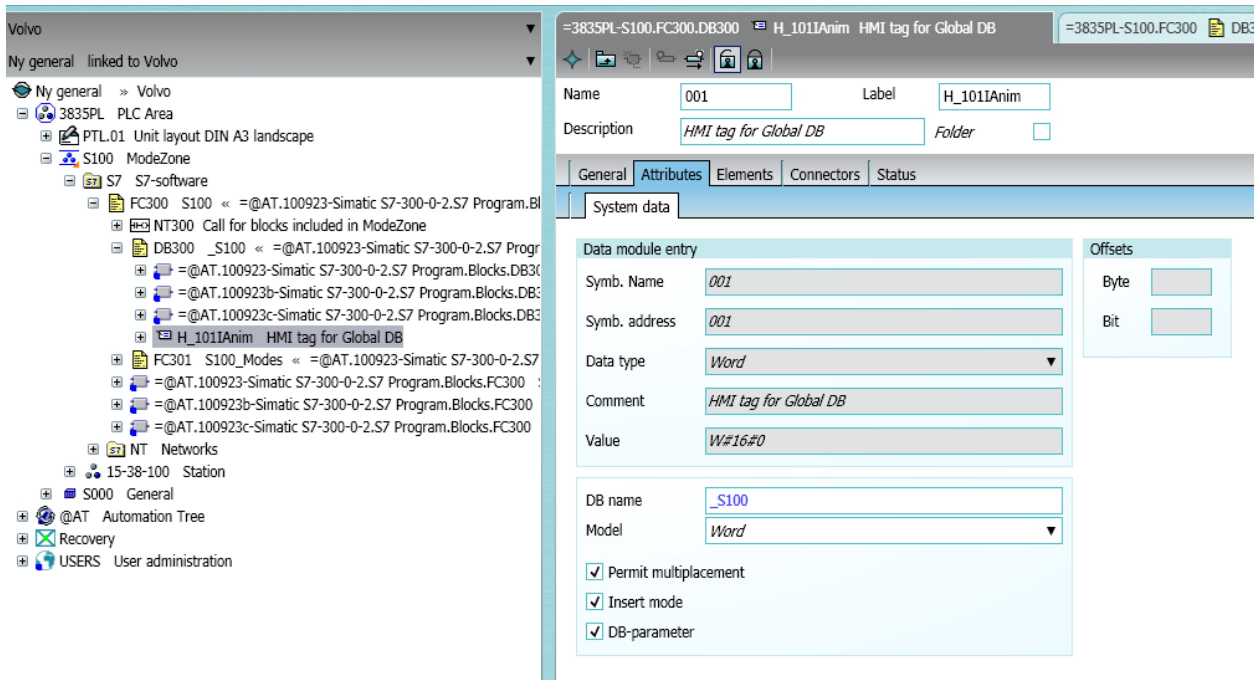
A.9 Adding H_IAnim tags and configuring the WinCC flexible project fragments

To assign variables to theHMI tags from WinCC flexible the DB address must be known. For the H_Anim tags this is done automatically but it has to be done manually for the H_IAnim tags by adding a DB variable object. One DB variable for each device is added under the data block for the mode zone and one for each mode zone is added under DB1.

1. Right click on the DB for the modezone and select HMI tag for global DB



2. Open the properties for the newly created DB variable. Select the Attributes -> System data tab. In the DB name field fill in the symbolic name for the mode zones DB. The label field shall contain the name of the H_IAnim variable. For the conveyor and turntable it is H_[Object name]IAnim. For the robots the variable name is H_Robot[robotnumber]FCIAnim e.g. H_Robot7231FCIAnim.



Volvo
Ny general linked to Volvo

- Ny general » Volvo
  - 3835PL PLC Area
    - PTL.01 Unit layout DIN A3 landscape
    - S100 ModeZone
      - S7 S7-software
        - FC300 S100 « =@AT.100923-Simatic S7-300-0-2.S7 Program.Bl
          - NT300 Call for blocks included in ModeZone
          - DB300 _S100 « =@AT.100923-Simatic S7-300-0-2.S7 Progr
            - =@AT.100923-Simatic S7-300-0-2.S7 Program.Blocks.DB3
            - =@AT.100923b-Simatic S7-300-0-2.S7 Program.Blocks.DB3
            - =@AT.100923c-Simatic S7-300-0-2.S7 Program.Blocks.DB3
            - H_101IAnim HMI tag for Global DB
          - FC301 S100_Modes « =@AT.100923-Simatic S7-300-0-2.S7
          - =@AT.100923-Simatic S7-300-0-2.S7 Program.Blocks.FC300
          - =@AT.100923b-Simatic S7-300-0-2.S7 Program.Blocks.FC300
          - =@AT.100923c-Simatic S7-300-0-2.S7 Program.Blocks.FC300
        - NT Networks
      - 15-38-100 Station
    - S000 General
  - @AT Automation Tree
  - Recovery
  - USERS User administration

=3835PL-S100.FC300.DB300 H_101IAnim HMI tag for Global DB | =3835PL-S100.FC300 DB3

Name: 001 Label: H_101IAnim
Description: *HMI tag for Global DB* *Folder*

General | Attributes | Elements | Connectors | Status

System data

Data module entry
Symb. Name: *001*
Symb. address: *001*
Data type: *Word*
Comment: *HMI tag for Global DB*
Value: *W#16#0*

Offsets
Byte:
Bit:

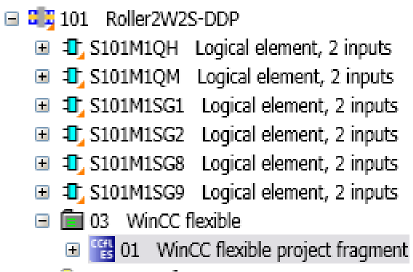
DB name: _S100
Model: *Word*

☑ Permit multiplacement
☑ Insert mode
☑ DB-parameter

3. Press OK to save.

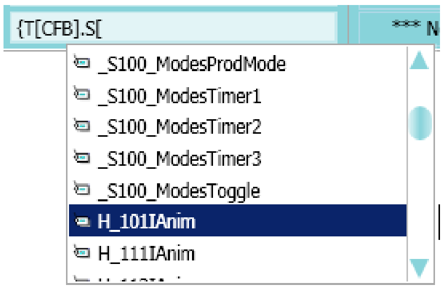
4. Repeat step 1 -3 for each device in the project. A DB variable must be added for the mode zones as well. This is done under DB1 in the S7 folder for the PLC cabinet. The name of the H_IAnim variable for the mode zone is H_[mode zone name]IAnim.
5. To be able to assign the DB variables to the HMI tags we need to update the symbol table first. To do this by go to General -> PLC cabinet and right click on symbol table and select Symbol table -> Refresh symbol table.

6. Now open a WinCC flexible project fragment under one of your devices.



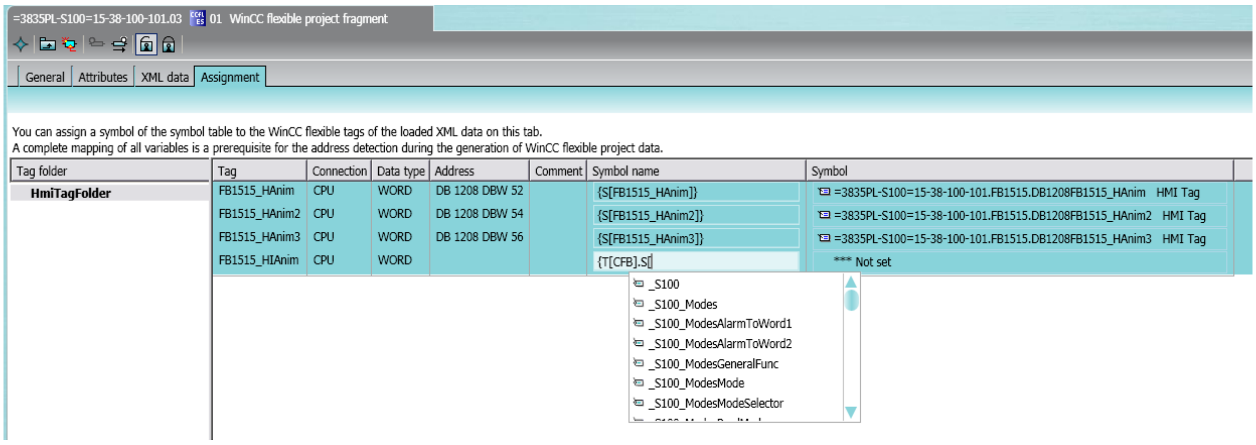
101 Roller2W2S-DDP
- S101M1QH Logical element, 2 inputs
- S101M1QM Logical element, 2 inputs
- S101M1SG1 Logical element, 2 inputs
- S101M1SG2 Logical element, 2 inputs
- S101M1SG8 Logical element, 2 inputs
- S101M1SG9 Logical element, 2 inputs
- 03 WinCC flexible
  - 01 WinCC flexible project fragment

7. Select the assignment tab. The H_Anim tags should already be connected. In the symbol name cell for the H_IAnim write the inline macro {T[CFB].S[*]} where * is the name of variable you want to connect.



{T[CFB].S[*** N

- _S100_ModesProdMode
- _S100_ModesTimer1
- _S100_ModesTimer2
- _S100_ModesTimer3
- _S100_ModesToggle
- H_101IAnim
- H_111IAnim



=3835PL-S100=15-38-100-101.03 01 WinCC flexible project fragment

General | Attributes | XML data | Assignment

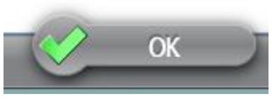
You can assign a symbol of the symbol table to the WinCC flexible tags of the loaded XML data on this tab.
A complete mapping of all variables is a prerequisite for the address detection during the generation of WinCC flexible project data.

Tag folder: HmiTagFolder

Tag	Connection	Data type	Address	Comment	Symbol name	Symbol
FB1515_HAnim	CPU	WORD	DB 1208 DBW 52		{S[FB1515_HAnim]}	=3835PL-S100=15-38-100-101.FB1515.DB1208FB1515_HAnim HMI Tag
FB1515_HAnim2	CPU	WORD	DB 1208 DBW 54		{S[FB1515_HAnim2]}	=3835PL-S100=15-38-100-101.FB1515.DB1208FB1515_HAnim2 HMI Tag
FB1515_HAnim3	CPU	WORD	DB 1208 DBW 56		{S[FB1515_HAnim3]}	=3835PL-S100=15-38-100-101.FB1515.DB1208FB1515_HAnim3 HMI Tag
FB1515_HIAnim	CPU	WORD			{T[CFB].S[	*** Not set

- _S100
- _S100_Modes
- _S100_ModesAlarmToWord1
- _S100_ModesAlarmToWord2
- _S100_ModesGeneralFunc
- _S100_ModesMode
- _S100_ModesModeSelector

8. Press OK to save.

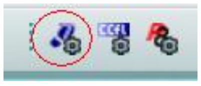


9. Repeat step 6 – 8 for all your devices and mode zones.

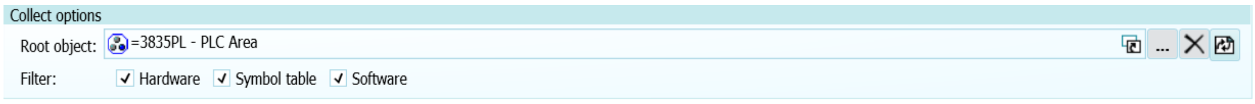
A.10 Generating code to STEP 7

To generate STEP 7 code requires that all previous steps are completed.

1. Open the STEP 7 generator: The STEP 7 generator can be found either in the symbol bar or in the menu bar under Generators.



2. Drag the PLC area object from the navigation tree to Root object. Automation Designer will then collect all FB:s, FC:s and DB:s that exists in the project.

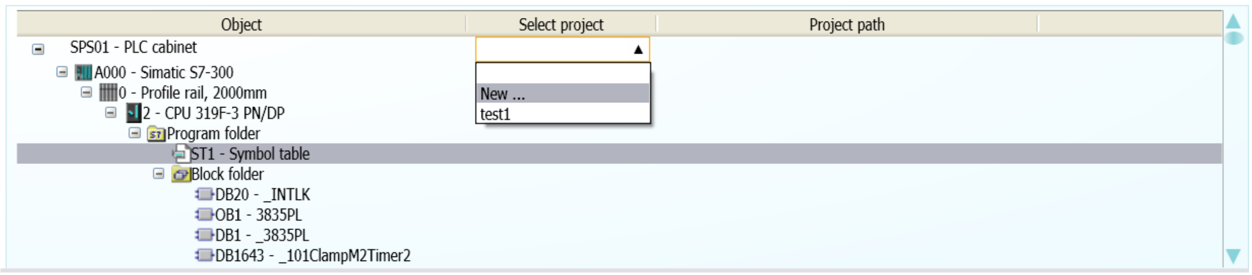


Collect options

Root object: =3835PL - PLC Area

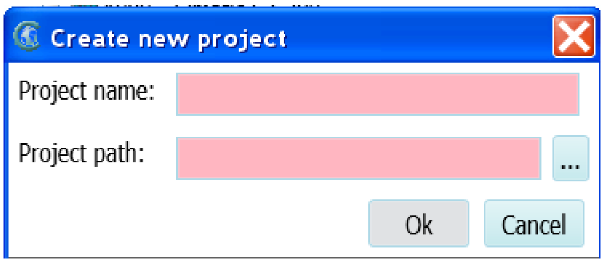
Filter: ✓ Hardware ✓ Symbol table ✓ Software

3. Open the drop down menu under Select project and click on New.



Object	Select project	Project path
SPS01 - PLC cabinet		
A000 - Simatic S7-300		
0 - Profile rail, 2000mm		
2 - CPU 319F-3 PN/DP		
Program folder		
ST1 - Symbol table		
Block folder		
DB20 - _INTLK		
OB1 - 3835PL		
DB1 - _3835PL		
DB1643 - _101ClampM2Timer2		

4. Select a name for the project and choose where you want to save your exported project.

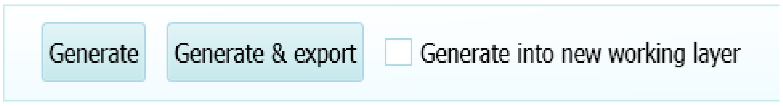


Create new project

Project name:

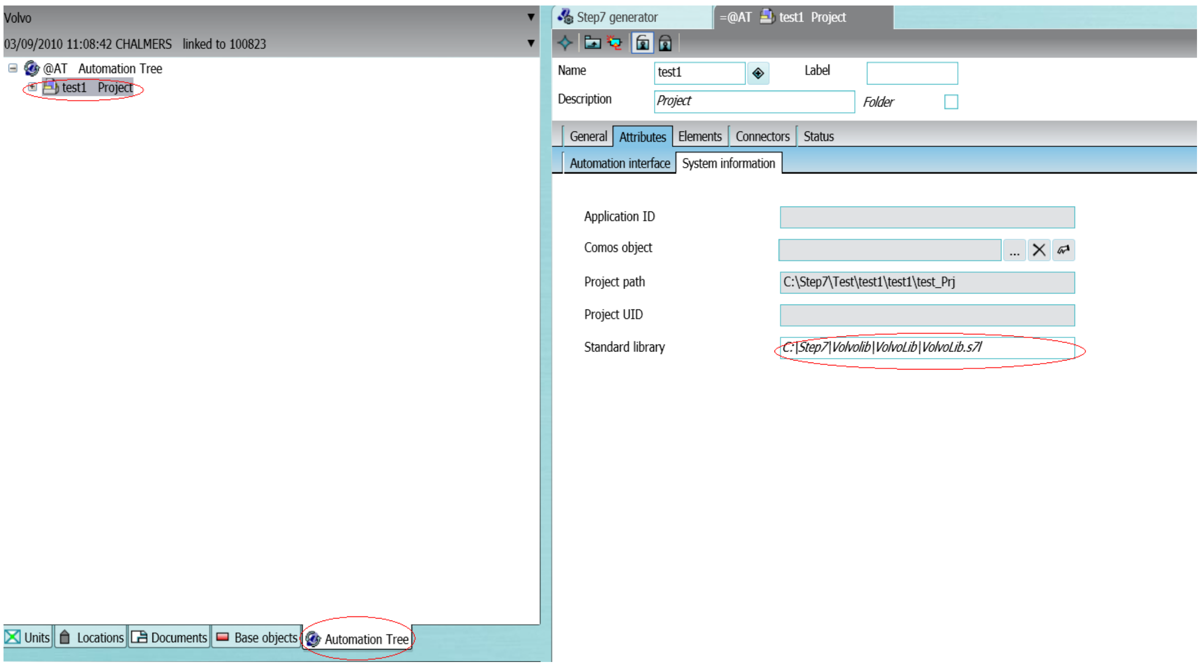
Project path:

5. Press Generate. This will make Automation Designer execute all inline macros in the FB:s, FC:s and DB:s that specifies the project specific names contained in the STEP 7 source code. Automation Designer will create an Automation Tree which contains the source code which now is ready for export.



6. Before exporting the project open the properties for the generated project in the Automation Tree. Make sure that the path for the standard library points to the correct location. Otherwise Automation Designer won't be able to import the library FB:s, FC:s and

DB:s.


Volvo

03/09/2010 11:08:42 CHALMERS linked to 100823

@AT Automation Tree
test1 Project

Units | Locations | Documents | Base objects | Automation Tree

Step7 generator | =@AT test1 Project

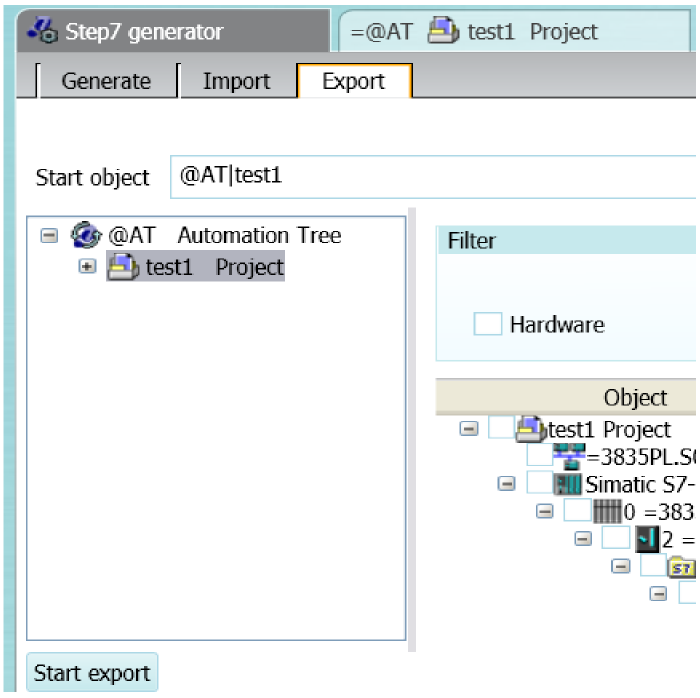
Name	test1	Label	
Description	*Project*	*Folder*	☐

General | Attributes | Elements | Connectors | Status

Automation interface | System information

Application ID	
Comos object	
Project path	C:\Step7\Test\test1\test1\test_Prj
Project UID	
Standard library	*C:\Step7\Volvolib\VolvoLib\VolvoLib.s7l*

7. Open the STEP 7 generator and select the export tab. Drag the generated project to Start object. If this is the first time a project has been generated the project might not show up in the automation tree. In this case just close and reopen the STEP 7 generator.


Step7 generator | =@AT test1 Project

Generate | Import | Export

Start object @AT|test1

@AT Automation Tree
test1 Project

Filter

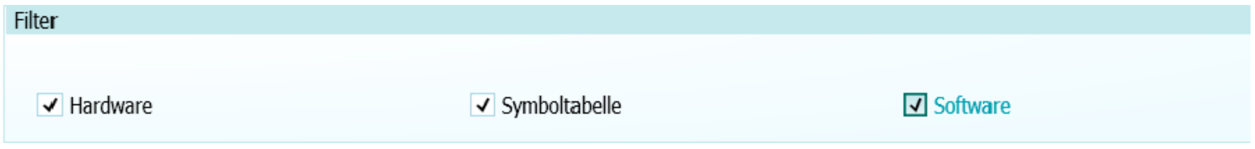
☐ Hardware

Object

test1 Project
=3835PL.S
Simatic S7-
0 =383
2 =

Start export

8. Check all checkboxes. This will make sure all necessary objects will be exported.

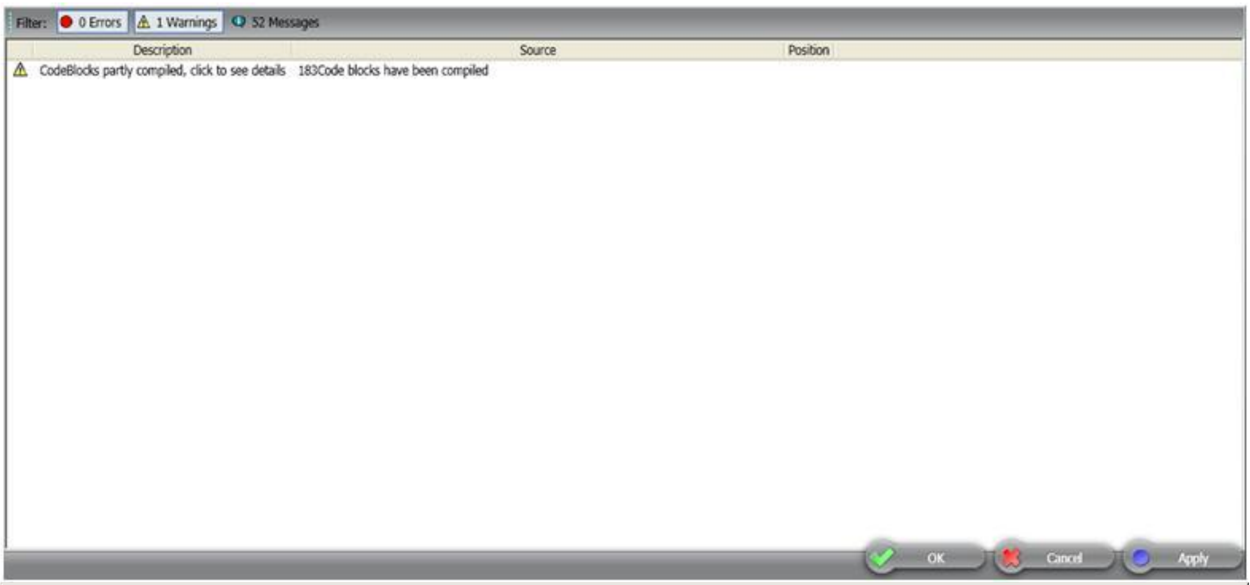

Filter

✓ Hardware ✓ Symboltabelle ✓ Software

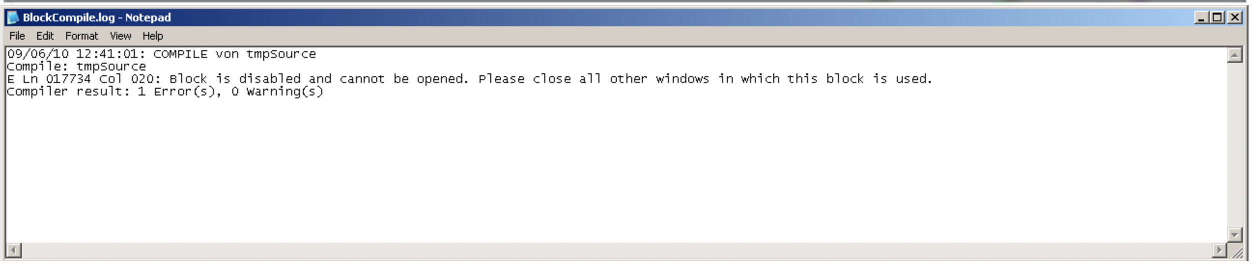
9. Press Start export to compile the source code into STEP 7

Start export

10. If there are no error messages a new project that is possible to open in STEP 7 will have been created. If you get the warning message “CodeBlocks partly compiled” it means that STEP 7 failed to compile the code. Double click on the error message will display the compilation log from STEP 7 which describes why the compilation failed. The error in the compilation log below is caused because OB1 is used by another process at the same time. To fix this problem close both Automation Designer and STEP 7. Reopen STEP 7 and follow the steps below.



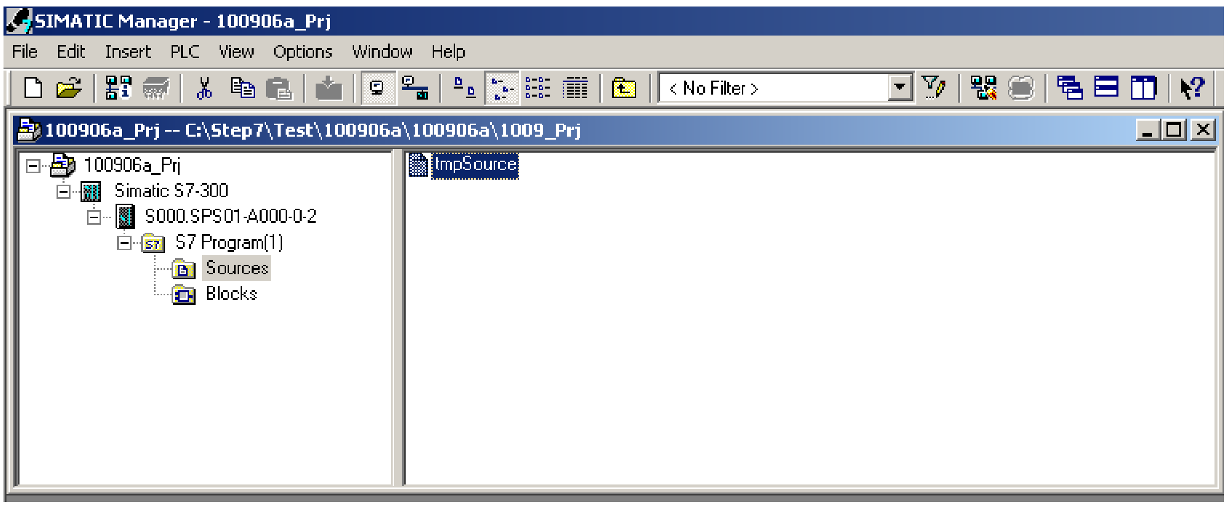
Description	Source	Position
CodeBlocks partly compiled, click to see details	183Code blocks have been compiled	



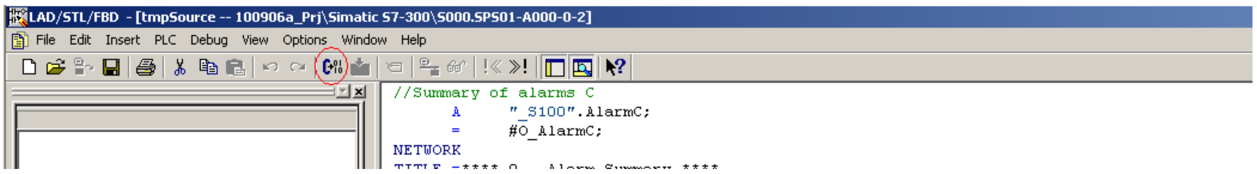
```
09/06/10 12:41:01: COMPILE von tmpSource
Compile: tmpSource
E Ln 017734 Col 020: Block is disabled and cannot be opened. Please close all other windows in which this block is used.
Compiler result: 1 Error(s), 0 Warning(s)
```

11. Select File->Open and choose your project

12. Open tmpSource in the source folder


SIMATIC Manager - 100906a_Prj
File Edit Insert PLC View Options Window Help
< No Filter >
100906a_Prj -- C:\Step7\Test\100906a\100906a\1009_Prj
100906a_Prj
Simatic S7-300
S000.SPS01-A000-0-2
S7 Program(1)
Sources
Blocks
tmpSource

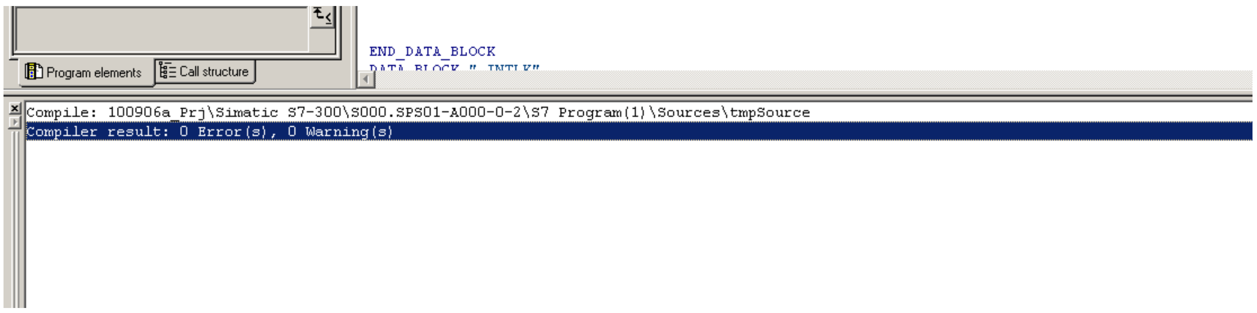
13. Press the compile button.



LAD/STL/FBD - [tmpSource -- 100906a_Prj\Simatic S7-300\S000.SPS01-A000-0-2]

```
//Summary of alarms C
      A      "_S100".AlarmC;
      =      #O_AlarmC;
NETWORK
```

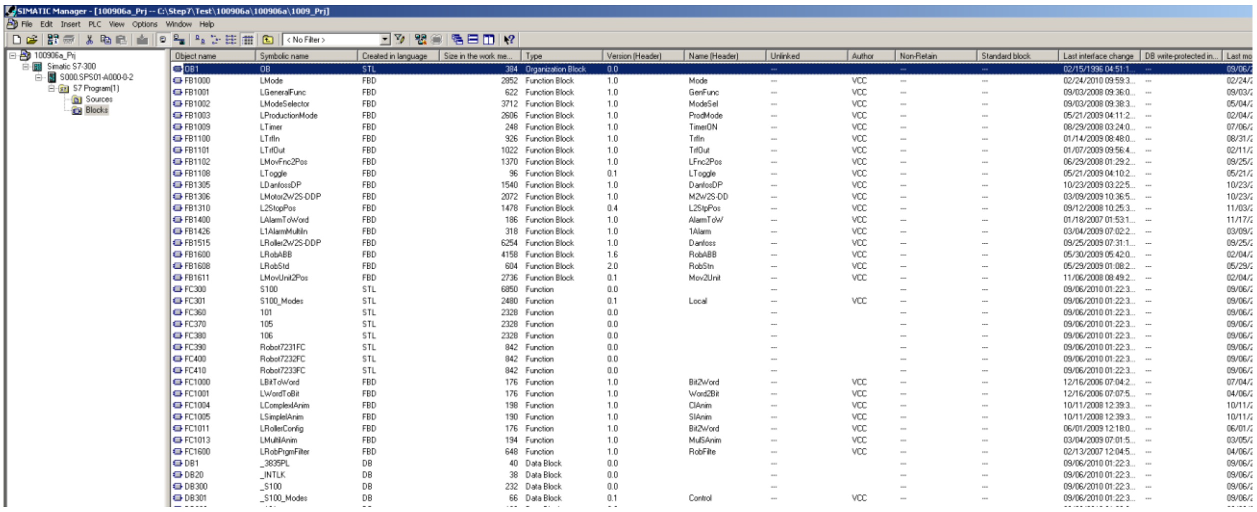
14. The log should now say 0 Errors, 0 Warnings. In case any errors remain double click the error message to display where in the code the error occurs.



```
END_DATA_BLOCK
```

```
Compile: 100906a_Prj\Simatic S7-300\S000.SPS01-A000-0-2\S7 Program(1)\Sources\tmpSource
Compiler result: 0 Error(s), 0 Warning(s)
```

15. Go back to the project view and choose the "Blocks" folder. Your project should now look something like in the picture below.



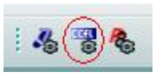
SIMATIC Manager - [100906a_Prj -- C:\Step7\Test\100906a\100906a\1009_Prj]

Object name	Symbolic name	Created in language	Size in the work me...	Type	Version (Header)	Name (Header)	Unlinked	Author	Non-Retain	Standard block	Last interface change	DB write-protected in...	Last mo
OB1	OB	STL	384	Organization Block	0.0		---		---	---	02/15/1996 04:51:1...	---	09/06/2
FB1000	LMode	FBD	2852	Function Block	1.0	Mode	---	VCC	---	---	02/24/2010 09:59:3...	---	02/24/2
FB1001	LGeneralFunc	FBD	622	Function Block	1.0	GenFunc	---	VCC	---	---	09/03/2008 09:36:0...	---	09/03/2
FB1002	LModeSelector	FBD	3712	Function Block	1.0	ModeSel	---	VCC	---	---	09/03/2008 09:38:3...	---	05/04/2
FB1003	LProductionMode	FBD	2606	Function Block	1.0	ProdMode	---	VCC	---	---	05/21/2009 04:11:2...	---	02/04/2
FB1009	LTimer	FBD	248	Function Block	1.0	TimerON	---	VCC	---	---	08/29/2008 03:24:0...	---	07/06/2
FB1100	LTrfIn	FBD	926	Function Block	1.0	TrfIn	---	VCC	---	---	01/14/2009 08:48:0...	---	08/31/2
FB1101	LTrfOut	FBD	1022	Function Block	1.0	TrfOut	---	VCC	---	---	01/07/2009 09:56:4...	---	02/11/2
FB1102	LMovFnc2Pos	FBD	1370	Function Block	1.0	LFnc2Pos	---	VCC	---	---	06/29/2008 01:29:2...	---	09/25/2
FB1108	LToggle	FBD	96	Function Block	0.1	LToggle	---	VCC	---	---	05/21/2009 04:10:2...	---	05/21/2
FB1305	LDanfossDP	FBD	1540	Function Block	1.0	DanfosDP	---	VCC	---	---	10/23/2009 03:22:5...	---	10/23/2
FB1306	LMotor2W2S-DDP	FBD	2072	Function Block	1.0	M2W2S-DD	---	VCC	---	---	03/09/2009 10:36:5...	---	10/23/2
FB1310	L2StopPos	FBD	1478	Function Block	0.4	L2StpPos	---	VCC	---	---	09/12/2008 10:25:3...	---	11/03/2
FB1400	LAlarmToWord	FBD	186	Function Block	1.0	AlarmToW	---	VCC	---	---	01/18/2007 01:53:1...	---	11/17/2
FB1426	L1AlarmMultIn	FBD	318	Function Block	1.0	1Alarm	---	VCC	---	---	03/04/2009 07:02:2...	---	03/09/2
FB1515	LRoller2W2S-DDP	FBD	6254	Function Block	1.0	Danfoss	---	VCC	---	---	09/25/2009 07:31:1...	---	09/25/2
FB1600	LRobABB	FBD	4158	Function Block	1.6	RobABB	---	VCC	---	---	05/30/2009 05:42:0...	---	02/04/2
FB1608	LRobStd	FBD	604	Function Block	2.0	RobStn	---	VCC	---	---	05/29/2009 01:08:2...	---	05/29/2
FB1611	LMovUnit2Pos	FBD	2736	Function Block	0.1	Mov2Unit	---	VCC	---	---	11/06/2008 08:49:2...	---	02/04/2
FC300	S100	STL	6850	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC301	S100_Modes	STL	2480	Function	0.1	Local	---	VCC	---	---	09/06/2010 01:22:3...	---	09/06/2
FC360	101	STL	2328	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC370	105	STL	2328	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC380	106	STL	2328	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC390	Robot7231FC	STL	842	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC400	Robot7232FC	STL	842	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC410	Robot7233FC	STL	842	Function	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
FC1000	LBitToWord	FBD	176	Function	1.0	Bit2Word	---	VCC	---	---	12/16/2006 07:04:2...	---	07/04/2
FC1001	LWordToBit	FBD	176	Function	1.0	Word2Bit	---	VCC	---	---	12/16/2006 07:07:5...	---	04/06/2
FC1004	LComplexAnim	FBD	198	Function	1.0	ClAnim	---	VCC	---	---	10/11/2008 12:39:3...	---	10/11/2
FC1005	LSimpleAnim	FBD	190	Function	1.0	SIAnim	---	VCC	---	---	10/11/2008 12:39:3...	---	10/11/2
FC1011	LRollerConfig	FBD	176	Function	1.0	Bit2Word	---	VCC	---	---	06/01/2009 12:18:0...	---	06/01/2
FC1013	LMultiAnim	FBD	194	Function	1.0	MulSAnim	---	VCC	---	---	03/04/2009 07:01:5...	---	03/05/2
FC1600	LRobPrgmFilter	FBD	648	Function	1.0	RobFilte	---	VCC	---	---	02/13/2007 12:04:5...	---	04/06/2
DB1	_3835PL	DB	40	Data Block	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
DB20	_INTLK	DB	38	Data Block	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
DB300	_S100	DB	232	Data Block	0.0		---		---	---	09/06/2010 01:22:3...	---	09/06/2
DB301	_S100_Modes	DB	66	Data Block	0.1	Control	---	VCC	---	---	09/06/2010 01:22:3...	---	09/06/2

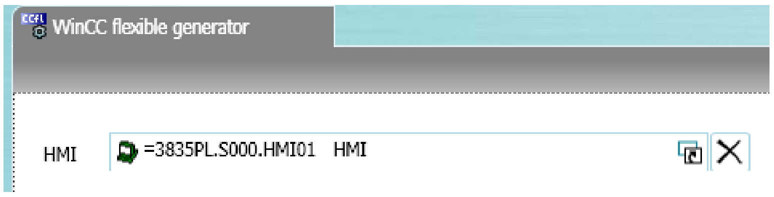
A.11 Generating pictures to WinCC flexible

To be able to generate WinCC flexible screens a successful generation of the PLC code must be made first.

1. Open the WinCC flexible generator: The WinCC flexible generator can be found either in the symbol bar or in the menu bar under Generators.



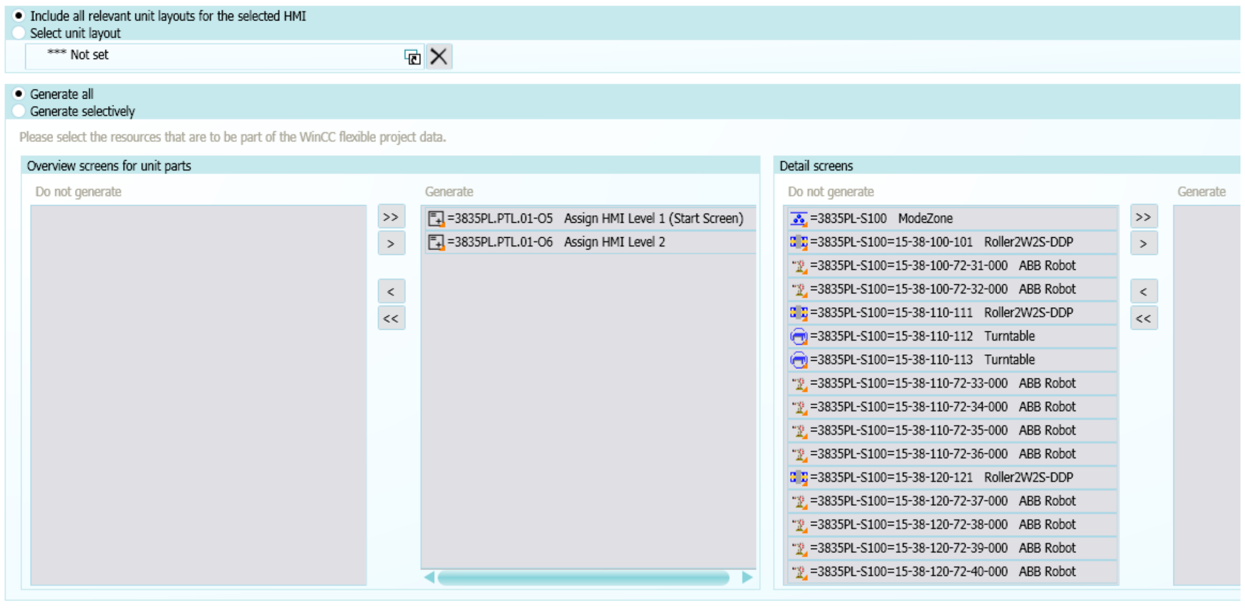
2. Drag the HMI object from the General folder to the HMI field.



WinCC flexible generator

HMI =3835PL.S000.HMI01 HMI

3. A list with all objects that have a WinCC flexible screen will appear. Generate all should be selected.



[✓ Include all relevant unit layouts for the selected HMI]
Select unit layout
*** Not set

[✓ Generate all]
Generate selectively

Please select the resources that are to be part of the WinCC flexible project data.

Overview screens for unit parts

Do not generate

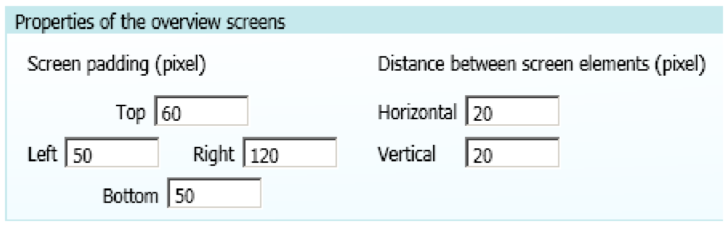
Generate
=3835PL.PTL.01-O5 Assign HMI Level 1 (Start Screen)
=3835PL.PTL.01-O6 Assign HMI Level 2

Detail screens

Do not generate
=3835PL-S100 ModeZone
=3835PL-S100=15-38-100-101 Roller2W2S-DDP
=3835PL-S100=15-38-100-72-31-000 ABB Robot
=3835PL-S100=15-38-100-72-32-000 ABB Robot
=3835PL-S100=15-38-110-111 Roller2W2S-DDP
=3835PL-S100=15-38-110-112 Turntable
=3835PL-S100=15-38-110-113 Turntable
=3835PL-S100=15-38-110-72-33-000 ABB Robot
=3835PL-S100=15-38-110-72-34-000 ABB Robot
=3835PL-S100=15-38-110-72-35-000 ABB Robot
=3835PL-S100=15-38-110-72-36-000 ABB Robot
=3835PL-S100=15-38-120-121 Roller2W2S-DDP
=3835PL-S100=15-38-120-72-37-000 ABB Robot
=3835PL-S100=15-38-120-72-38-000 ABB Robot
=3835PL-S100=15-38-120-72-39-000 ABB Robot
=3835PL-S100=15-38-120-72-40-000 ABB Robot

Generate

4. The next step is to decide the generated images distance for the edges and each other.

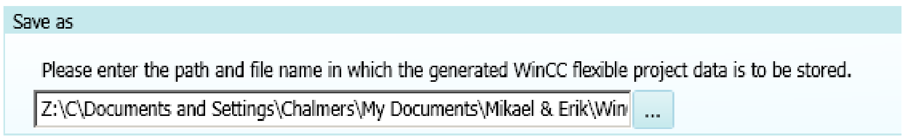


Properties of the overview screens

Screen padding (pixel)	
Top	60
Left	50
Right	120
Bottom	50

Distance between screen elements (pixel)	
Horizontal	20
Vertical	20

5. Select a destination to save the exported XML-file.



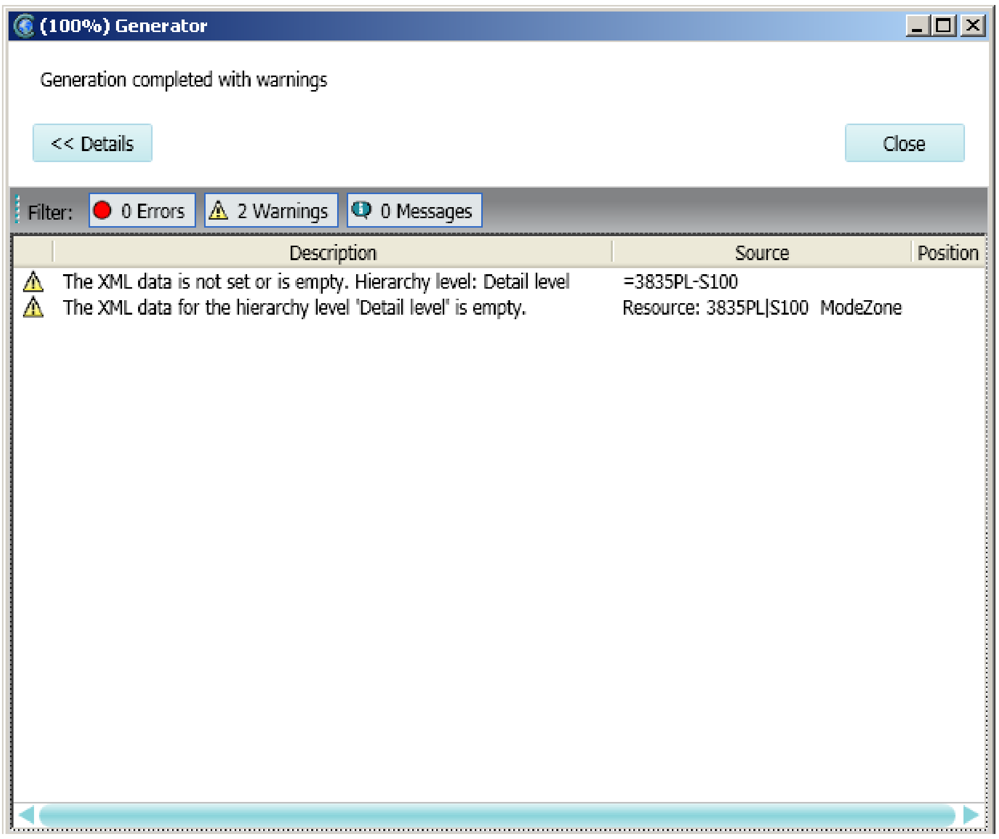
Save as

Please enter the path and file name in which the generated WinCC flexible project data is to be stored.

Z:\C\Documents and Settings\Chalmers\My Documents\Mikael & Erik\Win

6. Press start to generate the HMI screens.

7. You will get a warning message that the ModeZone object is missing a detail level screen. Ignore this warning.



Generation completed with warnings

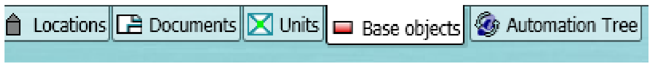
Description	Source	Position
⚠ The XML data is not set or is empty. Hierarchy level: Detail level	=3835PL-S100	
⚠ The XML data for the hierarchy level 'Detail level' is empty.	Resource: 3835PL|S100 ModeZone	

Appendix B

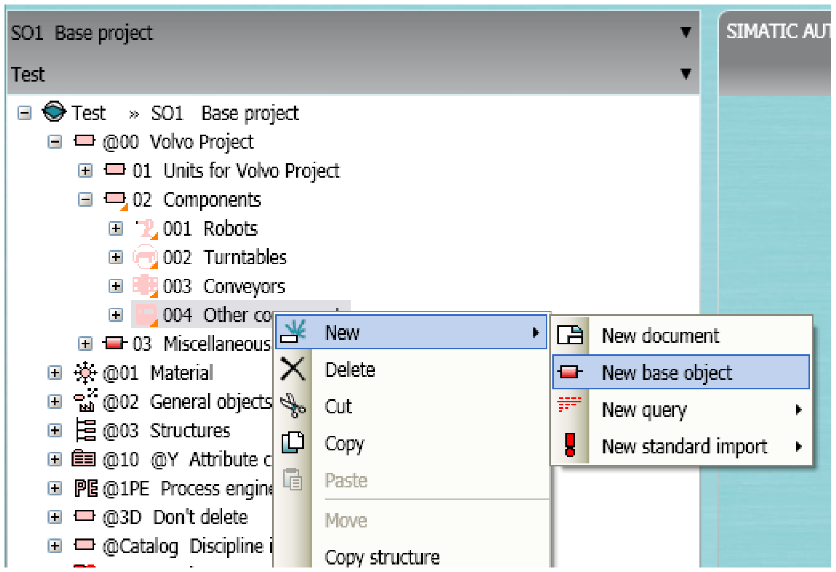
This appendix will describe how you can create a base object and a template inside Automation Designer. Remember that this can only be done if you are in the base project of Automation Designer, not the engineering project.

B.1 Creating a base object.

1. Open the base project and go to the base objects tab in the navigation tree.

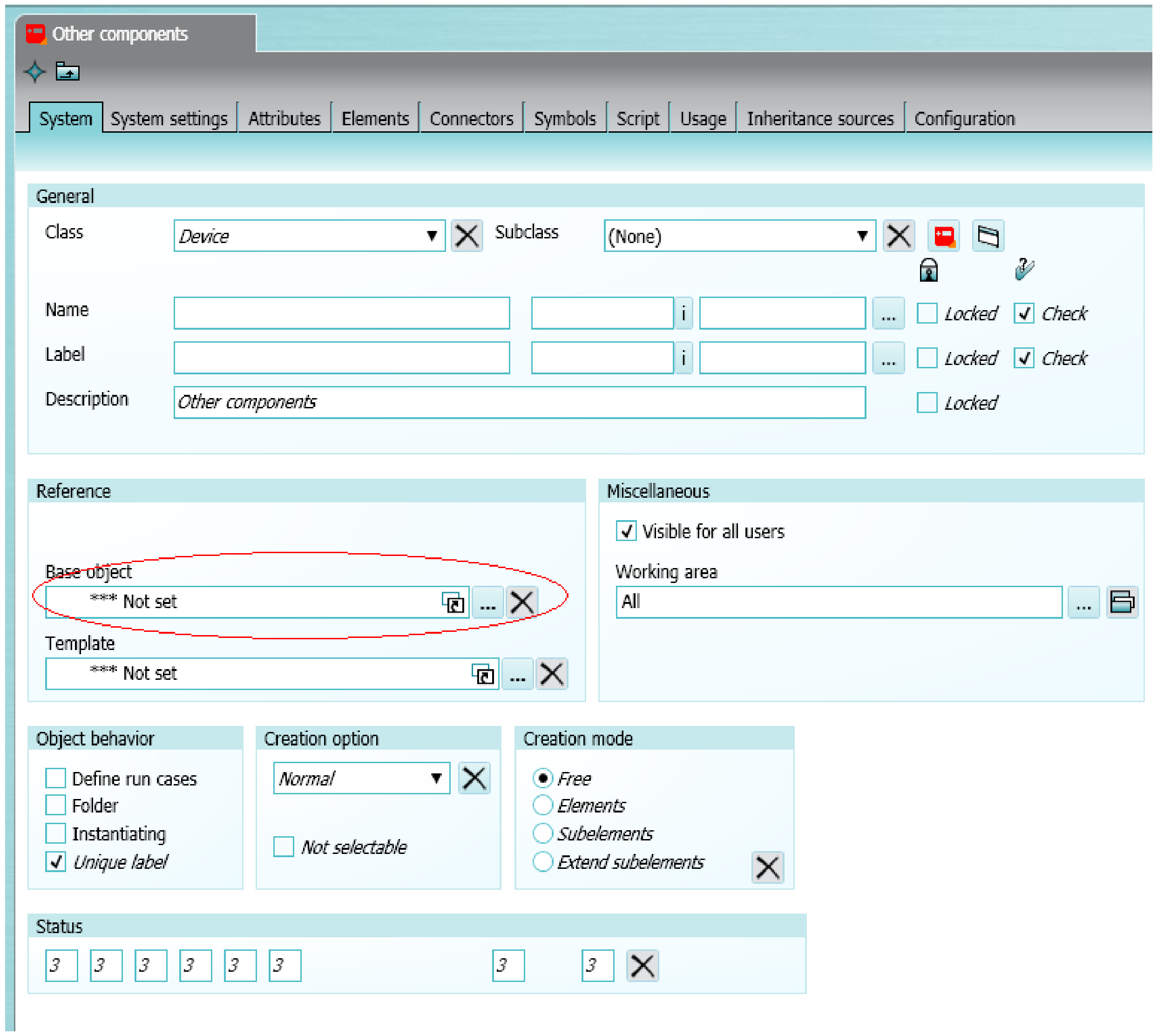


2. Navigate to the position in the navigation tree where you want to create your new base object. Right click and select New -> New base object.



3. The newly created base object will inherit its unset properties from its parent base object. It is also possible for the new object to inherit its properties from other objects in the base project.

This is done by assigning the desired base object as a reference.



Other components

System | System settings | Attributes | Elements | Connectors | Symbols | Script | Usage | Inheritance sources | Configuration

General

Class: *Device* — Subclass: (None)

Name: — Locked [✓ *Check*]

Label: — Locked [✓ *Check*]

Description: *Other components* — Locked

Reference

Base object: *** Not set

Template: *** Not set

Miscellaneous

[✓ Visible for all users]

Working area: All

Object behavior

Define run cases
Folder
Instantiating
[✓ *Unique label*]

Creation option

Normal

Not selectable

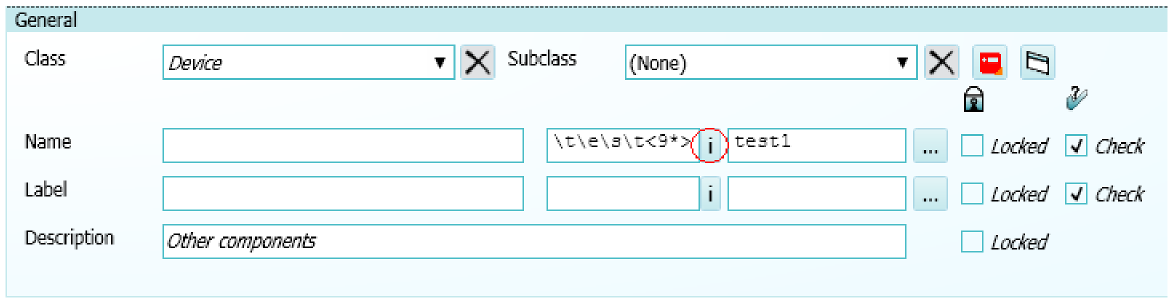
Creation mode

[✓ *Free*] *Elements* *Subelements* *Extend subelements*

Status

3 *3* *3* *3* *3* *3* *3* *3*

4. The naming conventions for the base are set in the system tab. To access the syntax options for naming rules press the “I” button. It is also possible to set names by scripts in the script tab.



General

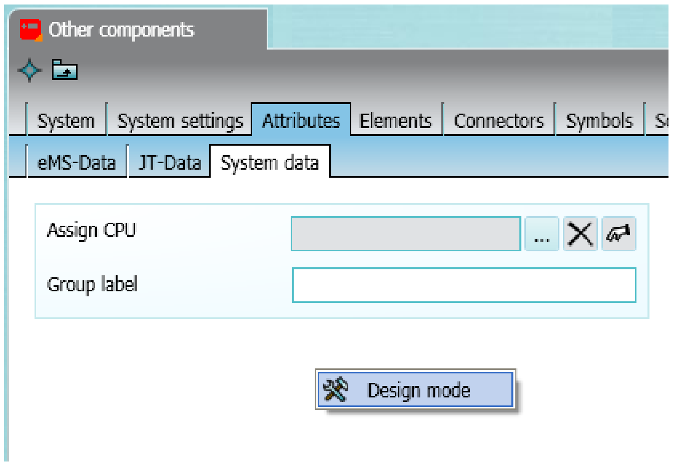
Class: *Device* — Subclass: (None)

Name: `\t\e\s\t<9*>` i `test1` — Locked [✓ *Check*]

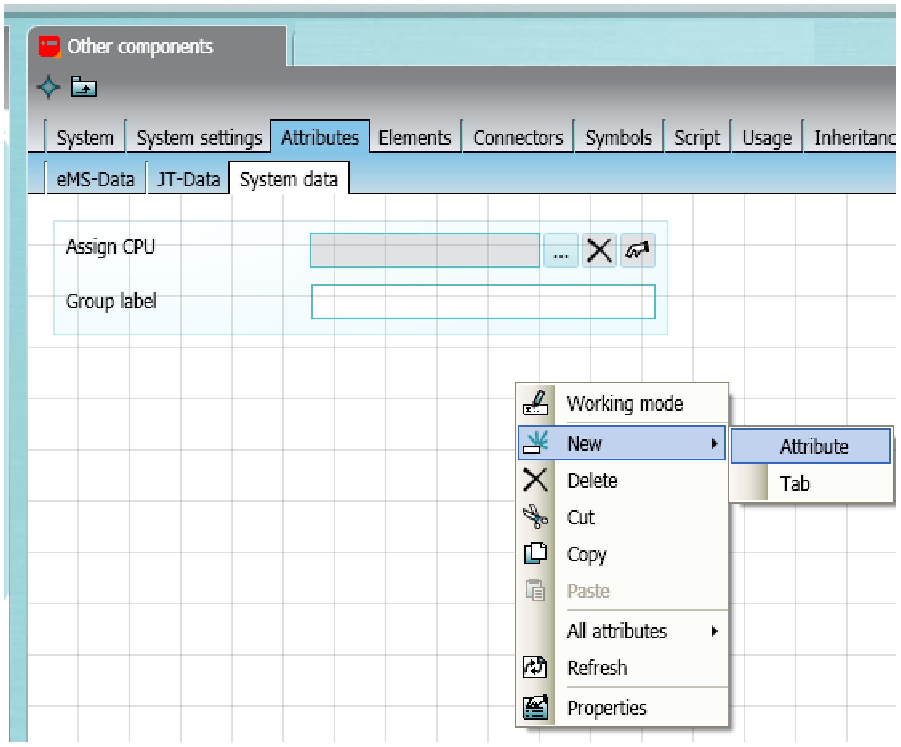
Label: — Locked [✓ *Check*]

Description: *Other components* — Locked

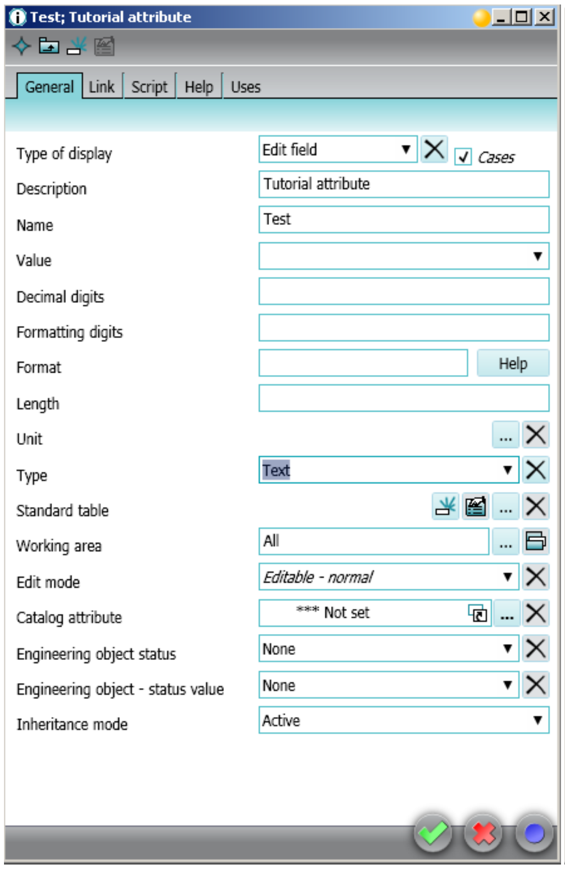
5. To add new attributes to your base object go to the attributes tab. Right click anywhere in the window and select Design mode.



6. To add a new attribute right click and select New -> Attribute



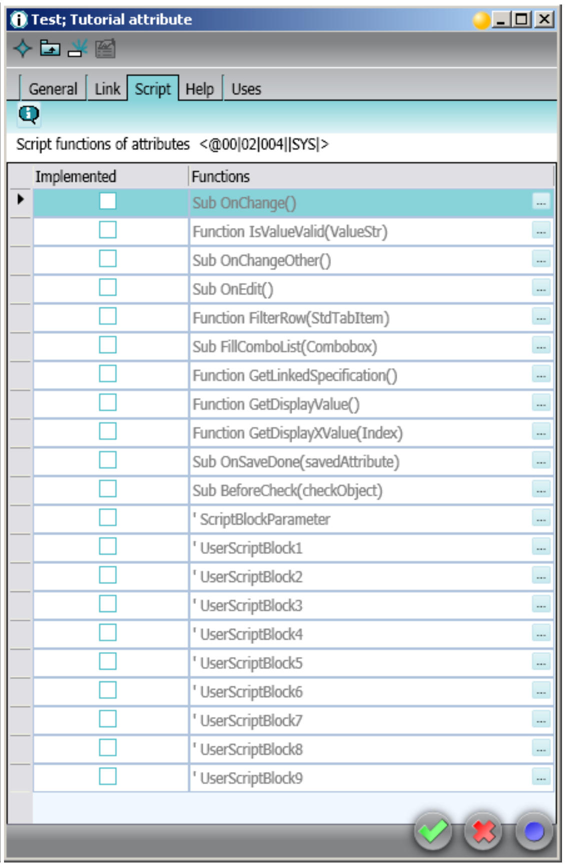
7. In the new window that appears you can fill in the details about the attribute. In the scripts tab scripts can be added for manipulating the attribute.



Test; Tutorial attribute

General | Link | Script | Help | Uses

Type of display	Edit field ✓ Cases
Description	Tutorial attribute
Name	Test
Value	
Decimal digits	
Formatting digits	
Format	
Length	
Unit	
Type	Text
Standard table	
Working area	All
Edit mode	*Editable - normal*
Catalog attribute	*** Not set
Engineering object status	None
Engineering object - status value	None
Inheritance mode	Active



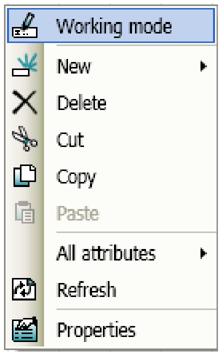
Test; Tutorial attribute

General | Link | Script | Help | Uses

Script functions of attributes <@00|02|004||SYS|>

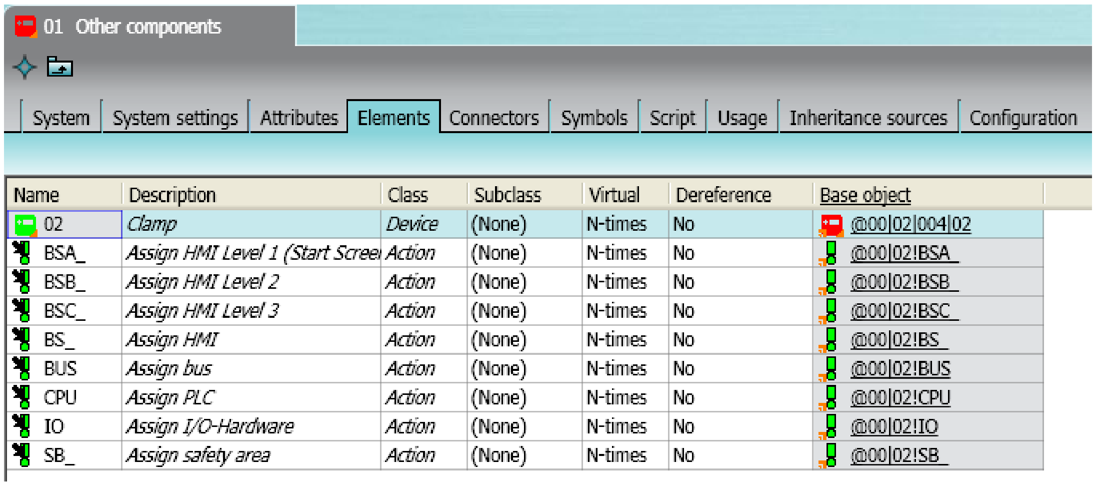
Implemented	Functions
☐	Sub OnChange()
☐	Function IsValueValid(ValueStr)
☐	Sub OnChangeOther()
☐	Sub OnEdit()
☐	Function FilterRow(StdTabItem)
☐	Sub FillComboList(Combobox)
☐	Function GetLinkedSpecification()
☐	Function GetDisplayValue()
☐	Function GetDisplayXValue(Index)
☐	Sub OnSaveDone(savedAttribute)
☐	Sub BeforeCheck(checkObject)
☐	' ScriptBlockParameter
☐	' UserScriptBlock1
☐	' UserScriptBlock2
☐	' UserScriptBlock3
☐	' UserScriptBlock4
☐	' UserScriptBlock5
☐	' UserScriptBlock6
☐	' UserScriptBlock7
☐	' UserScriptBlock8
☐	' UserScriptBlock9

8. Press confirm to save the changes.
9. Right click and select working mode to leave the design mode.



10. Open the Elements tab.
11. In the element tab we can add actions that we want to be able to perform on the object. The actions are later performed by eBlocks in the templates. We can also add devices that we want to have available when right clicking on the object. The items with a black arrow in the name

column are inherited from a parent object.

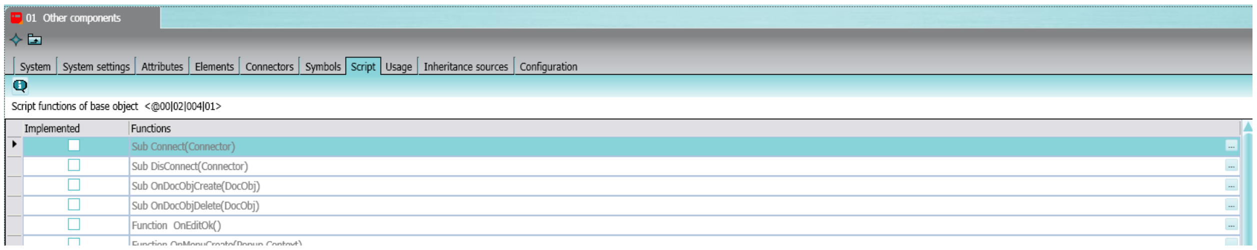


01 Other components

System | System settings | Attributes | Elements | Connectors | Symbols | Script | Usage | Inheritance sources | Configuration

Name	Description	Class	Subclass	Virtual	Dereference	Base object
02	*Clamp*	*Device*	(None)	N-times	No	@00|02|004|02
BSA_	*Assign HMI Level 1 (Start Scree*	*Action*	(None)	N-times	No	@00|02!BSA_
BSB_	*Assign HMI Level 2*	*Action*	(None)	N-times	No	@00|02!BSB_
BSC_	*Assign HMI Level 3*	*Action*	(None)	N-times	No	@00|02!BSC_
BS_	*Assign HMI*	*Action*	(None)	N-times	No	@00|02!BS_
BUS	*Assign bus*	*Action*	(None)	N-times	No	@00|02!BUS
CPU	*Assign PLC*	*Action*	(None)	N-times	No	@00|02!CPU
IO	*Assign I/O-Hardware*	*Action*	(None)	N-times	No	@00|02!IO
SB_	*Assign safety area*	*Action*	(None)	N-times	No	@00|02!SB_

12. In the script tab there is the possibility to add scripts for different occasions.



01 Other components

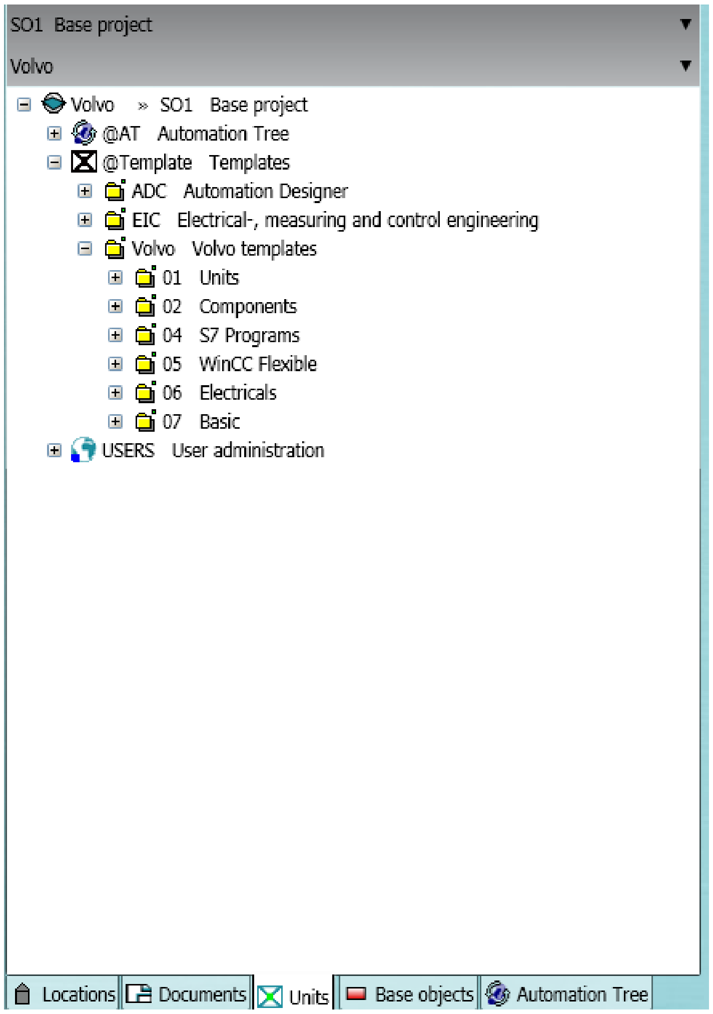
System | System settings | Attributes | Elements | Connectors | Symbols | Script | Usage | Inheritance sources | Configuration

Script functions of base object <@00|02|004|01>

Implemented	Functions
☐	Sub Connect(Connector)
☐	Sub DisConnect(Connector)
☐	Sub OnDocObjCreate(DocObj)
☐	Sub OnDocObjDelete(DocObj)
☐	Function OnEditOk()
☐	Function OnMenuCreate(Popup Context)

B.2 Creating a template.

1. In your base project, open the units tab in the navigation tree.

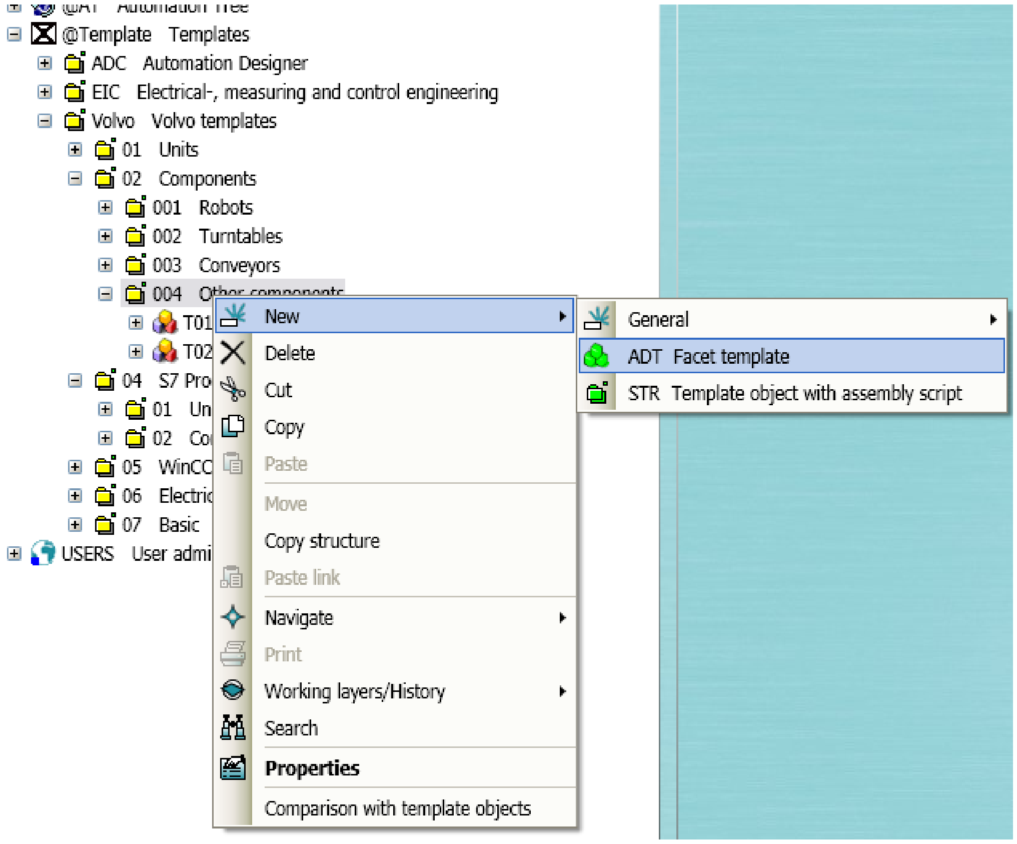


SO1 Base project

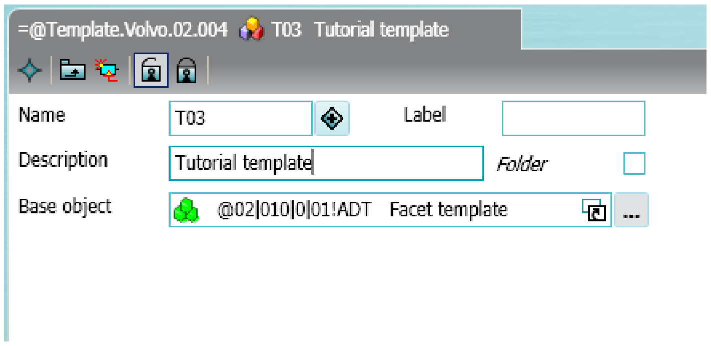
Volvo

```
⊟ Volvo » SO1 Base project
  ⊞ @AT Automation Tree
  ⊟ @Template Templates
    ⊞ ADC Automation Designer
    ⊞ EIC Electrical-, measuring and control engineering
    ⊟ Volvo Volvo templates
      ⊞ 01 Units
      ⊞ 02 Components
      ⊞ 04 S7 Programs
      ⊞ 05 WinCC Flexible
      ⊞ 06 Electricals
      ⊞ 07 Basic
  ⊞ USERS User administration
```

2. Navigate to where you want to create your new template. Right click and select New -> ADT Facet template.



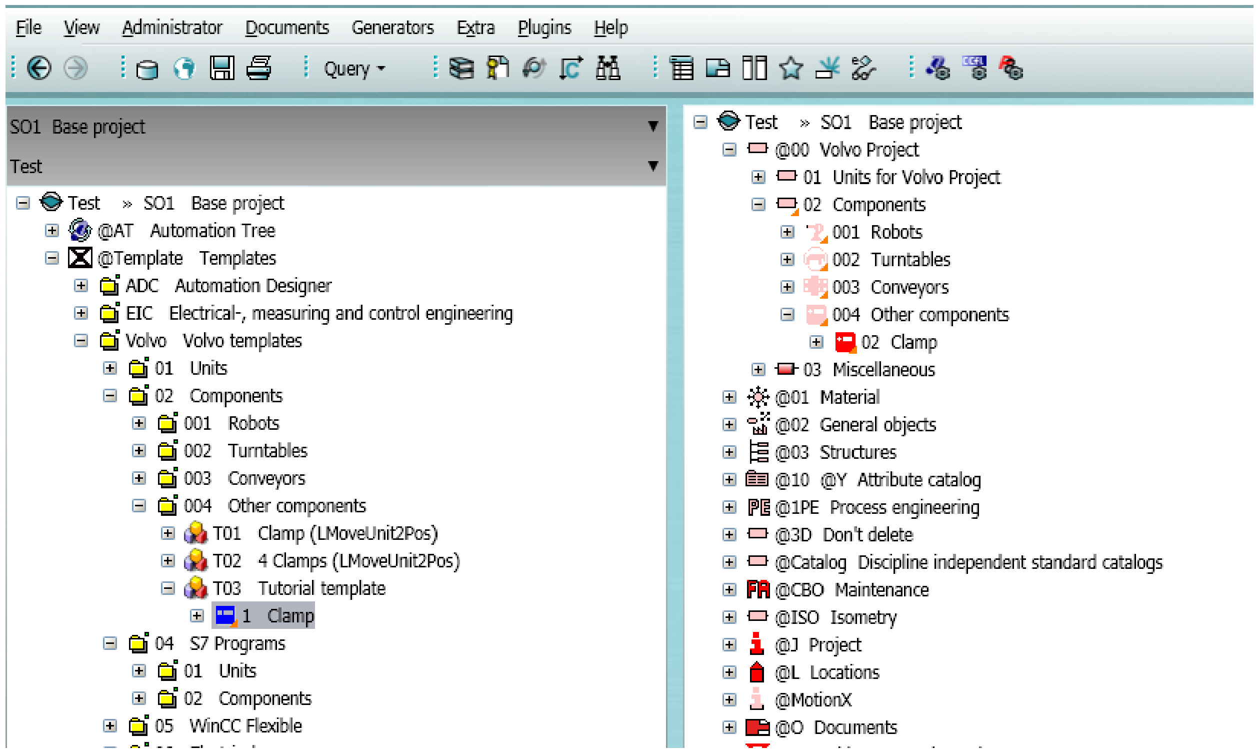
3. Open the properties for the newly created template and assign it a proper description.


=@Template.Volvo.02.004 T03 Tutorial template

Name: T03
Label:
Description: Tutorial template
Folder
Base object: @02|010|0|01!ADT Facet template

4. To add base objects to your template just copy or drag them to your new template in the navigation tree. All attributes for the added base object can be changed so they have a unique

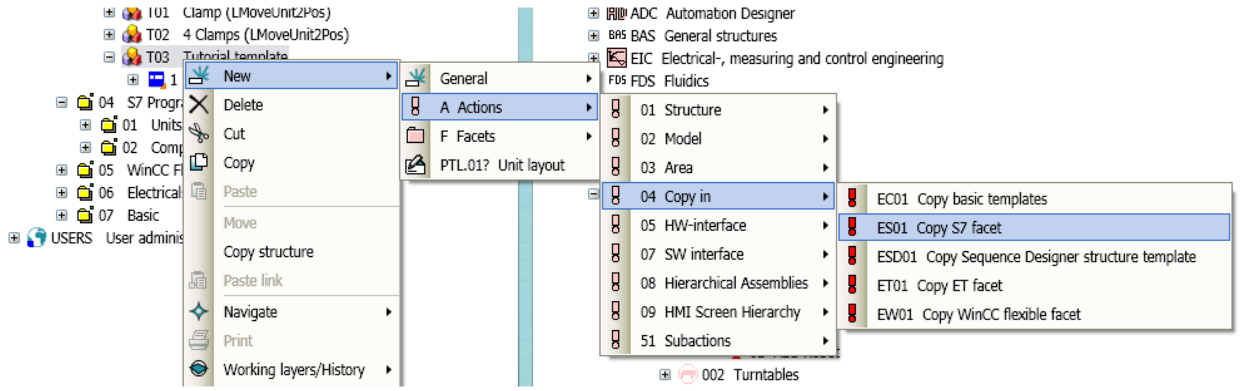
configuration for the current template.


File View Administrator Documents Generators Extra Plugins Help
Query
SO1 Base project
Test

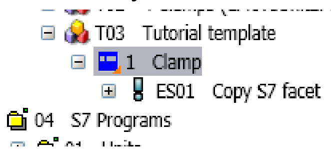
```
⊟ Test  » SO1  Base project
  ⊞ @AT  Automation Tree
  ⊟ @Template  Templates
    ⊞ ADC  Automation Designer
    ⊞ EIC  Electrical-, measuring and control engineering
    ⊟ Volvo  Volvo templates
      ⊞ 01  Units
      ⊟ 02  Components
        ⊞ 001  Robots
        ⊞ 002  Turntables
        ⊞ 003  Conveyors
        ⊟ 004  Other components
          ⊞ T01  Clamp (LMoveUnit2Pos)
          ⊞ T02  4 Clamps (LMoveUnit2Pos)
          ⊟ T03  Tutorial template
            ⊞ 1  Clamp
      ⊟ 04  S7 Programs
        ⊞ 01  Units
        ⊞ 02  Components
      ⊞ 05  WinCC Flexible
```

```
⊟ Test  » SO1  Base project
  ⊟ @00  Volvo Project
    ⊞ 01  Units for Volvo Project
    ⊟ 02  Components
      ⊞ 001  Robots
      ⊞ 002  Turntables
      ⊞ 003  Conveyors
      ⊟ 004  Other components
        ⊞ 02  Clamp
    ⊞ 03  Miscellaneous
  ⊞ @01  Material
  ⊞ @02  General objects
  ⊞ @03  Structures
  ⊞ @10  @Y  Attribute catalog
  ⊞ @1PE  Process engineering
  ⊞ @3D  Don't delete
  ⊞ @Catalog  Discipline independent standard catalogs
  ⊞ @CBO  Maintenance
  ⊞ @ISO  Isometry
  ⊞ @J  Project
  ⊞ @L  Locations
  ⊞ @MotionX
  ⊞ @O  Documents
```

5. eBlocks are used to perform actions in a template. A list of available eBlocks can be found by right clicking on the template object and select New -> Actions.



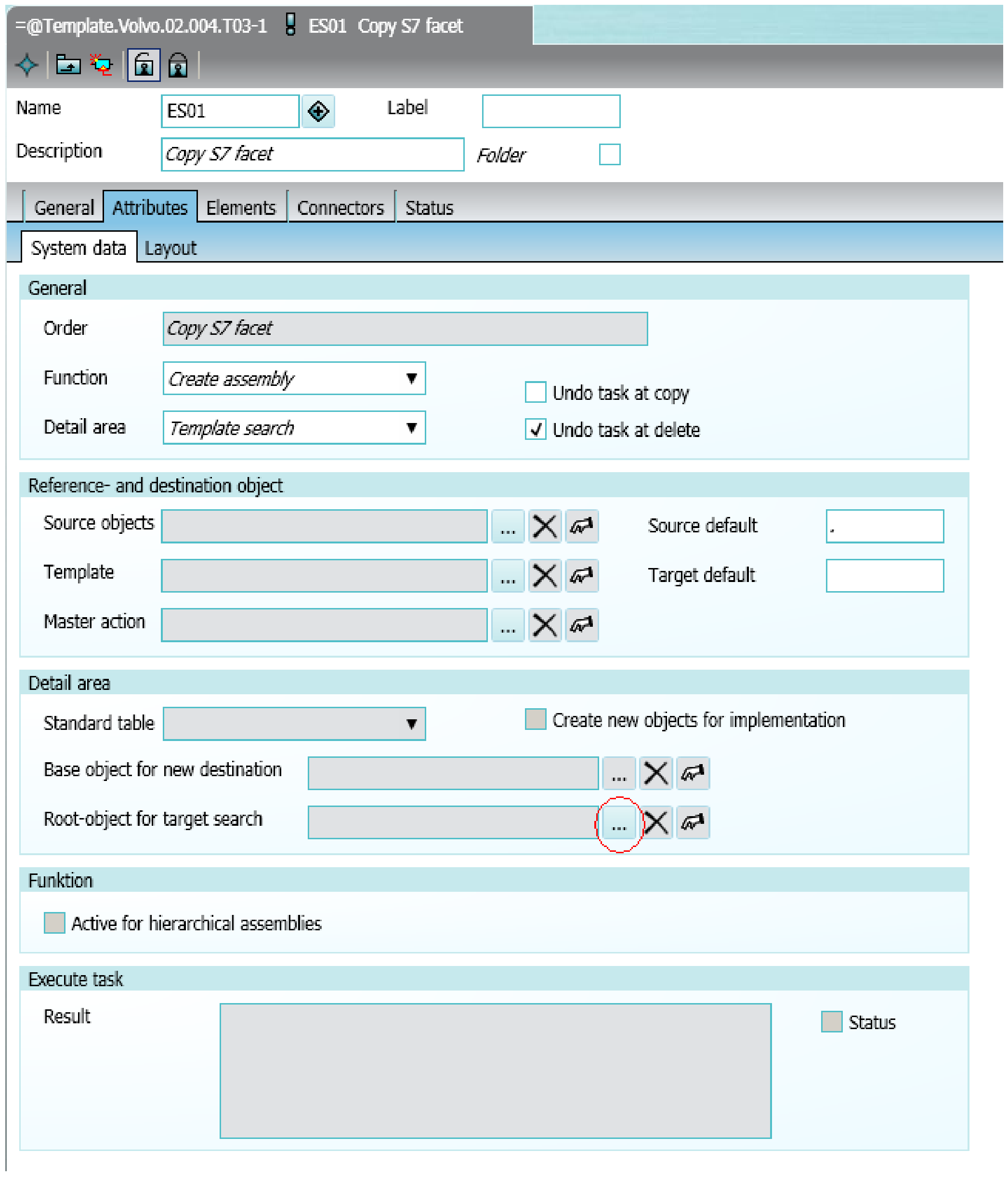
6. After adding a eBlock it must be moved so it is below the resource object in the hierarchy. To move an objet hold Ctrl and drag the object to its new position.



```
⊟ T03  Tutorial template
  ⊟ 1  Clamp
    ⊞ ES01  Copy S7 facet
04  S7 Programs
```

7. Open the properties for the eBlock.
8. An eBlock needs a source object. By default this will be the parent object. It is also possible to set the target object. In the example below we want to add a folder with S7-code. To assign

target template for this press the button with three dots.



=@Template.Volvo.02.004.T03-1 ES01 Copy S7 facet

Name	ES01	Label	
Description	*Copy S7 facet*	*Folder*	☐

General

Order	*Copy S7 facet*	
Function	*Create assembly*	☐ Undo task at copy
Detail area	*Template search*	☑ Undo task at delete

Reference- and destination object

Source objects		Source default	.
Template		Target default	
Master action			

Detail area

Standard table		☐ Create new objects for implementation
Base object for new destination		
Root-object for target search		

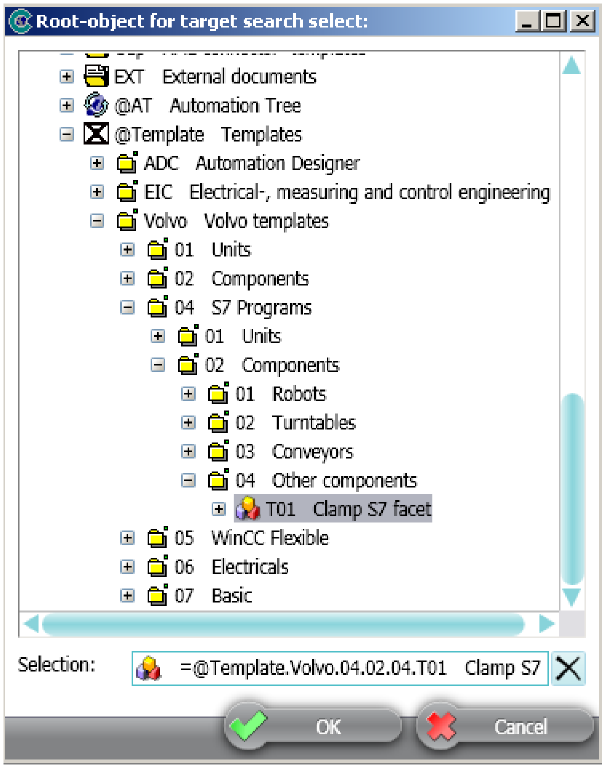
Funktion

☐ Active for hierarchical assemblies

Execute task

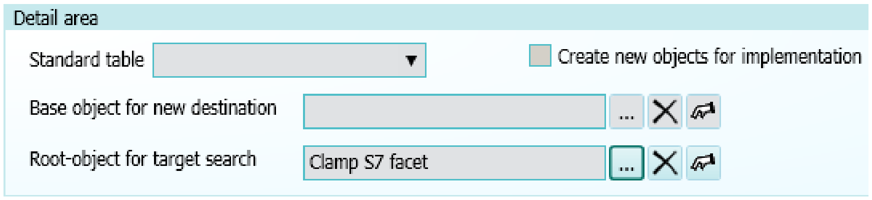
Result ☐ Status

9. Navigate to an appropriate template and press OK.



Selection: =@Template.Volvo.04.02.04.T01 Clamp S7

10. When the eBlock is executedin the engineering project the selected template will be copied in.



Detail area

Standard table	
Create new objects for implementation	☐
Base object for new destination	
Root-object for target search	Clamp S7 facet

11. Press OK to save