School of
Architecture, Computing & Engineering

Virtual Machine and Code Generator for PLC-Systems

A Thesis submitted to the University of East London for the degree of Master of Philosophy

Thomas Gasser
Dipl. El Ing. FH

Student Number: 0537311

Supervisor
Dr. D. Xiao, University of East London

30.09.2013

Acknowledgments

This thesis is the result of six interesting years of research in the domain of PLC-Systems in the area of building automation, building control.

Many thanks are going to Prof. Dominic Palmer-Brown for his assistance and encouragement in the early stages of my work.

I wish to express my gratitude to my Director of Studies Dr. David Xiao for his continuous and effective guidance, and his valuable comments. He was a source of very helpful, constructive criticism.

Special thanks go to Prof. Dr. Harald Wild, who encouraged and supported me in the past seven years. His guidance, his critique and his friendship are most valuable to me!

Thomas Gasser

Abstract:

In the programming of PLC-Systems (PLC = Programmable Logic Controller) in building automation there were no vital changes over the past few years. Most users of PLC-Systems in building automation do the programming in FBD (functional block diagram). The users always start their projects with the programming of the PLC software. After that the HMI (Human Machine Interface) or the SCADA-System (Supervisory Control And Data Acquisition) are added as subsequent Tasks. Programming tools like CoDeSys build on this bottom up approach. Most of the time the user has to cope with different tools for the different tasks and the programming tools are built for a specific hardware. Almost every PLC manufacturer has its own implementation of the IEC 61131 standard with various interpretations. So a change of hardware means that the programming tool, the way how to program the PLC-System, the HMI tool and the SCADA tool changes. The interfaces between the tools are sometimes pretty clumsy and software, generated with these tools, is often not easily transferred to the PLC-System of another PLC-System manufacturer.

This investigation proposes a top down approach to a project. For the first time the starting point of a project will be the plant diagram. The programming will be intuitively done in the plant diagram. Template objects (e.g. for pumps) created with the object orientated paradigm will be used to accomplish this task. The created PLC software will run in a virtual machine on the PLC-System and thus will be reusable on PLC-Systems of different manufacturers, which will eliminate the dependency on certain hardware.

The merging together of SCADA, HMI development and PLC programming will lead to a more natural way in programming PLC-Systems. In a first step the user draws his plant diagram with template objects (TO) and afterwards binds the template objects on the plant diagram together. With this approach the user only has to cope with a single software tool. The independency from the PLC hardware will be an additional alleviation for the user.

The aim of the investigation is to determine the realisability of a generic SCADA-System FBD programming language, a virtual machine and a code generator for PLC-Systems.

Table of contents

List of figures

List of tables

List of listings

Chapter 1: Introduction

In IEC 601131-3 Standard, IEC (2003), two graphical programming languages are defined, FBD and LD (Ladder diagram). Ladder logic is the oldest programming language for PLC-Systems. Programming in LD is done by copying the wiring diagram into the PLC-System programming tool. FBD consists of functional blocks which are connected together as it is done with logical gates.

Using the SCADA programming language, the logic operations are created a special programming mode in the background in while the process charts are drawn. The programming will adhere to the ISO 16484-3 Standard ISO (2005). Kandare et al (2003) shows an approach to PLC code generation based on a real world problem. An abstract functional block diagram is used to generate ST (Structured Text). ST is also part of the IEC 61131-3 Standard and is similar to high level languages as C++ or Java. With the SCADA programming language the conjunction data is saved in a definition file. The definition file is used by the code generator to create the source code for the selected PLC-System. Ways have to be found, to ensure that the generated PLC code is consistent and that in a PLC cycle every conjunction is evaluated in the correct order. The generator has to generate the correct PLC code for every possible solution a user may program, Aho et al (2007).

Different approaches to code generation for PLC-Systems have been used in previously done investigations. Some work using Petri nets as modelling base, Thieme and Hanisch (2002), Klein et al (2003). It has to be determined how a generic code generator can be realized to enable an easy implementation of an additional PLC-System. For this task, the principles of compiler development, Aho et al (2007), are used. It is essential, that only a small part of the generator, the backend, has to be changed if an additional PLC-System should be implemented. For this purpose it is essential to assess how data is stored in different PLC-Systems. A prototype of the code generator has to be developed. The source code produced by the code generator is saved on various target systems in a similar way (e.g. in data blocks or arrays) and run on a virtual machine. The virtual machine checks the logical operations repeatedly. If there are any changes of states in the logical operations, the respective code is executed. Suitable solutions are developed for an implementation of a virtual machine on a PLC-System with the highest possible gain of performance.

Virtual machines are covered in detail in Smith and Nair (2005). Performance was tested for the Saia-Burgess PLC-Systems.

Developing a programming language for SCADA-Systems which is independent of PLC hardware manufacturers is addressed. The SCADA-System programming language should be a kind of a functional block language as it is much easier for people to work with functional blocks and do the programming graphically than with an abstract programming language. LabVIEW from National Instruments or MATLAB from the MathWorks are good examples for graphical programming tools.

1.1 Traditional approach:

As shown in Figure 1.1-1: Traditional approach below the traditional approach in a building automation project stipulates, that first the software for the PLC is developed and as a subsequent task the visualisation is adapted to the PLC software.

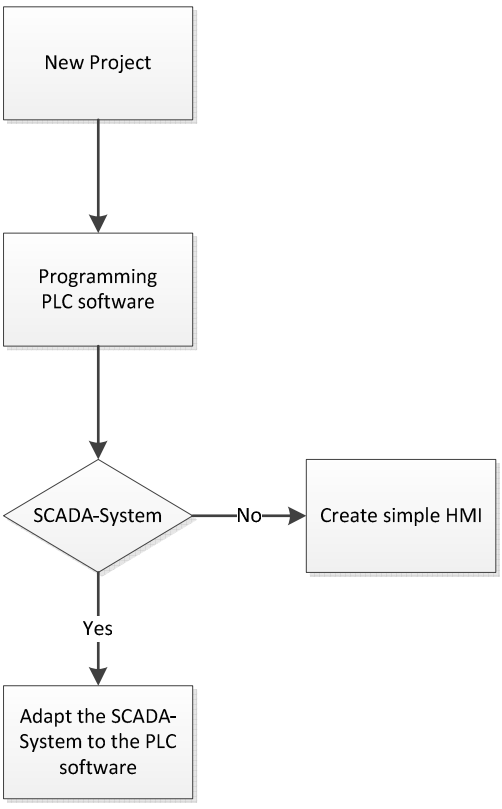

Figure 1.1-1: Traditional approach

1.2 New approach:

With the SCADA-System programming language a novel approach was chosen (Figure 1.2-1: New approach). The PLC software is developed using the object oriented paradigm. The whole engineering and the design of the plant diagrams is done with template objects. In special applications a huge amount of the PLC code which runs the plant can be generated. The necessary connections between the TO (Template Objects) is built afterwards in IL (Instruction list), FBD or ST. IL, FBD and ST are part of the IEC 61131-3 standard IEC (2003). In standard applications the whole PLC code can be generated without the need to create additional code to link the TO.

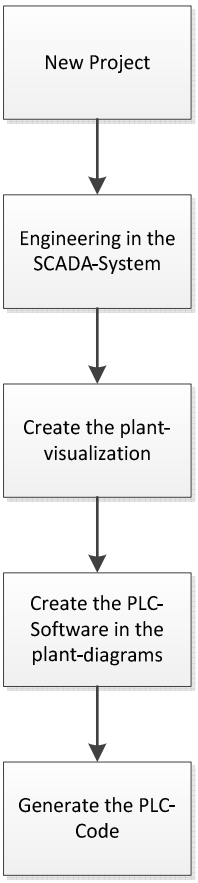

Figure 1.2-1: New approach

1.3 Objectives

There have been several objectives in this investigation.

1. To develop PLC software using the object oriented paradigm.
2. To develop a generic SCADA-System-FBD programming language
3. To develop a virtual machine for PLC-Systems.
4. To develop a code generator for PLC-Systems which uses the definition files of a SCADA-System as input.

Chapter 2: Literature Survey

A literature survey has been conducted on the aims of this investigation. Several related fields of interest have been included into the survey to broaden the horizon of this investigation and to get new ideas for the programming of PLC controllers.

It was quite difficult to find any literature covering code generation for PLC controllers in the field of building automation. The object oriented paradigm and code generation in the area of building automation seems to be a rather unexplored field. Nevertheless there are some helpful sources.

2.1 Code generation

Some of the most helpful suggestions for this investigation were found in *Code Generation in Action* by Herrington (2003). Herrington (2003) describes the usefulness of code-generation and its pitfalls in detail. He explains code generation in a way that not only engineers understand it but also managers can see the usefulness of code generation and the possibility of saving time and money by the use of it. The savings stem from the saved time in the engineering process and the constant high quality of the generated software. Automatically generated software is much less susceptible to programming errors than software which is written and rewritten every time it is used by hand.

The spectrum of code generation reaches from very simple generators for the body of a C++ class and the appropriate header file to highly complex generators which automatically build database access software. Various generators can be assembled to highly sophisticated frameworks.

In most IDE's (Integrated Development Environment) for modern programming languages there are code generators used to create running skeleton applications for the different types of application that can be developed. These generators are very useful and taking over the, most of the time, boring and time intensive task of setting up a new software project from the programmer. It may be seen as a disadvantage that as a consequence the knowledge of what a software application is based on, is

no longer of much relevance. This can of course bring forth new problems because of non-qualified or unexperienced programmers writing complex applications.

2.1.1 Generative programming

The concept of generative programming is described by Czarnecki and Eisenecker (2000). They show generative programming methods and Metaprogramming to be similar as the concepts mentioned in chapter 2.1 and introduced by Herrington (2003). Czarnecki and Eisenecker (2000) introduce Domain Engineering as a means to develop a workbench of reusable parts in a family of systems. Similar thoughts are expressed in “Software factories” by Greenfield and Short (2006). The authors of both works emphasize that generative methods are a powerful tool to improve the quality of software because of the use and reuse of well tested libraries and functions.

2.2 Compiler construction

A most enlightening source is the dragon book by Aho *et al.* (2007). He and his co-authors provide a very deep insight into the methods used in the field of compiler construction. Some of these methods are very helpful in this investigation. Chapter 3 of the dragon book covers lexical analysis which is, as Aho *et al.* (2007) states very clearly, the foundation of text processing of all sorts. Text processing is of great importance for this investigation.

Lexical analysis is the first phase in the compiling process. The lexical analyser is used to analyse the source code character by character. The source code is grouped into lexemes and a sequence of tokens for each lexeme is generated as output. This output is used as a basis for the parser which does the semantic analysis.

2.3 Virtual machines

Craig (2004) describes the formal background of virtual machines. As described by Smith and Nair (2005) there are two different kinds of virtual-machines.:

- Process Virtual Machines
- System Virtual Machines

2.3.1 Process virtual machines

Smith and Nair (2005) states that a process virtual machine is used to provide a user application with a virtual application binary interface (ABI). Process virtual machines are used to emulate, replicate and optimize. In the past years the focus has been laid on high-level-language VM's. The two best known examples for high-level-language process virtual machines are the Java VM architecture and the Microsoft common language infrastructure (CLI) which is used as the basis of the .NET framework. The main goal of this type of virtual machine is to achieve platform independence. This should enable cross platform portability of application software. To achieve this goal it is necessary to take into account, that the instruction set architecture (ISA) is different from one platform to another. Both of the above mentioned systems are using an ISA based on bytecode. For each target platform a virtual machine has to be developed to run the bytecode.

2.3.2 System virtual machines

These systems are the origin of the term virtual machine. These virtual machines are emulators for a whole system. A single host platform with its hardware is used by different VM's whereupon every VM's is given the illusion of having its own Hardware. A well-known type of system virtual machine is the hosted VM. In this type of VM the virtualization software runs on a host operating system like a standard, native application. The VM itself is controlled and executed by the virtualization software. All native drivers and system calls can be used by the VM through the virtualization software. The virtualization software brings an additional software layer into the system which may result in a degradation of performance. VMware is a well-known representative of such a system. Zhou and Chen (2009) shows an example of a system virtual machine for PLC-System, the main idea of the research is to simplify the process of to port an existing PLC-Software to another PLC-Hardware.

2.4 PLC-Systems

A Programmable Logic Controller (PLC) is a computer which is used for automation purposes. PLCs are the replacement for hard-wired relay systems. PLCs are used in almost every part of our lives from assembly lines to the heating systems of modern buildings. Bonfè *et al.* (2009) shows the use of a PLC-System to implement a robot control system where Weina and Liwei (2011) using a Siemens PLC-System to

control a manipulator transportation control system. PLC-Systems are most flexible and can easily adapt to changing requirements.

2.4.1 Standardisation

PLC-Systems are subject to conditions of the IEC-61131 standard. As can be seen in IEC (2003) for most parts of a PLC-System strict guidelines are set by the standard but nevertheless there is some room for the manufacturers to distinguish themselves from their competitors.

2.4.2 Modeling

Basile *et al.* (2013) uses Petri nets as a means to model the system behaviour of complex industrial processes. Min *et al.* (2013) proposes a component based modelling system to tackle the problems of cyclic programs and to synthesize the necessary PLC code.

2.4.3 Special Articles

Today the protection systems of nuclear reactors are realized by the use of PLC-Systems. Yoo *et al.* (2013) proposes a platform change from PLC-Systems to FPGA (Field-Programmable Gate Array). Vogel-Heuser *et al.* (2013) compares the UML-based engineering versus the engineering by the use of IEC 61131-3 in teaching PLC-System programming.

2.4.4 History

The first PLC-System, the Modicon 084, was introduced in 1969 by Modicon. It was Modicons answer to a request from General Motors Hydra-Matic to replace their hard-wired relay systems, Wikipedia (2013).

2.4.5 System design

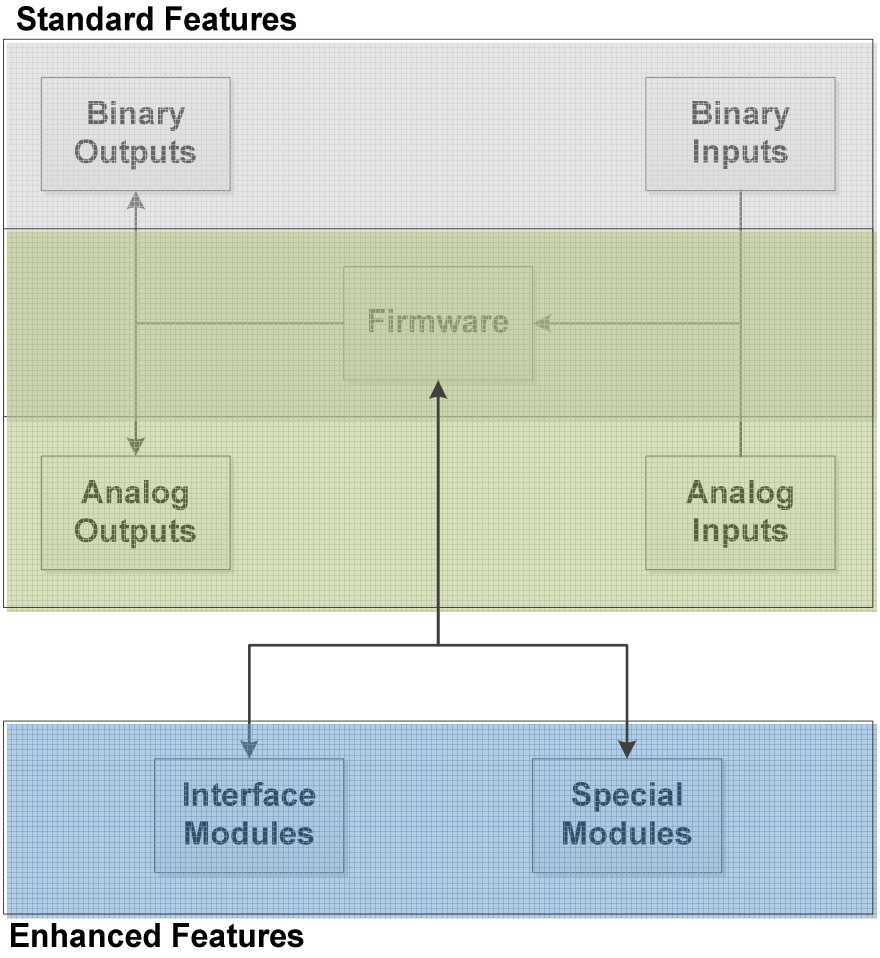

Figure 2.4-1: PLC-System design

In its simplest form a PLC-System consists of binary input and binary outputs respectively. The inputs and outputs represent the PLC interface to the controlled environment. In most systems there are analog signals to be taken into account for the control algorithms. These can be temperature, pressure, frequency and many more. In more complex systems the enhanced features of PLC-Systems such as Interface modules or special modules are often used. In the field of building automation these can be comprised of fieldbus interfaces, an interface to DALI lightening devices or to M-Bus energy measurement devices. In other fields of application there may be special modules as weighting modules or modules to control stepper motors. The way of how these real world signals are processed differs slightly from one PLC manufacturer to another. The most common system to process the acquired data can be seen below in Figure 2.4-2: Data processing.

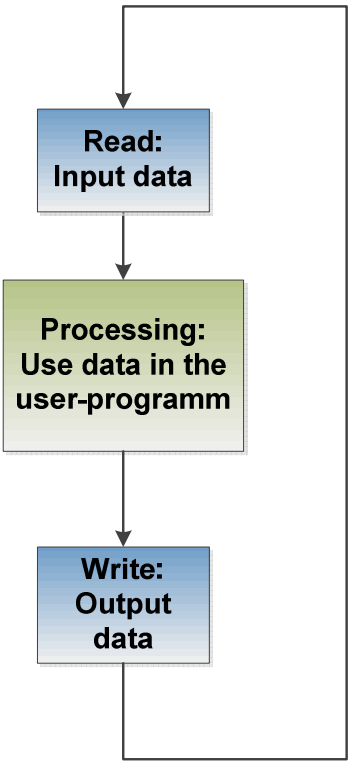

Figure 2.4-2: Data processing

The data processing is normally split into three phases:

- First phase: Read input data from the input modules.
- Second phase: Process the input data with the user program.
- Third phase: Write the changes of the output data to the output modules

The first and the third phase are controlled by the operating system of the PLC-System. After reading the input values the operating system gives control to the user program which processes the newly acquired input data into the new output data. Then the operating system takes control and writes the updated output data to the output modules. After that a whole cycle has been run through and a new cycle starts with the first phase. There are systems with a fixed cycle time which are working with various tasks where every task can have its own, fixed cycle time. On the other hand there are systems where the cycle time is floating and therefore changing according to the user program.

In addition there are some exceptions to the above mentioned cyclic behaviour. The Saia-Burgess PLC-Systems for instance do not exactly work after the above mentioned system. As can be seen at Saia-Burgess (2013). These systems are running through the user program in a cyclical manner but they are reading from the

input modules and writing to the output modules at the sametime. The cycle time is floating depending on the user program.

2.4.6 Programming

The different programming languages of a PLC-System are subject to Part 3 of the IEC61131 standard.

The standard describes five different PLC programming languages:

- Sequential Function Chart (SFC)
- Instruction List (IL)
- Structured Text (ST)
- Ladder Diagram (LD)
- Function Block Diagram (FBD)

Every one of these programming languages is aimed at a different type of users. Historically the first and therefore oldest PLC programming language is the ladder diagram which provides means of copying the electrical wiring diagram into the PLC-System. It is therefore often used by technicians with a wide knowledge of hard-wired relay systems. Which programming language a programmer will select depends heavily on the field of his work, his skills of programming and his knowledge of the PLC-System used. In the field of building automation the programming language most frequently used is FBD because of its minimal training requirements. An attempt to create a system to verify the developed PLC programs was made by Biallas *et al.* (2012).

2.5 SCADA-Systems

SCADA is an acronym for Supervisory Control And Data Acquisition. As described in Wikipedia (2013) a SCADA-System is an industrial control system (ICS). As distinguished from other ICS, SCADA-Systems can be large scale and control multiple plants over large distances. SCADA-Systems are used in the infrastructure automation for instance in distributed heating systems.

Usually ICS's are illustrated with the automation pyramid. SCADA-Systems can be found on the Enterprise and the Control Level respectively.

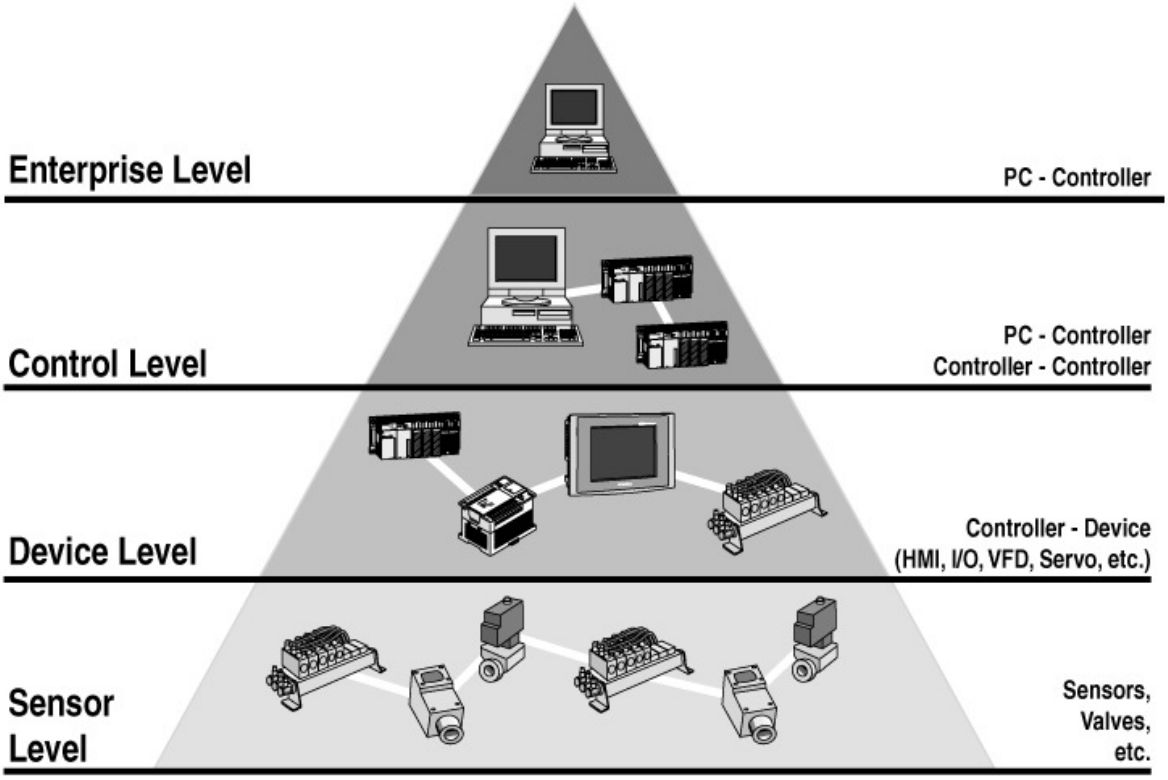

Figure 2.5-1: Automation pyramid (http://www.al-pcs.com)

2.5.1 SCADA components

As shown in Figure 2.5-1: Automation pyramid (http://www.al-pcs.com) there are various levels in an automation system. The automation pyramid is derived from the Open Systems Interconnection Model (OSI-Model).

Sensor, field level

Similarly as in the OSI-Model the lowest level, called layer in the OSI-Model, is where the interaction with the real-world takes place. This is the level where the actuators and sensors are placed.

Device level

The signals captured by the sensors in the field or the signals needed by the actuators are connected to the controllers on the device level. The controllers can be PLC-Systems or remote IO-Systems. Usually one controller is designed to control an entire entity, a heat pump for instance, of a site by itself, without need of the SCADA-

System. On this level simple Human Machine Interfaces (HMI) such as WEB-Panels or text based terminals can be found.

Control level

On this level the data exchange between the different controllers of a site or of various sites takes place. More sophisticated HMI-System can be found. They are part of the SCADA-System and therefore offer much more information to the operator of the site. On this level there are tools to thoroughly analyse the entities of a site and to optimise them to a very high level of efficiency.

Enterprise level

This level is optional and in smaller systems not present at all. Operators with their facilities distributed over a wide area use this level to realise centralised data acquisition in a control centre. This is usually done over the internet by the use of standard IT protocols and portal solutions. On this level automated data acquisition for accounting purposes takes place. Therefore the SCADA-Systems offer interfaces to accounting systems and business software.

2.5.2 SCADA concepts

Usually modern SCADA-Systems are built in a most modular manner. The user can choose the modules suitable for the project at hand. Below in Figure 2.5-2: SCADA concept (www.promosnt.ch) the conceptual structure of the ProMoS NT SCADA-System can be seen as an example.

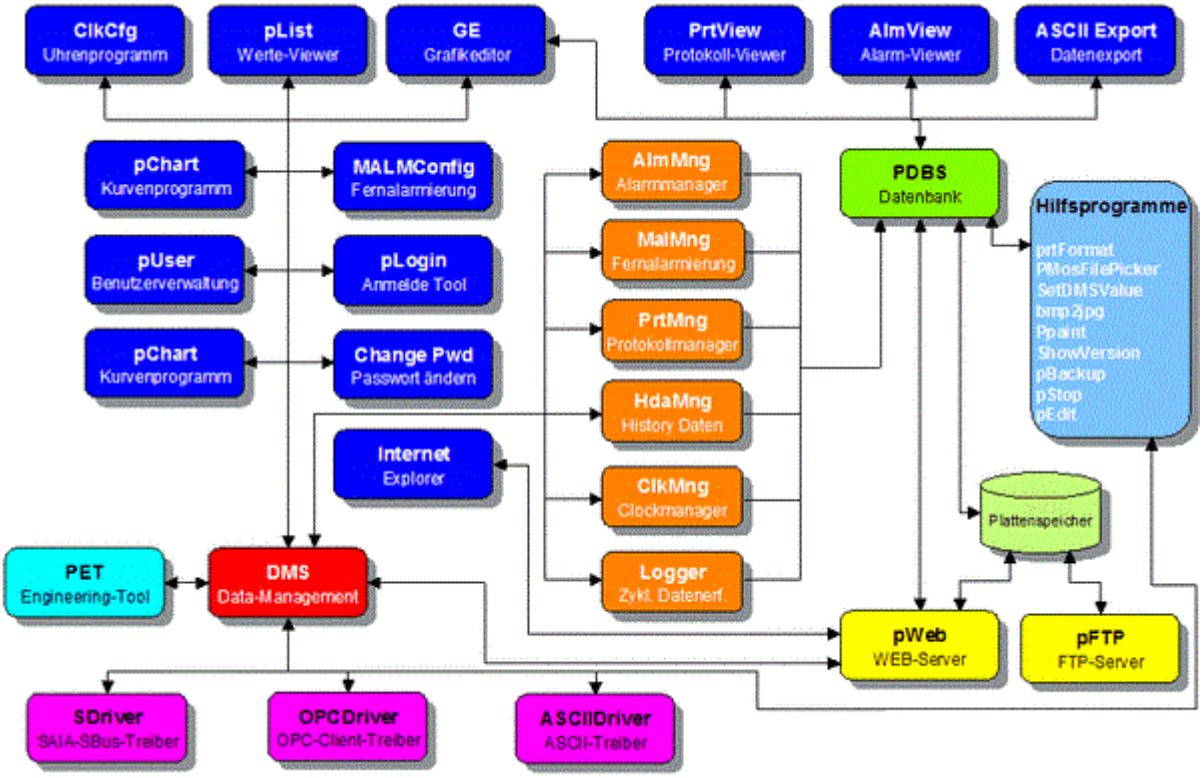

Figure 2.5-2: SCADA concept (www.promosnt.ch)

In the past few years the standard networking protocols well known from the IT world have found their way into PLC-Systems and therefore into SCADA-Systems. Thanks to those protocols it is possible to build complex, distributed systems. Different sites can be linked together over wide areas. This makes it easier to supervise them and to react if any disturbance occurs. In case of an incident an operator can check the site on the SCADA-System and take appropriate actions. Reports on the energy consumption can be generated automatically to simplify accounting.

Chapter 3: Research Methodology

3.1 The Qt application framework

For this investigation the application framework used is Qt. Qt is a cross-platform application and UI framework for developers using C++ or QML, a CSS & JavaScript like language. Qt's Visual Studio 2010 plugin has been used to allow a smooth workflow between Qt and Visual Studio 2010.

3.2 The software design

The Software is designed as a Client / Server System. In this investigation the pTool-Application acts as the server in the System and the PLC code generator, sGen, is the client. Only the code generator for the Saia-Burgess PLC-Systems has been developed.

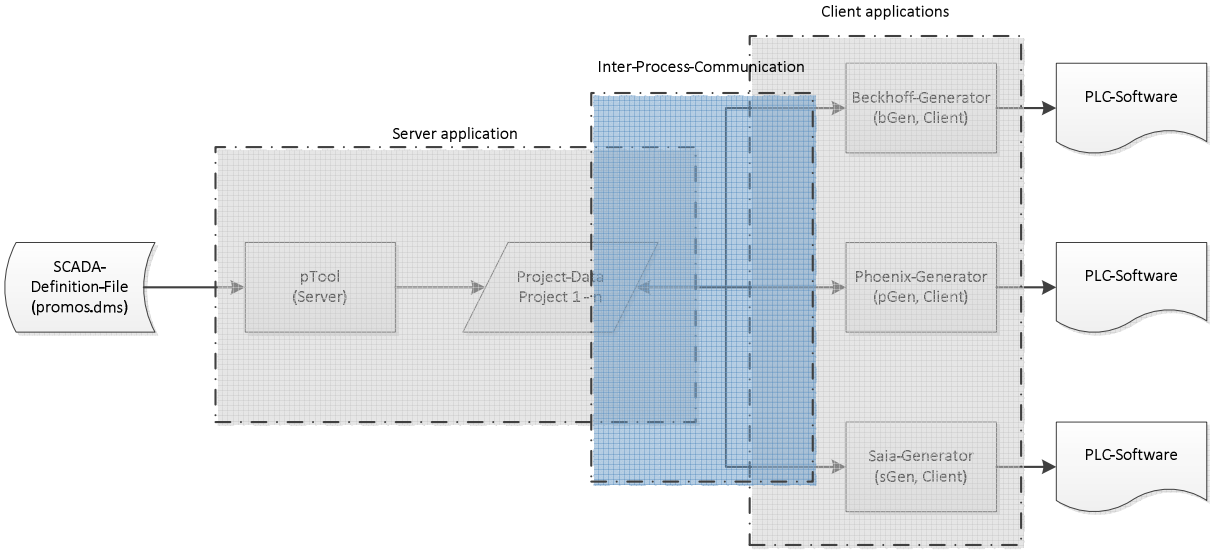

Figure 3.2-1: Software design

3.2.1 The server application (pTool)

At the moment the application can open one project at a time due to an internal restriction. Nevertheless the software architecture has been planned in such a way that it enables the application to act as pivot in the engineering process of PLC software for the building automation field with multiple projects open and exchanging parts with each other.

3.2.2 The client application (sGen)

At the moment one client application has been developed. The application generates code for PLC-Systems from Saia-Burgess.

3.3 Inter-process communication

As described by Wikipedia (2013) Inter-process communication (IPC) is used in computing as a means for threads or processes to communicate among each other. IPC is a generic term for a class of methods to fulfil this task. The communicating processes can run on the same machine or can run on different machines connected by a network. In addition to the use as means of communication there is a second use of IPC which is the synchronization of processes or threads using shared resources.

3.3.1 Systems used for Inter-process communication

There are several different systems to implement IPC in an application.

Method	Description
Socket	Data sent over a network interface in the form of a stream. The recipient process can either be found on the same computer or on another computer attached to the network.
Pipe	A bidirectional stream of data using the host systems standard input and output devices. The data stream is read character by character.
Named pipes	The named pipe is an extension to the traditional pipe concept. It is also known as FIFO because of its behaviour. On UNIX / Linux Systems named pipes are system persistent and have to be deleted when not used anymore. On Windows Systems the named pipe is deleted after the last reference to the pipe is closed. Usually named pipes are implemented as files.
Semaphore	A structure that is used to synchronize threads or processes using shared resources.
Shared memory	An area of memory to which multiple processes are given access, allowing all of them to change it and read changes made by other processes.

Table 3.3-1: IPC Methods

3.3.2 Sockets

A socket is an endpoint in a line of communication, usually of an IPC. Gay (2000) describes sockets to be similar to the telephones used when calling someone else. The telephone represents the socket and the telephone network represents the network used by the sockets. In this investigation the easier to use named pipes were preferred to the use of sockets.

3.3.3 Named pipes

This investigation uses a QLocalSocket to implement IPC. On a Windows System the QLocalSocket is implemented as a named pipe. The implementation is done as a client-server system and therefore named pipes are working much like sockets.

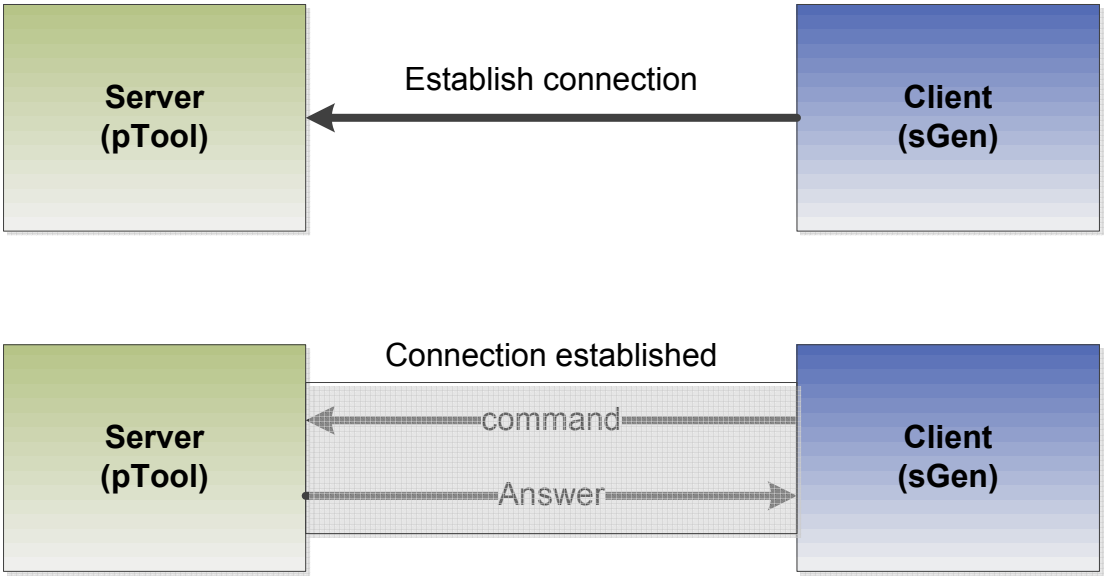

Figure 3.3-1: IPC over named pipe

3.4 Data structures

Data structures and Algorithms used with them are described by Loudon (2000) and Sedgwick (1992). Data structures are used to store and organize data in a most efficient way. This with regard to the type of data to be stored, its handling and the way it should be used. Data structures are a key feature to design efficient algorithms. Efficiency is always judged in regard to time and memory consumption. In the sections to follow, the two main data structures used in this investigation are described in detail.

3.4.1 Hashing

For this investigation, the use of hashing in connection with hash tables is of interest. The use of hashing in cryptology has not been used in this work and therefore has not been researched.

The hash table data structure is used to store and map key value pairs. This sort of data structure is also referred to as an associative array. With this data structure it is possible to directly refer to data sets in a table by the use of arithmetical transformations converting keys into table addresses. If $key \in \{1, 2..n\}$ then one can store a set of data with a $key = i$ in the table at the position with an $index = i$. With this simple method, the key value is directly used as pointer to the associated value. Sedgwick (1992) describes hashing to be a generalisation of the above outlined simple method for searching algorithms where such specific knowledge on the key is not available. The first step in the process of searching by the use of hashing is to find a suitable hash function which transforms the key into a table address. In the ideal case the hash function transforms different keys into different table addresses. In most of the cases the hash function will not be perfect and therefore it will transform more than one key into the same table address. Thus the second step in the process of searching by the use of hashing will be concerned with collision resolution.

3.4.2 The hash function

It is essential to find the right hash function and an effective implementation algorithm to achieve good hashtable performance. This could be quite a difficult task to achieve. As Wikipedia (2013), Rahm (2013) and University of Paderborn (2006) are describing it is a basic requirement for a hash function to uniformly distribute the hashvalues. If the hash values are not distributed uniformly, the number of collisions and the cost for their resolution will increase. To ensure uniformity by design the function may be tested empirically by the use of statistical tests.

As mentioned in 3.4.1 Hashing, if the number of keys is known beforehand, a perfect hash function can be used to create a hash table with no collisions. Perfect hashing can provide constant time for lookups even in the worst case.

3.4.3 Hash table statistics

The crucial factor of a hash table is the load factor α. It is calculated as follows: $\alpha = \frac{n}{m}$ where n is the number of entries and m is the number of places or buckets in the hash table. The load factor should stay below a certain value for the hash table to perform well. On the other hand it is not of great use to bring it near 0 which means that memory is wasted.

3.4.4 Collision resolution

As in most cases the hash function will not be perfect. Collision resolution has to be considered. There are various approaches to this. Below different ways of collision resolution are described. Each of these is more suited for a specific type of data.

3.4.4.1 Separate chaining

The variant of collision resolution known as separate chaining implements each bucket as an independent entity with a list of entries stored. The cost of operation on such a hash table is: $t_{HT} = t_{bucket} + t_{list}$ where $t_{bucket} = constant.$

To achieve good performance of the hash table each bucket should have zero or one and seldom two or three entries. Thus the focus will be on structures that are efficient in time and space for these cases.

3.4.4.2 Separate chaining with linked lists

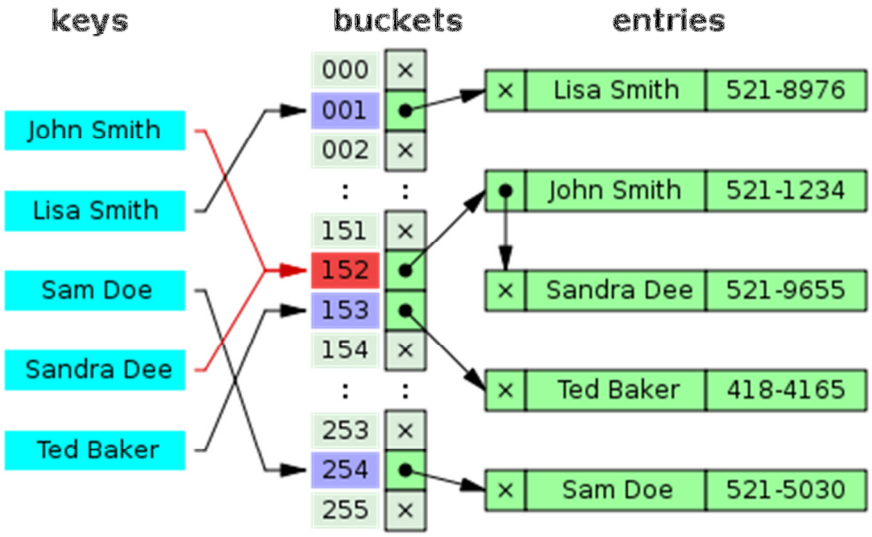

Figure 3.4-1: Hash collision resolved by separate chaining (Wikipedia)

As Wikipedia (2013) describes, one solution to solving collisions is by separate chaining using linked lists. This alternative is popular because it requires only basic data structures with relatively simple algorithms. The hash function used can be kept simple but may be unsuitable for other methods. The cost of a table operation is that of scanning the entries of the selected bucket for the desired key. If the distribution of keys is sufficiently uniform, the average cost of a lookup depends only on the average number of keys per bucket which is in fact on the load factor α. Chained hash tables remain cost effective even when the number of table entries n is much higher than the number of buckets. Their performance degrades more gracefully (linearly) with the load factor α. The worst case for this type of hash table is the situation where all entries are in the same bucket. In this case the hash table will be inefficient and the cost of operation will be the cost to search the list contained in the bucket. The biggest advantage of this type is the fact that there can be more entries in the table than buckets are available.

3.4.4.3 Separate chaining with list heads

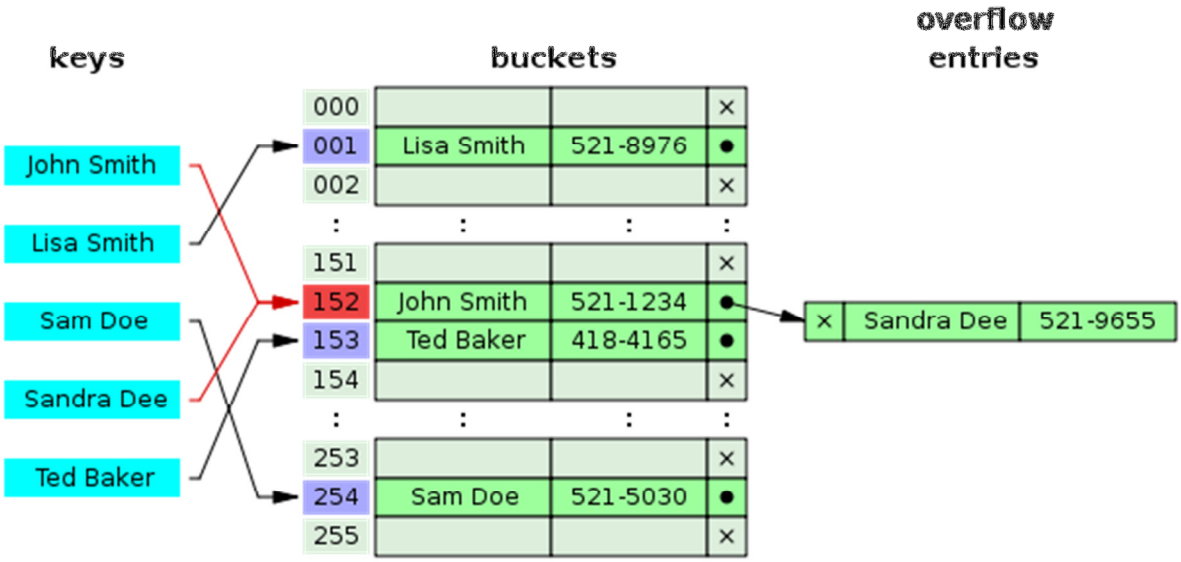

Figure 3.4-2: Hash collision by separate chaining with head records in the bucket array (Wikipedia)

Another type of collision resolution described by Wikipedia (2013) is the separate chaining with list heads. The first entry of each chain is stored in the bucket itself. Thus one pointer traversal can be saved in most cases. Cache efficiency of the hash table increases by the use of this solution.

3.4.4.4 Open addressing

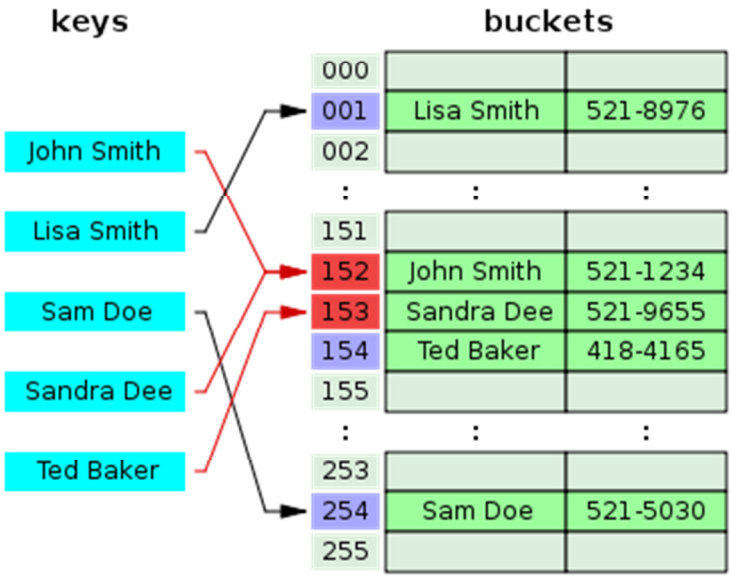

Figure 3.4-3: Hash collision resolved by open addressing with linear probing (Wikipedia)

Collision resolving by the use of open addressing means that every entry is stored in the bucket array itself. If some new data has to be inserted the system looks for free buckets. If the targeted bucket is already taken algorithms are used to search for the next free bucket where the entry will then be made. If a key is searched for, the same algorithms are used to scan through the buckets until either the entry is found or an empty bucket appears which indicates that the hash table does not contain the key searched for. The term “open addressing” underlines the basic concept that a place where an item is stored is not directly deducible from its hash value.

There are several algorithms used to store data in the hash table.

Name	Description
Linear probing	Fixed interval between probes (usually 1)
Quadratic probing	The interval between probes is increased by adding the consecutive outputs of a quadratic polynomial to the originally calculated hash value
Double hashing	The interval is calculated by another hash function

Table 3.4-1: Probing algorithms for open addressing

As can be seen in Figure 3.4-4 below, a major disadvantage of all open addressing algorithms can be seen when using linear probing as example. The performance graph shows a similar behaviour to a diode. Approximately at a load factor of $\alpha = 0.8$ a kind of avalanche effect took place and the performance degraded dramatically.

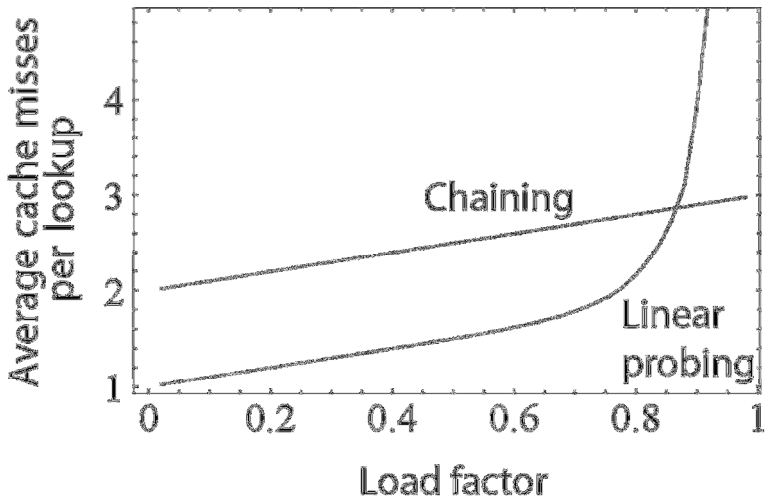

Figure 3.4-4: This graph compares the average number of cache misses required to look up elements in tables with chaining and linear probing (Wikipedia)

Open addressing requires some additional features of the hash function. The hash function should not only distribute the keys uniformly over the buckets but also minimise the clustering of hashvalues that are consecutive in the probe order. This is, as elucidated by Wikipedia (2013), fundamentally different to the use of separate chaining, where the only concern is that too many objects map to the same hash value; whether adjacent or nearby being completely irrelevant. In contrast to chaining open addressing does not need the time overhead used to allocate the memory for each new entry, therefore it can be implemented even on systems without memory allocator. Also there is no extra indirection required to access the first entry of each bucket. It has as well better locality of reference, particularly with linear probing. With small data entry sizes, these characteristics can result in better performance than chaining, especially for lookups. If the hash table has to be serialised this can be achieved with much less effort because no pointers are used. To conclude one can say, it is better to use open addressing with hash tables where the data entries are so small that they can be stored within the hash table and can fit into a cache line. They are particularly suited for elements of one word or less. If a high load factor α is to be expected, the data entries tending to be large or the data entries being of variable sizes, chained hash tables often perform as well or better.

3.4.5 Tree's

The tree data structure used in this investigation is a means to display hierarchical lists. The Qt-Framework provides a class which implements such a tree. This class is called QTreeView and it is part of Qt's model / view framework.

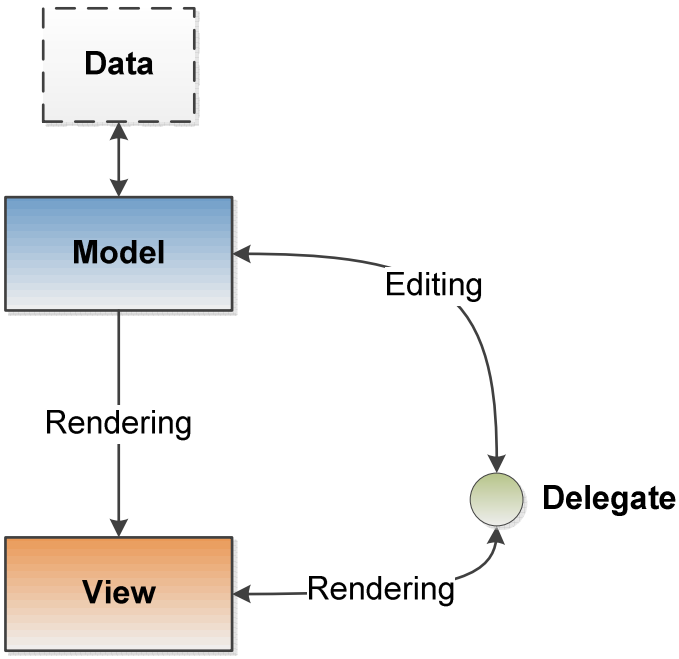

Figure 3.4-5: Qt model / view architecture

The model / view concept originates from the Smalltalk programming language and it is used to separate the data from its presentation, called the view.

The model represents the interface to the data. It is the only component in the architecture to communicate with the source of data. What the communication with the data looks like is determined by the type of data handled and the implementation of the model itself. The view obtains the model indexes from the model. Those indexes are references to data items. By the use of these model indexes the view can retrieve data items from the data source via the model. The delegate is called controller in other implementations of the model / view architecture. It is used to render the data and in case an item is edited in the view it communicates with the model directly also using model indexes.

Chapter 4: Implementation

4.1 Introduction

As described in Chapter 1: Introduction a wholly new approach of how a project in the field of building automation is set up and realised is introduced by this investigation. The introduced approach is a paradigm change compared to the traditional approach. The bottom up approach (see Figure 1.1-1: Traditional approach) is changed into a top down approach (see Figure 1.2-1: New approach). The new approach orients itself on the object orientated paradigm well known from high level programming languages as C++, Stroustrup (2000). Furthermore it is based on the graphical programming languages known from the IEC 61131-3 Standard, IEC (2003).

4.2 Initial position

Some of the principles regarding SCADA-Systems used in this investigation are in use since 1999 by MST Systemtechnik AG in their SCADA-System called ProMoS NT, MST Systemtechnik AG (2013). The results of this investigation initially were planned to be an improvement to the current code generation system implemented in PromoS NT.

4.2.1 Template object concept

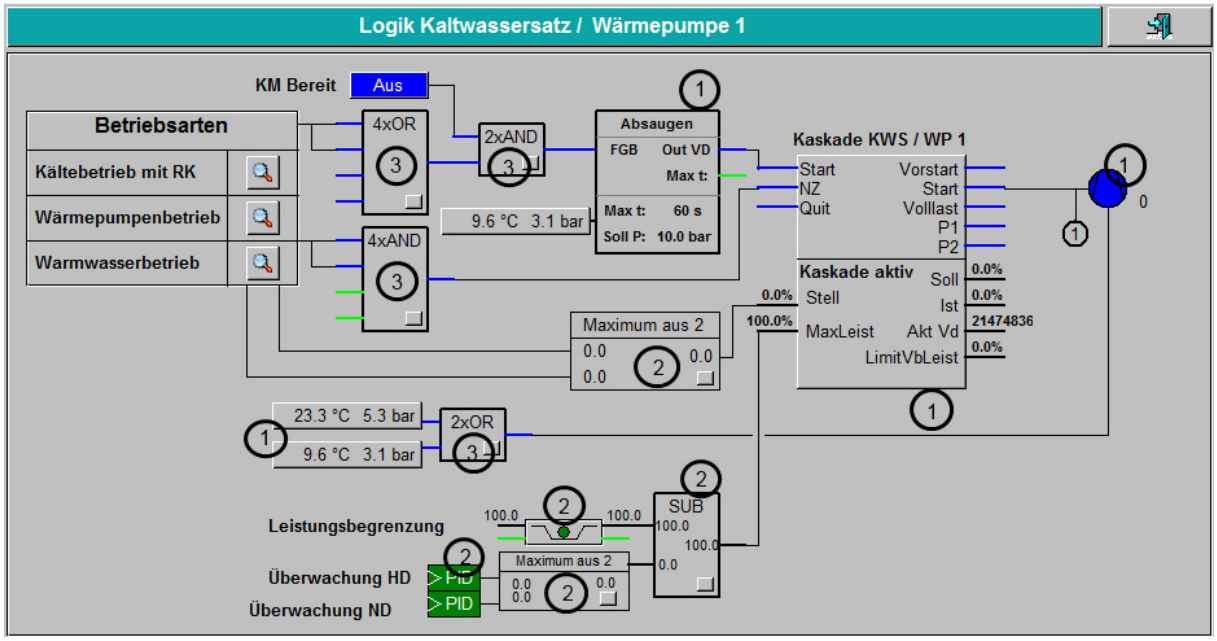

Figure 4.2-1: Logical plant diagram (runtime mode)

Figure 4.2-1: Logical plant diagram shows a typical Plant-Diagram used to program a heat-pump-system. In the table below the different elements, called TO (Template Objects) will be described.

Number	Description
1	Highly sophisticated templates from a library for refrigeration applications
2	Templates from the standard template library
3	Templates to do logical connections

Table 4.2-1: Description of the TO

The TO concept is derived from the class concept well known in high level programming languages as C++, Stroustrup (2000). With the TO concept often used plant functions are encapsulated, an interface to access the public member functions, the graphic icon, and the necessary user interfaces are designed.

TO-Part	Description
IL-Code	PLC software
Graphic Icon	Graphical representation for the TO in the plant diagrams (see Figure 4.2-1: Logical plant diagram (runtime mode))
1..n User Interface(s)	A means for the user to interact with the TO.
Generator Template	Used by the code generator to generate the function block calls.

Table 4.2-2: Parts of a TO

Well designed, tested and documented TO or even whole TO Libraries can accelerate the engineering process dramatically. Huge portions of PLC code can be reused in a wide range of projects.

Original equipment manufacturers (OEM's) are given the means to develop standard applications were the whole PLC code can be generated. Thus the reusability of the code can be pushed even further.

4.2.2 Programming on the plant diagram

Currently there already is a way to do the connections on the plant diagram. To link a certain output of a TO to an input of another TO one has to switch from runtime mode to programming mode.

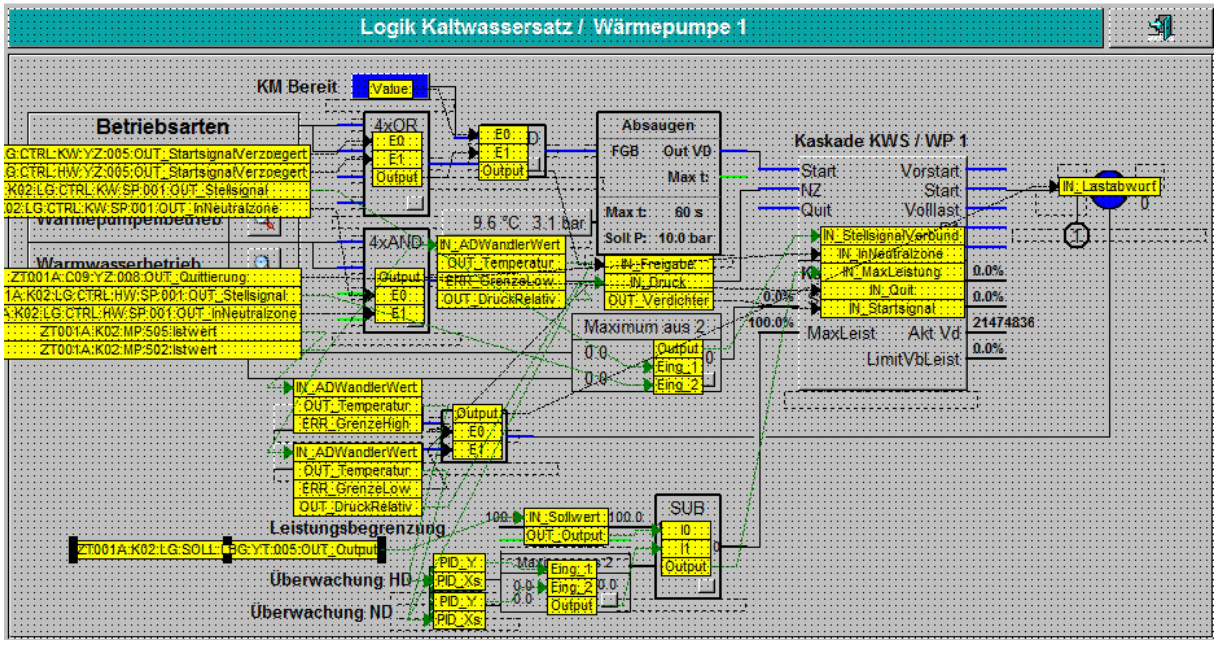

Figure 4.2-2: Logical plant diagram (programming mode)

As an example the *WARN_Absaugzeit* Signal of the *Absaugen* TO should be linked to the *IN_ManuellMinus* Signal of the *Kaskade KWS / WP 1* TO.

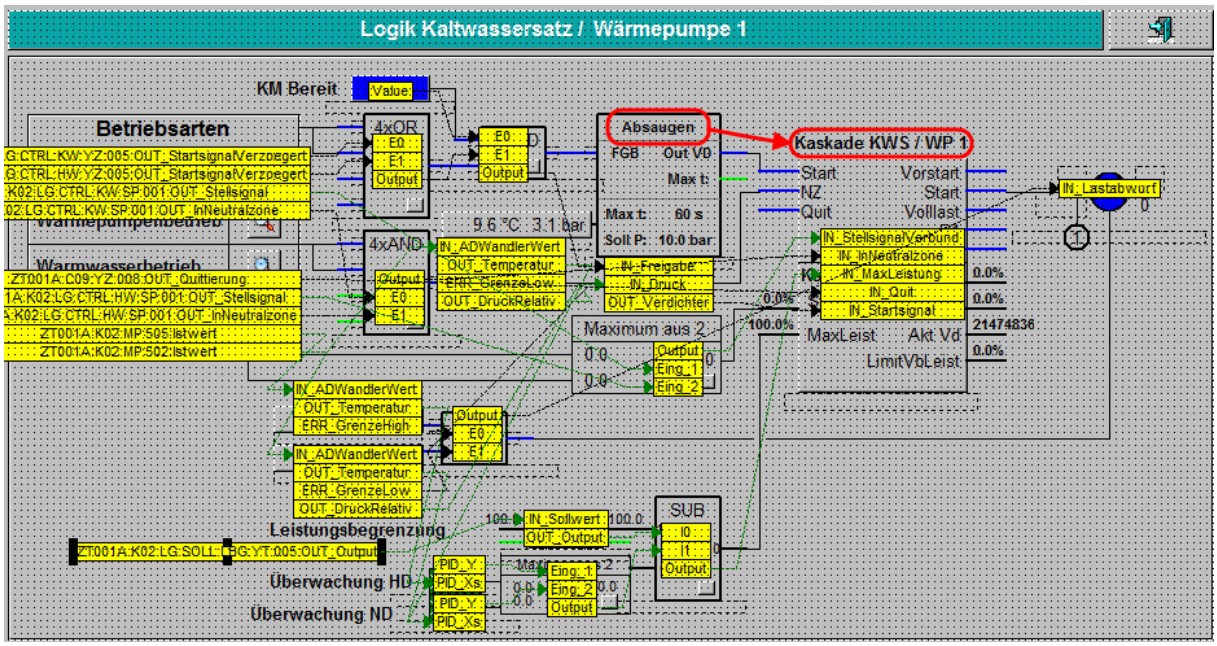

Figure 4.2-3: Logical plant diagram (example connection)

To link these two signals together the first step is to click on the source TO (Absaugen) and to choose the desired output signal (WARN_Absaugzeit) from the appearing list.

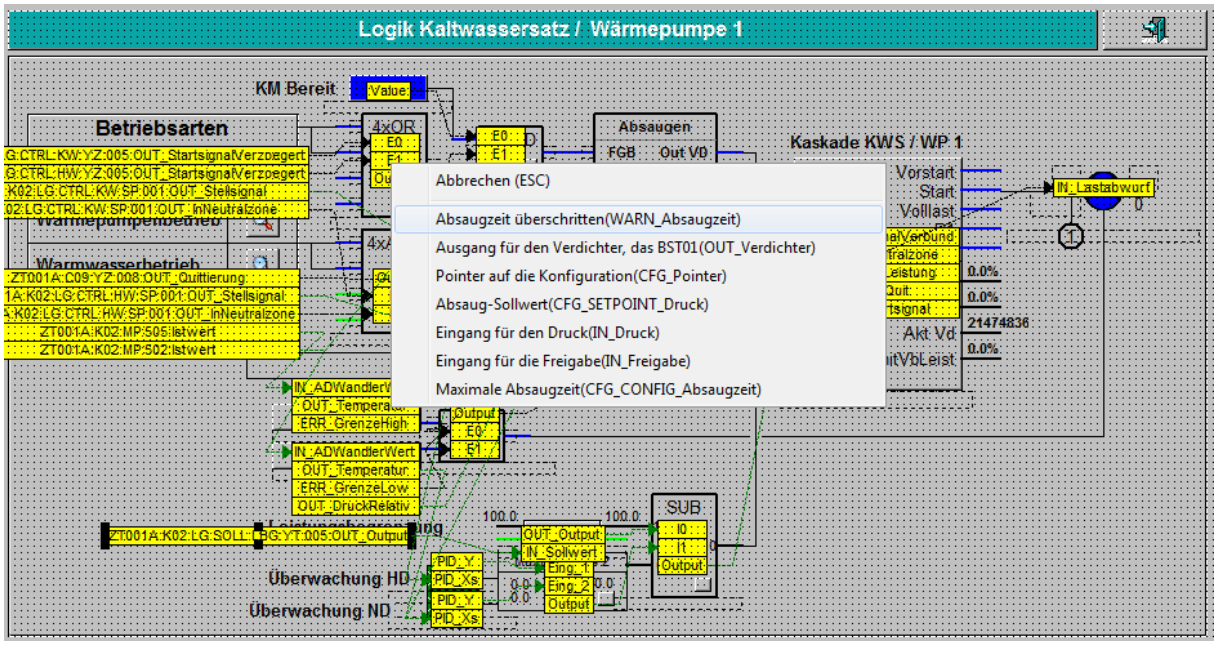

Figure 4.2-4: Logical plant diagram (choose output signal)

After the desired signal has been chosen the second step is to click on the target TO (Kaskade KWS / WP 1) and to choose the desired input signal (IN_ManuellMinus) from the appearing list.

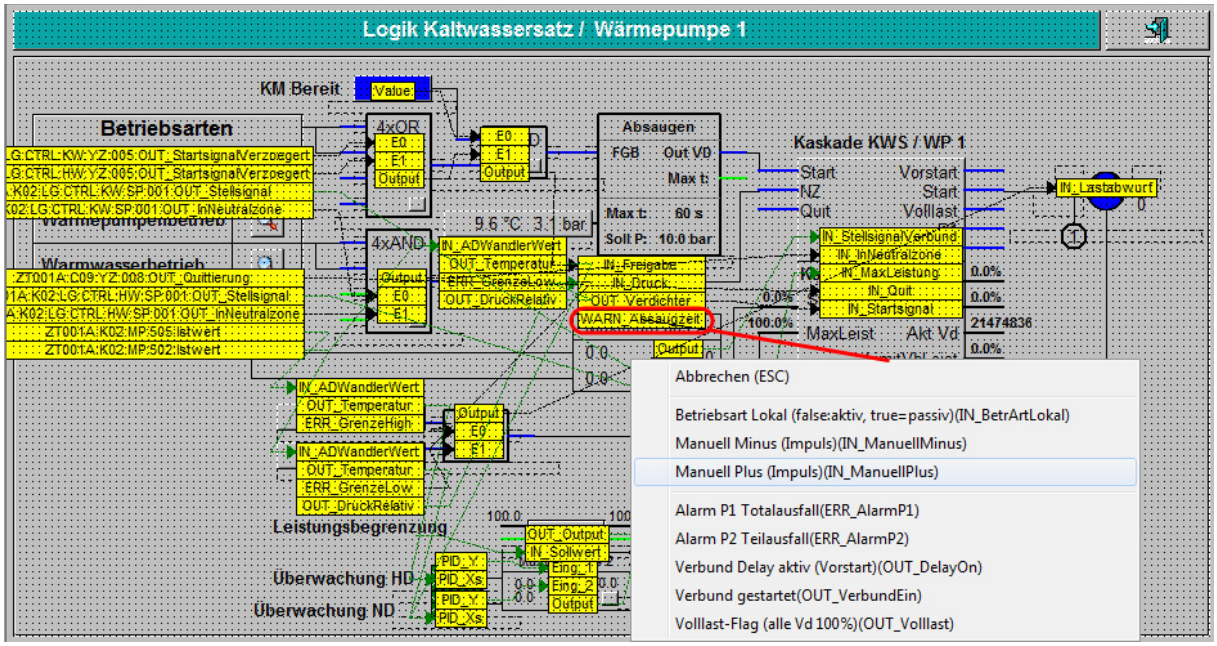

Figure 4.2-5: Logical plant diagram (choose input signal)

After the input signal is chosen the two signals are linked.

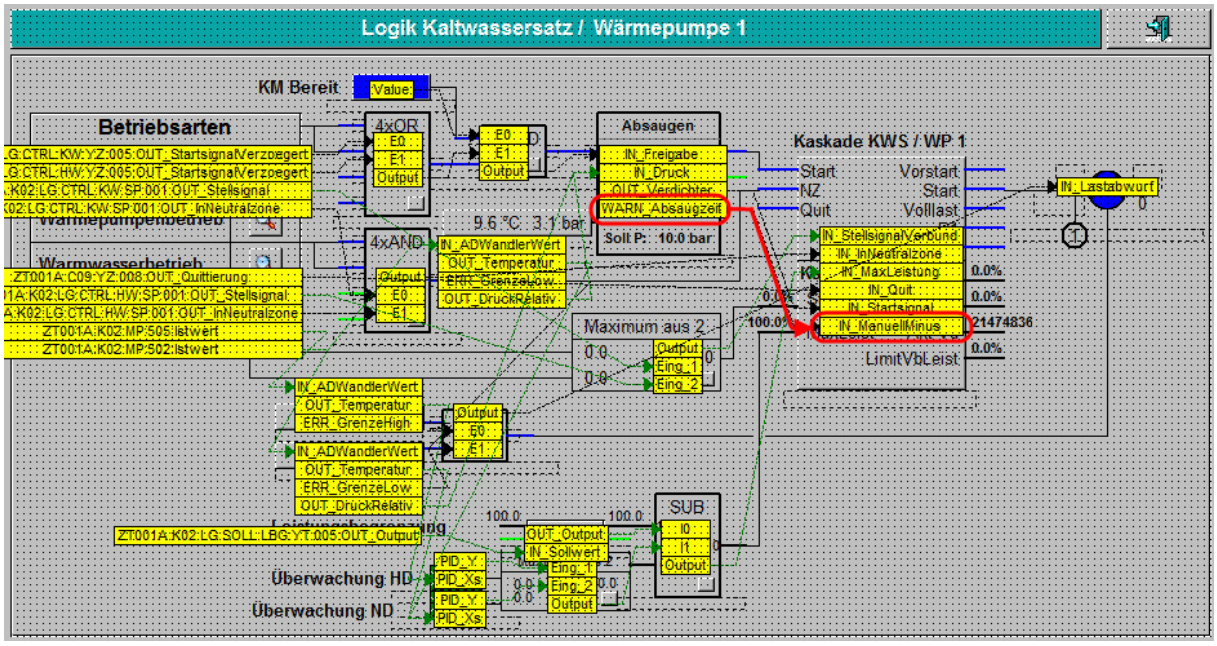

Figure 4.2-6: Logical plant diagram (signals linked)

The colouring of the link lines can be changed by the user. For the above example the colouring is described in the table below:

Colour	Description
Black	Link-type is a binary signal with state 0
light-green	Link-type is a binary signal with state 1
dark-green	Link-type is an analogue signal

Table 4.2-3: Colouring of the link lines

The connection established in the above example will result in the following line of generated IL (Instruction List) code:

```
;*******************************************************************************

        PB              PBscoKaskade ; Manage scoKaskade-Objects

; Kaskade KWS / WP 1 [ZT001A:K02:LG:FGB:YZ:006]

DB      4013    [ ]     1,0,0,60,0,200,0,0,0,
0,3,10,1,3,10,1  ; Kaskade KWS / WP 1
```

```
STH     F __frame.fNull ; IN-PAR
OUT     F K02.LG.FGB.YZ_006.IN_BetrArtLokal

STH     F K02.LG.FGB.YZ_008.WARN_Absaugzeit    ; IN-PAR
OUT     F K02.LG.FGB.YZ_006.IN_ManuellMinus

STH     F K02.LG.FGB.YZ_002.Output  ; IN-PAR Verknüpfung Neutralzone Leistungsregler
OUT     F K02.LG.FGB.YZ_006.IN_InNeutralzone

COPY    R K02.LG.LAW.CL_001.Output ; IN-PAR Berechnung maximale Leistung
        R K02.LG.FGB.YZ_006.IN_MaxLeistung

STH     F C09.YZ_008.OUT_Quittierung          ; IN-PAR Signalisation
OUT     F K02.LG.FGB.YZ_006.IN_Quit

STH     F K02.LG.FGB.YZ_008.OUT_Verdichter          ; IN-PAR Absaugsteuerung
OUT     F K02.LG.FGB.YZ_006.IN_Startsignal

COPY    R K02.LG.FGB.YZ_005.Output ; IN-PAR Maximalauswahl Stellgrösse
        R K02.LG.FGB.YZ_006.IN_StellsignalVerbund


CFB     scoKaskade      ; Scheco Kaskaden-Modul zur Verdichter-Freigabe
        DB      4013    ; [=01] CFG_AnzVd
        F       __frame.fNull    ; [=02] CFG_KkTyp
        F       1440    ; [=03] ERR_AlarmP1
        F       1441    ; [=04] ERR_AlarmP2
        F       C09.YZ_008.OUT_Quittierung  ; [=05] HW_QuitEing
        F       1737    ; [=06] IN_BetrArtLokal
        F       1747    ; [=07] IN_InNeutralzone
        F       1749    ; [=08] IN_ManuellMinus
        F       1750    ; [=09] IN_ManuellPlus
        R       1509    ; [=10] IN_MaxLeistung
        R       1510    ; [=11] IN_Modus
        F       1755    ; [=12] IN_Quit
        F       1761    ; [=13] IN_Startsignal
        R       1520    ; [=14] IN_StellsignalVerbund
        R       2147    ; [=15] OUT_AktVd
```

```
F       1800    ; [=16] OUT_DelayOn
R       1572    ; [=17] OUT_IstLeist
R       1584    ; [=18] OUT_pDbRemVdRec
R       1585    ; [=19] OUT_pDbRemVdSend
R       1591    ; [=20] OUT_Sollleist
R       2148    ; [=21] OUT_TotalVbLeistung
F       1830    ; [=22] OUT_VerbundEin
F       1832    ; [=23] OUT_Volllast
DB      __PDB+8         ; [=24] Datablock
DB      __PDB+9         ; [=25] Datablock[8]
DB      __PDB+17        ; [=26] Datablock[8]
DB      __PDB+25        ; [=27] Datablock[8]

EPB                     ; End Programblock scoKaskade

;*****************************************************************************
```

Listing 4.2-1: Generated Function Block call with parameters

4.3 Current logical template objects

In Figure 4.2-1: Logical plant diagram (runtime mode) LTO (Logical TO) can be seen (also see Table 4.2-1: Description of the TO line 3). These are the logical TO used at the moment.

LTO Name	Description
AND02	AND gating with two inputs
AND04	AND gating with four inputs
ORH02	OR gating with two inputs
ORH04	OR gating with four inputs

Table 4.3-1: Current LTO

They have some major disadvantages which are the fixed number of inputs, the fixed digital technology functions and the cost of resources. These facts make it difficult to increase the number of input or to change the digital technology function if necessary. The current LTO has been implemented to enable programming the plant logic on the plant diagram. They are similar to convenient classes in an application framework such as Qt. Hence they are implemented in a very common way which

applies input and output resources for each instance used. The current implementation also contains some engineering inconveniences. If one needs to increase the number of used inputs on a LTO where all inputs are already in use, an additional LTO is needed or, in case the maxed LTO is one with only two inputs, the LTO has to be changed. The same process takes place if the digital technology function has to be changed. This slows the engineering process significantly. To avoid this, the basic digital technology functions (AND, OR) have to be available as one logic TO with up to ten inputs with the possibility to alter the digital technology function at runtime.

In the life cycle of a project the dependencies of an input parameter can change. To make a program modification as simple as possible, the number of used input parameters of a logical TO should be easy to increment or decrement respectively. It should be possible to easily add or remove connections and afterwards download the edited program to the running PLC. To realize this and to make the programming of a facility on the plant diagram more efficient, the logic template object is necessary. One possible solution of an implementation of such a template object on a PLC-System of Saia-Burgess is shown in the following section. The template objects can be modelled according to the UML 2.0 standard Kecher (2006). Some attempts have been made to use UML to generate PLC code similar to Sacha (2005) and Lee *et al.* (2002) but have been abandoned because it is not in the scope of this investigation.

4.4 System design

As can be derived from the objectives in Chapter 1.3 Objectives the LTO will be run on a virtual machine. This is a completely new approach. On PLC-Systems from Saia-Burgess due to compatibility reasons the PLC software also runs on a virtual machine. This fact ensures the usability, reusability of existing software on new hardware. Only the software libraries have to be updated and the software has to be compiled for the new target hardware to reuse the software of an existing plant while migrating to a new hardware generation.
Thus the virtual machine developed in this investigation runs inside another virtual machine.

4.4.1 System overview

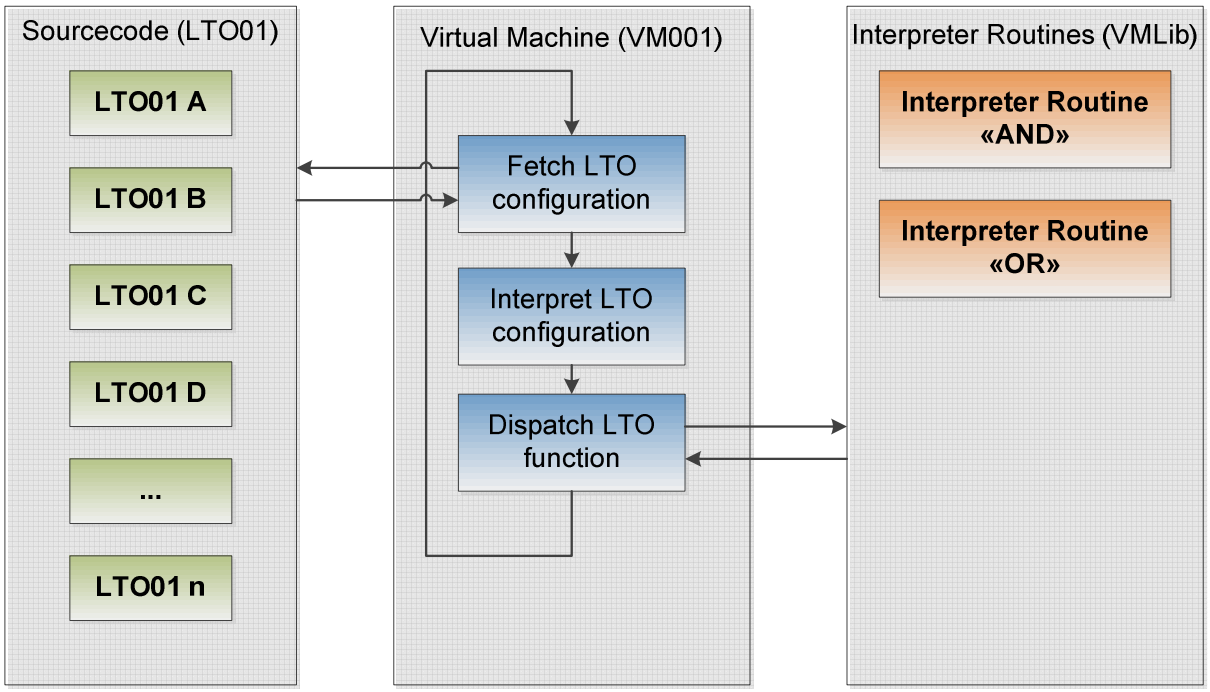

Figure 4.4-1: System overview

4.4.2 Storage of the data

The connection data is stored in appropriate data structures. The data structure used depends on the PLC-System. For this investigation, PLC-Systems from Saia-Burgess are used exclusively. On Saia-Burgess PLC-Systems, data blocks are used to store the generated code. A data block is a one-dimensional array, consisting of 32-bit registers. The data type usually used with data blocks is DWU (Double Word Unsigned). 7999 data blocks are available with up to 16384 32-bit registers.

The data blocks from 0 up to 3999 can only store 383 32-bit values. Because of hardware reasons the access to these data blocks is significantly slower than to the ones in the higher memory area. For this reason, the data blocks starting from 4000 are used.

4.5 Virtual machine on the PLC-System

In Chapter 2.3 Virtual machines the different types of virtual machines are discussed. For this investigation only process virtual machines are of importance. A process

virtual machine deals with one single task. This is also true for the virtual machine developed in this investigation.

4.5.1 Process virtual machine

A case in point for a process virtual machine is the java vm. The byte code produced by the java language processor is interpreted in the virtual machine as a single process. The source code can be interpreted in different ways, Smith &Nair (2005).

- Decode-and-dispatch interpretation
- Indirect threaded interpretation
- Predecoding and direct threaded interpretation
- Binary translation

For the implementation on a PLC-System only the first and second type are of importance because no transformation from one ISA (Instruction Set Architecture) into another is made. Below these two types are discussed.

4.5.2 Decode and dispatch Interpretation

A decode and dispatch interpreter is structured around a central loop that decodes an instruction and then dispatches it to an interpreter routine based on the type of instruction. This kind of interpretation involves a lot of branches, which can be a very time consuming task. There is an interpreter routine for every source instruction.

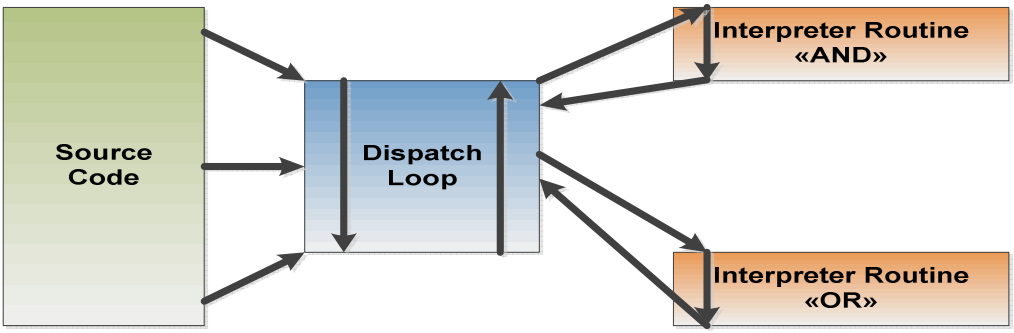

Figure 4.5-1; Decode and dispatch interpretation

4.5.3 Indirect threaded interpretation

In the indirect threaded interpreter the central loop is omitted (see Figure 4.5-1; Decode and dispatch interpretation). To every interpreter routine the necessary code to fetch and dispatch the next instruction is added.

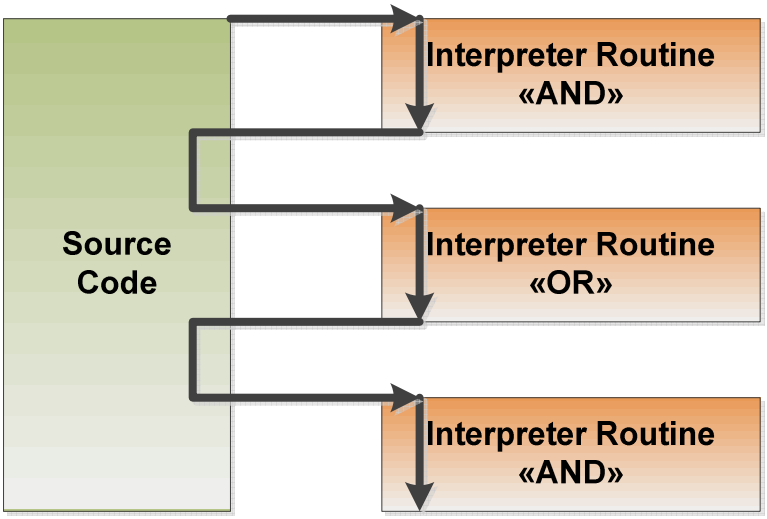

Figure 4.5-2: Indirect threaded Interpretation

4.5.4 Chosen type of virtual machine

For this investigation, the *decode and dispatch interpretation* is used. The solution implemented uses the ideas of this type of virtual machine. It has to be adapted to the needs of PLC-Systems. The interpreter routines are implemented as functional blocks according to the IEC 61131-3 norm, IEC (2003).

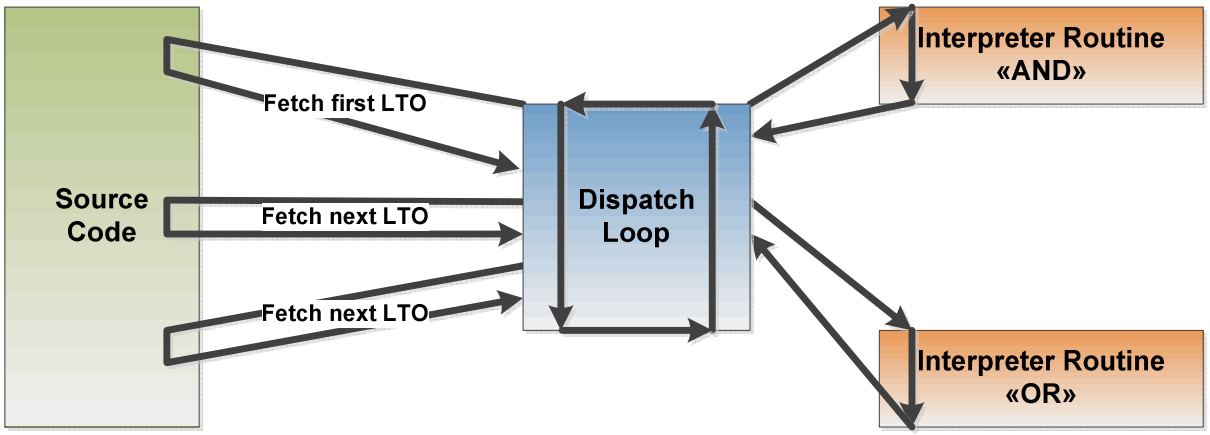

Figure 4.5-3: Implemented virtual machine

4.5.5 Design of the virtual machine

The virtual machine has been developed as a TO. It is called VM001. The VM001 TO consists of 21 parameters.

	Parameter-Name	Parameter-Description	Datatype
1	CFG_CycleTime	Cycletime of the VM	DWU
2	CFG_LTO_Cnt	Number of processed LTO's	DWU
3	CFG_Reserve2	CFG_Reserve2	DWU
4	CFG_Reserve3	CFG_Reserve3	DWU
5	CFG_Reserve4	CFG_Reserve4	DWU
6	CFG_Reserve5	CFG_Reserve5	DWU
7	CFG_Reserve6	CFG_Reserve6	DWU
8	CFG_Reserve7	CFG_Reserve7	DWU
9	CFG_Reserve8	CFG_Reserve8	DWU
10	CFG_Reserve9	CFG_Reserve9	DWU
11	CFG_LTO_Pointer0	Pointer to the LTO 0	DWU
12	CFG_LTO_Pointer1	Pointer to the LTO 1	DWU
13	CFG_LTO_Pointer2	Pointer to the LTO 2	DWU
14	CFG_LTO_Pointer3	Pointer to the LTO 3	DWU
15	CFG_LTO_Pointer4	Pointer to the LTO 4	DWU
16	CFG_LTO_Pointer5	Pointer to the LTO 5	DWU
17	CFG_LTO_Pointer6	Pointer to the LTO 6	DWU
18	CFG_LTO_Pointer7	Pointer to the LTO 7	DWU
19	CFG_LTO_Pointer8	Pointer to the LTO 8	DWU
20	CFG_LTO_Pointer9	Pointer to the LTO 9	DWU
21	CFG_Pointer	Pointer tot he configuration	FLT

Table 4.5-1: VM01: Parameters

- Parameters 1 to 10 are used for configuration purposes or are reserved for future development.
- Parameters 11 to 20 are used to store the pointers to the configuration data blocks of the LTOs processed by the VM001 TO. By means of these pointers the VM001 TO can fetch the necessary data from the LTO linked to it. All configuration data and the LTO link information are stored in the configuration data block of the VM001 TO.
- Parameter 21 represents the pointer to configuration data and link information. The pointer can be used to alter the configuration of the VM001 at runtime.

4.5.6 Implementation of VM001

The VM001 TO acts as a decode and dispatch loop in the system. To simplify the implementation the interpreter routines are placed in a separate library file, VMLib.src (also see Appendix B Source code VMLib.src).

4.6 The new logical template object

A new LTO (Logical Template Object), the LTO01, has been developed to remove the above discussed disadvantages. With the newly developed LTO it is possible to change the digital technology function at runtime and the number of inputs has been increased to ten. Of course this increased number of inputs does not completely remove the possibility of running out of unused inputs but it minimises it. In the following section the LTO01 TO is outlined in depth.

4.6.1 Design of the LTO01

The basic design can be seen in the table below.

	Parameter-Name	Parameter-Description	Datatype
1	CFG_Function	Logical function	DWU
2	CFG_NbrOfUsedInputs	Number of used Inputs	DWU
3	CFG_Reserve2	CFG_Reserve2	DWU
4	CFG_Reserve3	CFG_Reserve3	DWU
5	CFG_Reserve4	CFG_Reserve4	DWU
6	CFG_Reserve5	CFG_Reserve5	DWU
7	CFG_Reserve6	CFG_Reserve6	DWU
8	CFG_Reserve7	CFG_Reserve7	DWU
9	CFG_Reserve8	CFG_Reserve8	DWU
10	CFG_Reserve9	CFG_Reserve9	DWU
11	CFG_AddressInput0	Addresse input 0	DWU
12	CFG_AddressInput1	Addresse input 1	DWU
13	CFG_AddressInput2	Addresse input 2	DWU
14	CFG_AddressInput3	Addresse input 3	DWU
15	CFG_AddressInput4	Addresse input 4	DWU
16	CFG_AddressInput5	Addresse input 5	DWU
17	CFG_AddressInput6	Addresse input 6	DWU
18	CFG_AddressInput7	Addresse input 7	DWU
19	CFG_AddressInput8	Addresse input 8	DWU
20	CFG_AddressInput9	Addresse input 9	DWU
21	CFG_LogicInput0	Logic of input 0	DWU
22	CFG_LogicInput1	Logic of input 1	DWU
23	CFG_LogicInput2	Logic of input 2	DWU
24	CFG_LogicInput3	Logic of input 3	DWU
25	CFG_LogicInput4	Logic of input 4	DWU
26	CFG_LogicInput5	Logic of input 5	DWU

27	CFG_LogicInput6	Logic of input 6	DWU
28	CFG_LogicInput7	Logic of input 7	DWU
29	CFG_LogicInput8	Logic of input 8	DWU
30	CFG_LogicInput9	Logic of input 9	DWU
31	CFG_AddressOutput	Addresse output	DWU
32	CFG_AddressOutputInvers	Addresse invers output	DWU
33	IN_Input0	Logical input 0	BIT
34	IN_Input1	Logical input 1	BIT
35	IN_Input2	Logical input 2	BIT
36	IN_Input3	Logical input 3	BIT
37	IN_Input4	Logical input 4	BIT
38	IN_Input5	Logical input 5	BIT
39	IN_Input6	Logical input 6	BIT
40	IN_Input7	Logical input 7	BIT
41	IN_Input8	Logical input 8	BIT
42	IN_Input9	Logical input 9	BIT
43	OUT_Output	Output of LTO	BIT
44	OUT_OutputInvers	Invers output of LTO	BIT
45	CFG_Pointer	Pointer tot he configuration	FLT

Table 4.6-1: LTO01: Parameters

The LTO01 TO acts as a container for the data used by the VM001 TO. In the whole system the LTO01 TO represents the source code which in turn represents the plant logic. For the design of the new LTO some fundamental changes have been made to reduce the cost of resources. In the old LTO the input signals use their own PLC resources which are binary signals, called flags in Saia-Burgess PLC-Systems. As can be seen in Chapter 4.2.2 Programming on the plant diagram a binary conjunction results in the following line of IL code:

```
STH     F K02.LG.FGB.YZ_008.WARN_Absaugzeit      ; IN-PAR
OUT     F K02.LG.FGB.YZ_006.IN_ManuellMinus
```

Listing 4.6-1: Generated IL code for a binary conjunction

With this code generation concept the cost of binary resources is quite huge. For one simple binary conjunction two binary signals are necessary. The new design removes this waste of binary resources, because almost every binary input signal is a binary output signal of another TO. Instead of copying the binary state from the output signal of TO A to the input signal of LTO B a pointer to the output signal of TO A as input signal for LTO B can be used.

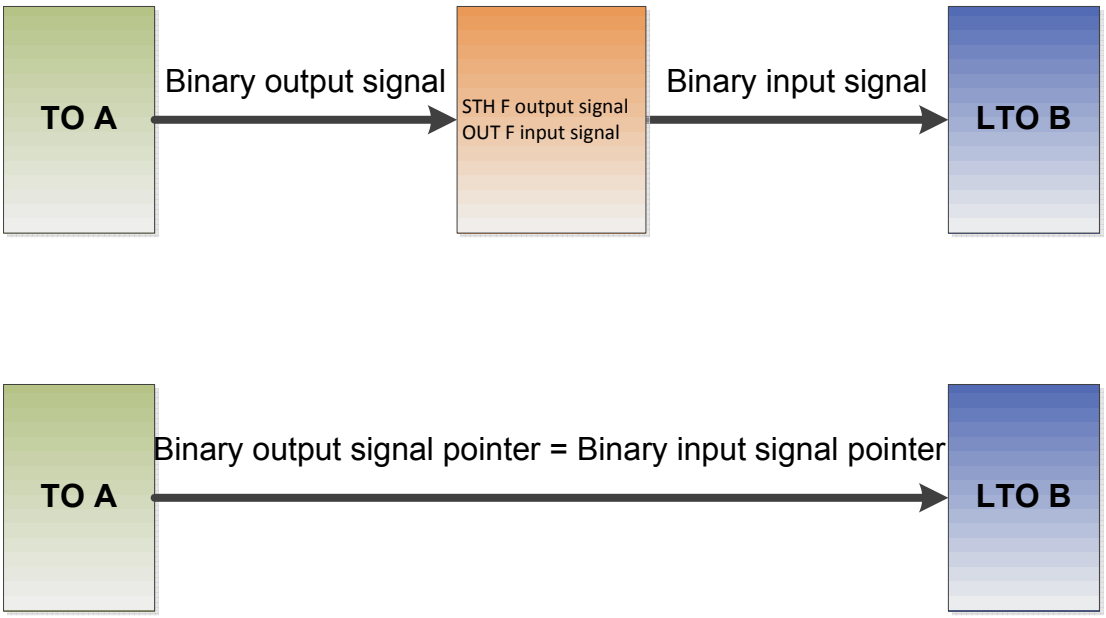

Figure 4.6-1: Reduce cost of binary resources

The LTO01 TO implements the above mentioned concept to reduce the cost of binary resources. The input signals are simple containers for the pointers to the particular output signals.

All configuration data is stored in a datablock (see Table 4.6-1: LTO01: Parameters).

- Parameters 1 to 10 are used for configuration purposes or are reserved for future development.
- Parameters 11 to 20 are used to store the input pointers to the appropriate output signals.
- Parameters 21 to 30 contain the information which binary state of the respective input signal is to be used.
- Parameters 31 and 32 hold the pointers to the output signal and to the inverted output signal respectively.
- Parameters 33 to 42 are the containers for the input pointers.
- Parameters 43 and 44 are the real output signals which can be used to establish conjunctions to other TO.
- Parameter 45 holds the pointer to the LTO01's configuration datablock. This parameter has to be linked to the VM (VM001 TO for the VM to be granted access to the data of the LTO01.

4.6.2 Implementation of LTO01

At the moment the LTO01 executes two basic digital technology functions. These are the AND-Function and the OR-Function. For each one of the ten input signals the used binary state can be adjusted by the values stored in parameters 21 to 30 of the configuration data block. The digital technology function and the number of input signals used can be parameterised with parameter 1 and 2 of the configuration data block. The LTO01 requires the programmer to use the input signals in sequence starting from input signal 1 to 10 to allow for the possibility of changing the number of used input signals at runtime.

4.7 Implemenation of the interpreter routines

As mentioned above, 4.5.6 Implementation of VM001, to keep the VM001 as simple as possible, the interpreter routines have been implemented in a separate library file (also see Appendix B Source code VMLib.src). For each implemented digital technology function a function block has been developed. A uniform interface has been designed for the function blocks.

4.7.1 Functionblock interface fbAnd

```
; ***************** Function: AND ******************
; ***************** Functionblock ******************

                FB      fbAnd                   ; Call Function AND
rLtoAddress             DEF  = 1                ; Pointer to the actual LTO
rInputAddresses         DEF = 2                 ; Pointer to the input addresses of the actual LTO
rInputLogics            DEF = 3                 ; Pointer to the input logic's of the actual LTO
rOutputAddresses        DEF = 4                 ; Pointer to the output addresses of the actual LTO
```

Listing 4.7-1: Functionblock interface fbAnd

4.7.2 Functionblock interface fbOr

```
; ***************** Function: OR *******************
; ***************** Functionblock ******************

                FB      fbOr                    ; Call Function OR
rLtoAddress             DEF  = 1                ; Pointer to the actual LTO
```

```
rInputAddresses     DEF = 2          ; Pointer to the input addresses of the actual LTO
rInputLogics        DEF = 3          ; Pointer to the input logic's of the actual LTO
rOutputAddresses    DEF = 4          ; Pointer to the output addresses of the actual LTO
```

Listing 4.7-2: Functionblock interface fbOr

4.7.3 Functionblock interface fbXor (skeleton only)

```
; ***************** Function: XOR ******************
; ***************** Functionblock ******************

            FB      fbXor            ; Call Function XOR
rLtoAddress         DEF  = 1         ; Pointer to the actual LTO
rInputAddresses     DEF = 2          ; Pointer to the input addresses of the actual LTO
rInputLogics        DEF = 3          ; Pointer to the input logic's of the actual LTO
rOutputAddresses    DEF = 4          ; Pointer to the output addresses of the actual LTO
```

Listing 4.7-3: Functionblock interface fbXor

4.7.4 Real local variables (temporary data)

To minimise the cost of resources a new feature using real local variables has been extensively applied. The temporary data has to be defined inside the block which is using them. As it is common practice with local variables in high level languages as C++, the variable exists only inside this block. The type of block is irrelevant for the behaviour of the temporary data.

```
; ***************** Local Variables *****************
FUNCTION_OR_LOCALS:
fInputs             TEQU F [10]              ; Temporary Flags
fLogics             TEQU F [10]              ; Temporary Flags
fInputsLinked       TEQU F [10]              ; Temporary Flags
fHfOutput           TEQU F                   ; Temporary Flag
fTempLogic          TEQU F                   ; Temporary Flag
rInputCount         TEQU R                   ; Temporary Register
rHrLtoAddress       TEQU R                   ; Temporary Register
rSaveTempIndex      TEQU R                   ; Temporary Register
rSaveInputAdresses  TEQU R                   ; Temporary Register
rSaveInputLogics    TEQU R                   ; Temporary Register
FUNCTION_OR_END_LOCALS:
; ***************** Local Variables *****************
```

Listing 4.7-4: Temporary data

4.7.5 Get the number of used inputs

By means of the first parameter of the functionblock, *rLtoAddress* (see 4.7.1 Functionblock interface fbAnd) the interpreter routine accesses the configuration data of the according LTO and gets the number of used inputs. To access the LTO data block a firmware function call is made (CSF; Call Special Function).

```
; ***************** Get Input Count ******************
FUNCTION_AND_GET_CNT:

        COPY  rLtoAddress                    ; Copy lto-address
              rHrLtoAddress                  ; to helper register

        CSF   S.SF.DBLIB.Library             ; Library number
              S.SF.DBLIB.GetDBItem           ; Read a single DB item
              rHrLtoAddress                  ; 1 R|K IN, DB number (any DB number)
              1                              ; 2 R|K IN, DB item
              rInputCount                    ; 3 R OUT, Value read

        SUB   rInputCount                    ; Subtract from InputCount
              1                              ; the constant 1
              rInputCount                    ; and save the value


FUNCTION_AND_END_GET_CNT:
; ************* End Get Input Count ******************
```

Listing 4.7-5: Get input count

4.7.6 Process the digital technology function

The processing of the digital technology functions is making heavy use of the index register of the Saia-Burgess PLC-System. In a first step the different indexes are created and saved.

```
; ***************** Process function *****************
FUNCTION_AND_PROCESS:
; ********** Get Inputs

        SEI   0                                  ; Set index to 0
        STI   rSaveTempIndex                     ; save index
        SEI   rInputAddresses                ; Set index register to the base adresse of the input
        STI   rSaveInputAdresses                 ; save index
        SEI   rInputLogics           ; Set index register to the base adresse of the input-logics
        STI   rSaveInputLogics                   ; save index
```

Listing 4.7-6: Prepare indexes

In the second step the state of the input is checked and saved to a temporary flag.

```
FUNCTION_AND_INPUT_LOOP:
```

```
        RSI     rSaveInputAdresses          ; restore index
        STHX    F 0                         ; If input is high

        RSI     rSaveTempIndex              ; restore index
        OUTX    fInputs                     ; set flag to high too
```

Listing 4.7-7: Check state of the input

In the third step the information which binary state of the input should be used in the conjunction is retrieved.

```
; ********** Get Logics
FUNCTION_AND_LOGIC_LOOP:

        RSI     rSaveInputLogics            ; restore index

        BITOX   1                           ; Get one bit
                R 0                         ; out of the register
                F fTempLogic                ; onto the flag
        STI     rSaveInputLogics            ; save index

        RSI     rSaveTempIndex              ; restore index

        STH     fTempLogic                  ; if temporary flag is high
        OUTX    fLogics                     ; set flag to high too

        STI     rSaveTempIndex              ; save index

        RSI     rSaveInputLogics            ; restore index
        INI     8191                        ; increment index
        STI     rSaveInputLogics            ; save index

        RSI     rSaveTempIndex              ; restore index
        INI     rInputCount                 ; increment index
        STI     rSaveTempIndex              ; save index

        GETX    rInputAddresses             ; copy address of next input
                rSaveInputAdresses          ; to register

        JR      H FUNCTION_AND_INPUT_LOOP   ; jump to loop if there are more inputs
```

Listing 4.7-8: Get logic information

In the fourth step the binary input state used in the conjunction is calculated by means of the binary XOR function.

A	B	Y ($Y := A \underline{V} B$)
0	0	0
0	1	1
1	0	1
1	1	0

Table 4.7-1: Truth table for the XOR-Function

A := State of the input signal

B := State of the configuration signal

Y := Result of the XOR function and used state in the conjunction

```
; ********** Link input-logic

        SEI     0                                   ; Set index to 0

FUNCTION_AND_INPUTLOGIC_LOOP:

        STHX    fInputs                             ; Link inputs
        XORX    fLogics                             ; with logic
        OUTX    fInputsLinked                       ; and save it to a new array

        INI     rInputCount                         ; increment index
        JR      H FUNCTION_AND_INPUTLOGIC_LOOP; jump to loop if more inputs to link
```

Listing 4.7-9: Calculate the binary state used in the conjunction

In the fifth step the processing of the digital technology function is done.

```
; ********** Process function

        SEI     0                                   ; Set index to 0

        ACC     H                                   ; Set accu to high state
        SET     fHfOutput                           ; Set helper flag to high state
FUNCTION_AND_FUNCTION_LOOP:

        STH     fHfOutput                           ; Link the state of the helper flag with the
        ANHX    fInputsLinked                       ; state of the linked input
        OUT     fHfOutput                           ; set the helper flag to the state of the input

        INI     rInputCount                         ; increment index
        JR      H FUNCTION_AND_FUNCTION_LOOP; jump to loop if more inputs to check
```

Listing 4.7-10: Processing of the digital technology function

In the sixth and last step of the interpreter routine, the output signals of the LTO are set accordingly to the result of the conjunction.

```
; ********** Set outputs

        SEI     rOutputAddresses            ; Set index to the base address of the output array

        STH     fHfOutput                       ; If the helper flag is high
        OUTX    F 0                             ; then set the output to high too
        OUTLX F 1                           ; and set the inverted output to low (and vice versa)

FUNCTION_AND_END_PROCESS:
; ************* End Process function ****************
```

Listing 4.7-11: Set the output of the LTO according to the conjunction result

4.8 The Server Application pTool

4.8.1 pTool basics

The Server Application pTool is design as a System-Tray-Icon application. That means no window is visible at runtime but a symbol in the system tray indicates that the application is running.

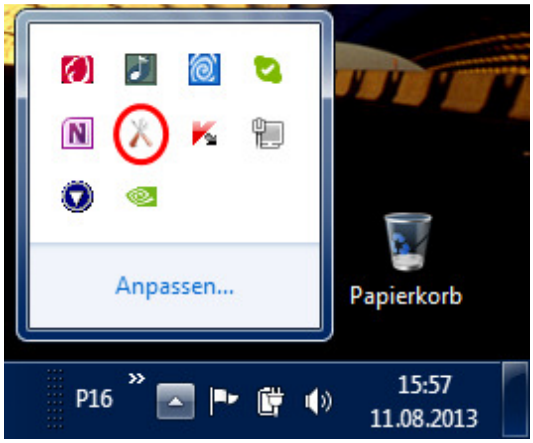

Figure 4.8-1: System-Tray-Icon of the pTool application

By means of a right-click on the system-tray-icon with the mouse the context menu of the application can be reached.

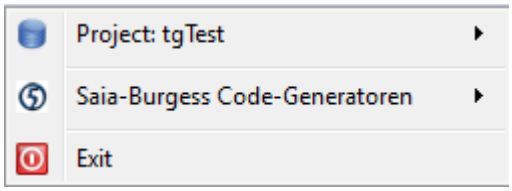

Figure 4.8-2: Context menu of the pTool application

A Project can be opened, closed or the project dialog can be opened via the project context menu.

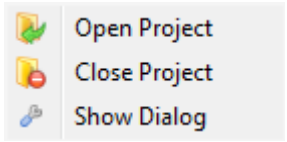

Figure 4.8-3: Project context menu

In the project dialog the project tree with all the necessary data for the generator application can be seen.

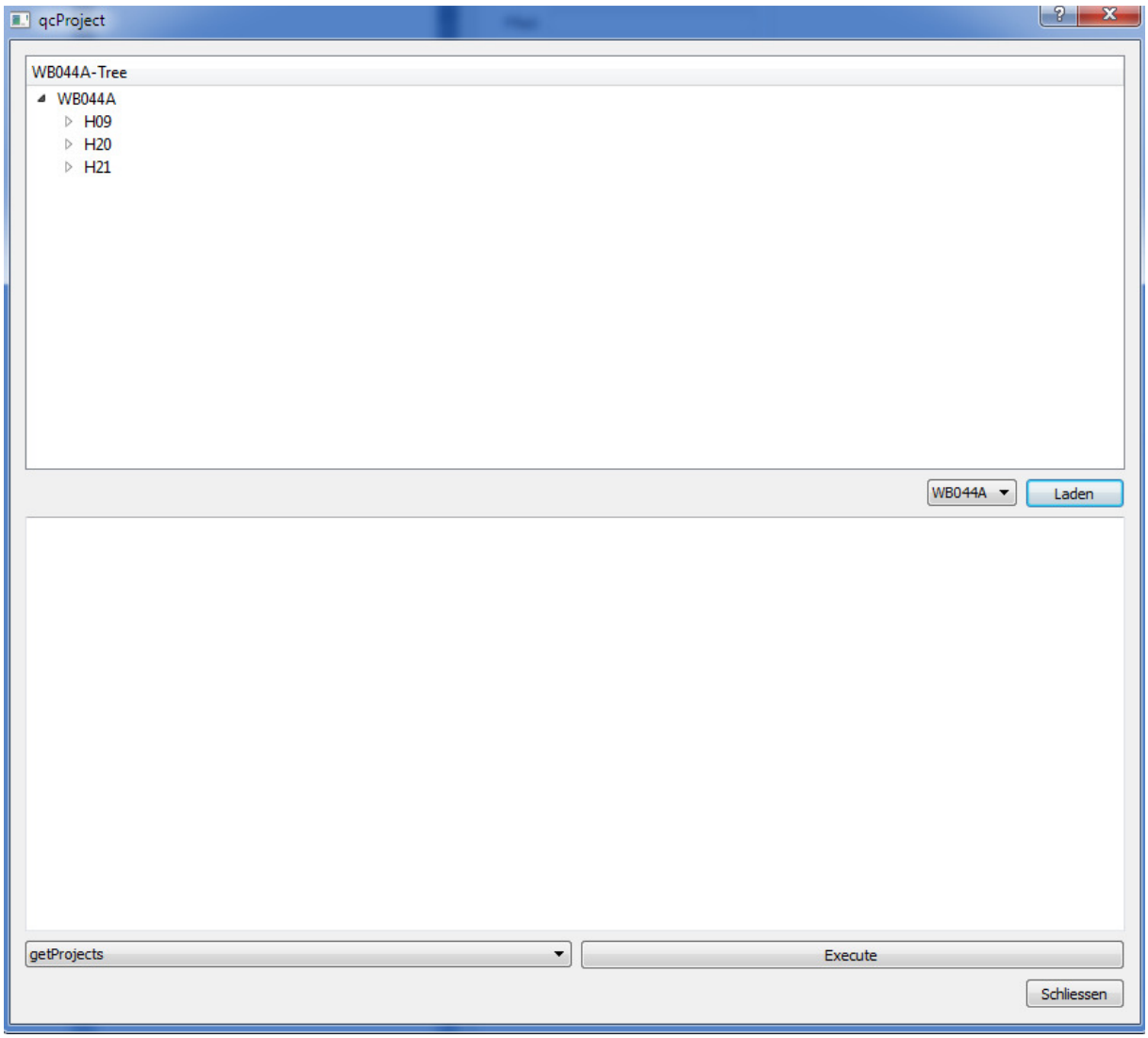

Figure 4.8-4: pTool: Project dialog

The data tree is composed of information that is read from the SCADA definition file. This file contains all the information about the project. The path to the file is passed to the pTool application at startup or if an open project is closed and another one is opened. If changes in the SCADA system are made, the pTool application has to be refreshed since its data-tree is in fact a mirrored image of the SCADA systems project tree.

4.8.2 Implementation overview

In this section a short implementation overview is given for the pTool-Application. The pseudocode uses the conventions introduced by Cormen *et al.* (2009) in their book about algorithms.

4.8.2.1 Load project

```
If !qsProjectPath.isEmpty()
	// Read System Tree
	readTree(qsProjectPath, "System", "pSystemTree.dms", *qfSystemTree)

	// Get the Cpu's
	qslCpus = getCpus(qsProjectPath)

	// Read Cpu tree's
	For i = 0 to i qslCpus.size()
		// Create a new QFile object
		qfCpuTree = new QFile

		// read Cpu-Tree to file
		readTree(qsProjectPath, qslCpus.at(i), qslCpus.at(i) % ".dms",
*qfCpuTree)

	// Fill CPU's into combobox
	qslTempList = qslCpus
	qslTempList.prepend("System")
	ui.qcbTree->addItems(qslTempList)

	// Load files into models
	createTrees()

	// Set return value
	iRetValue = 0

else iRetValue = -1

// Return value
return iRetValue
```

Listing 4.8-1: load project (pseudocode)

4.8.2.2 Execute command

```
// Check Command
iCommand = struInCmd->qsCommand.toInt()

// execute commands
switch(iCommand)
	// Send Project List
	case 0:
		// Clear Answer list
		qslAnswer.clear()

		// Sending Project list
		qvScratch = qslProjectNames.size()
		qslAnswer << qvScratch.toString()
		for i = 0 to qslProjectNames.size()
			qslAnswer << qslProjectNames.at(i)
		break

	// Send CPU List
	case 1:
		// Clear Reslut list
		qhProjects.value(struInCmd->qsProject)->qslResults.clear()

		// Clear Answer list
		qslAnswer.clear()

		// Execute Command
		if executeCommand(struInCmd->qsCpu, iCommand) == 0
			// Sending CPU list
			for i = 0 to qslResults.size()
				qslAnswer << qslResults.at(i)
		else << "Getting CPU's doesn't work"
	break

	// Send TO List
	case 2:
		// Clear Reslut list
		qhProjects.value(struInCmd->qsProject)->qslResults.clear()

		// Clear Answer list
		qslAnswer.clear()

		// Execute Command
		if executeCommand(struInCmd->qsCpu, iCommand) == 0
			// Sending CPU list
			for  i = 0 to qslResults.size(); ++i)
				qslAnswer << qslResults.at(i)
		else << "Getting TO's doesn't work"
	break

	// Send Instance List
	case 3:
		// Clear Reslut list
		qhProjects.value(struInCmd->qsProject)->qslResults.clear()

		// Clear Answer list
		qslAnswer.clear()

		// Execute Command
		if executeCommand(struInCmd->qsCpu, iCommand) == 0
		 // Sending CPU list
			for i = 0 to qslResults.size(); ++i)
```

```
                        qslAnswer << qslResults.at(i)

        else << "Getting Instances's doesn't work"
break

// Send Data
case 4:
        // Clear Reslut list
        qhProjects.value(struInCmd->qsProject)->qslResults.clear()

        // Clear Answer list
        qslAnswer.clear()

        // Execute Command
        if executeCommand(struInCmd->qsCpu, iCommand) == 0
                // Sending CPU list
                for i = 0 to qslResults.size()
                        qslAnswer << qslResults.at(i)
        else << "Getting instances by name doesn't work"
break

// Send Data
case 5:
        // Clear Reslut list
        qhProjects.value(struInCmd->qsProject)->qslResults.clear()

        // Clear Answer list
        qslAnswer.clear()

        // Execute Command
        if executeCommand(struInCmd->qsCpu, iCommand) == 0
                // Sending CPU list
                for i = 0 to qslResults.size()
                        qslAnswer << qslResults.at(i)
        else << "Getting Data doesn't work"
break

// Send Comment
case 6:
        // Clear Reslut list
        qhProjects.value(struInCmd->qsProject)->qslResults.clear()

        // Clear Answer list
        qslAnswer.clear()

        // Execute Command
        ifexecuteCommand(struInCmd->qsCpu, iCommand) == 0
                // Sending CPU list
                for i = 0 to qslResults.size(); ++i)
                        qslAnswer << qslResults.at(i)
        else << "Getting Comment doesn't work"
break

// Send Attribute Comment
case 7:
        // Clear Reslut list
        qhProjects.value(struInCmd->qsProject)->qslResults.clear()

        // Clear Answer list
        qslAnswer.clear()

        // Execute Command
        if executeCommand(struInCmd->qsCpu, iCommand) == 0)
```

```
                    // Sending CPU list
                    for i = 0 to qslResults.size()
                          qslAnswer << qslResults.at(i)
             else << "Getting Attribute Comment doesn't work"
      break

      // Send Project path
      case 8:
             // Clear Reslut list
             qhProjects.value(struInCmd->qsProject)->qslResults.clear()

             // Clear Answer list
             qslAnswer.clear()

             // Execute Command
             if executeCommand(struInCmd->qsCpu, iCommand) == 0)
                    // Sending CPU list
                    for i = 0 to qslResults.size()
                          qslAnswer << qslResults.at(i)
             else << "Getting Project Path doesn't work"
      break

      // Send Data for tree
      case 9:
             // Clear Reslut list
             qhProjects.value(struInCmd->qsProject)->qslResults.clear()

             // Clear Answer list
             qslAnswer.clear()

             // Execute Command
             if executeCommand(struInCmd->qsCpu, iCommand) == 0)
                    // Sending CPU list
                    for i = 0 to qslResults.size(); ++i)
                          qslAnswer << qslResults.at(i)
             else << "Getting Project Path doesn't work";
      break;

      // When others
      default:
             // Debug output
             "pTool: Unknow Command"

             // clear answer list
             qslAnswer.clear()
      break
```

Listing 4.8-2: execute command (pseudocode)

4.8.3 pTool commands

The pTool application accepts several commands from a client application (e.g. sGen). By means of these commands the client application can retrieve information from the server.

Command	Description
getProjects	Returns the projects managed by the pTool application
getCpus	Returns the cpus of a given project
getTos	Returns the TOs of a given project
getInstances	Returns plant TOs of a given project
getObjectsByType	Returns plant TOs with the TO-Type of a given project
getData	Returns project data of a given project
getComment	Returns the comment for each plant TO of a given project
getAttribComment	Returns the comment for the attributes of the plant TOs
getProjectPath	Returns the project path
getDataForTree	Returns the data necessary to build the generator tree

Table 4.8-1: pTool commands

4.8.4 The local server

The pTool application provides a local socket based server. This enables a client application to establish a connection to the pTool application and to send commands through a local socket. When the pTool application is starting up, the server is listening for incoming connections. Currently the server only accepts a single connection.

4.8.5 The configuration file pTool.ini

The relevant data is saved in a standard configuration file (Appendix J Configuration files). No data is written into the windows registry. This has been done to ensure the transfer to other operating systems such as Linux.

4.9 The Client Application, the code generator, sGen

4.9.1 sGen basics

The SCADA-System programming language is a kind of functional block language. The logic operations are created in a special programming mode while drawing the process charts. The conjunction data is saved in the project definition file. There are two kinds of conjunction parameters, input and output parameters respectively. The parameters are stored as shown in Figure 4.9-1 Storage of connection parameters (e.g. PAR_IN):

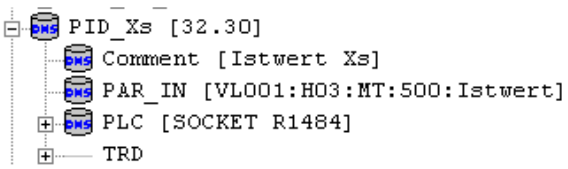

Figure 4.9-1 Storage of connection parameters (e.g. PAR_IN)

The sGen application establishes a connection to the pTool applications local socket server. The connection is made tvia a local socket, which is in fact a named pipe on a Windows system. Through the local socket the sGen application can retrieve the necessary data using the above described commands (Table 4.8-1: pTool commands). The retrieved data is utilized to create a dependency tree. By means of the dependency tree the PLC code will be generated. Sacha K. (2005) shows a way to automatically generate code for PLC-Systems. Unfortunately the PLC language generated is ladder logic. That is not the best choice for complex systems because of its origin as a way to copy the electrical wiring diagram into the PLC-System programming tool. Therefore ST (structured text) or IL (instruction list) should be the languages generated. In this investigation the language generated is Saia-Burgess IL. For PLC-Systems which provide ST, this should be the language generated. Because of its similarities to high level languages the generated code will be easier to understand for third party personnel.

4.9.2 The developed dll's

To simplify the sGen application a number of DLLs, dynamic link libraries, have been developed. DLLs were first introduced by Microsoft in their OS2 operating system and are Microsoft's version of the shared library concept. In this investigation all functionality that is thought to be of use in the implementation of a code generator for a different PLC-System has been implemented as a DLL. Thus another application can use the DLL too.

DLL-Name	Description
plib	Functions to retrieve data from the SCADA definition file
pTSocketLib	Implementation of a IPC-System using QT's Local Sockets
sGenLib	Functions used to generate code for Saia-Burgess PLC
uLib	Utillity library with functions in connection with QT-Widgets

Table 4.9-1: Developed DLLs

4.9.3 The debug window

To monitor the behaviour of the application a debug window and a custom message handler has been implemented to show the outputs generated at runtime.

```
void debugOutput(QtMsgType type, const char* msg) {
	// Create static Message Browser and set it up
	static QTextBrowser* qtbMessageBrowser = new QTextBrowser();
	qtbMessageBrowser->setWindowTitle("Debug Window");
	qtbMessageBrowser->move(500, 0);
	qtbMessageBrowser->show();

	// Check which message type it is
	switch(type) {
		// Debug message
		case QtDebugMsg:
			qtbMessageBrowser->append(QString("Debug : %1").arg(msg));
			break;
		// Warning
		case QtWarningMsg:
			qtbMessageBrowser->append(QString("Warning: %1").arg(msg));
			break;
		// Critical
		case QtCriticalMsg:
			qtbMessageBrowser->append(QString("Critical: %1").arg(msg));
			break;
		// Fatal
		case QtFatalMsg:
			qtbMessageBrowser->append(QString("Fatal: %1").arg(msg));
			break;
	}
}
```

```
int main(int argc, char *argv[]) {

	QApplication app(argc, argv);
	#ifdef Q_OS_WIN
		#ifndef QT_NO_DEBUG_OUTPUT
			qInstallMsgHandler(debugOutput);
		#endif
	#endif
	app.setApplicationName(app.translate("main", "sGen"));
	app.setOrganizationName("tG Soft");
	app.setOrganizationDomain("tGSoft.ch");
	sGen w;
	w.show();
	return app.exec();
}
```

Listing 4.9-1: The main function of the sGen application

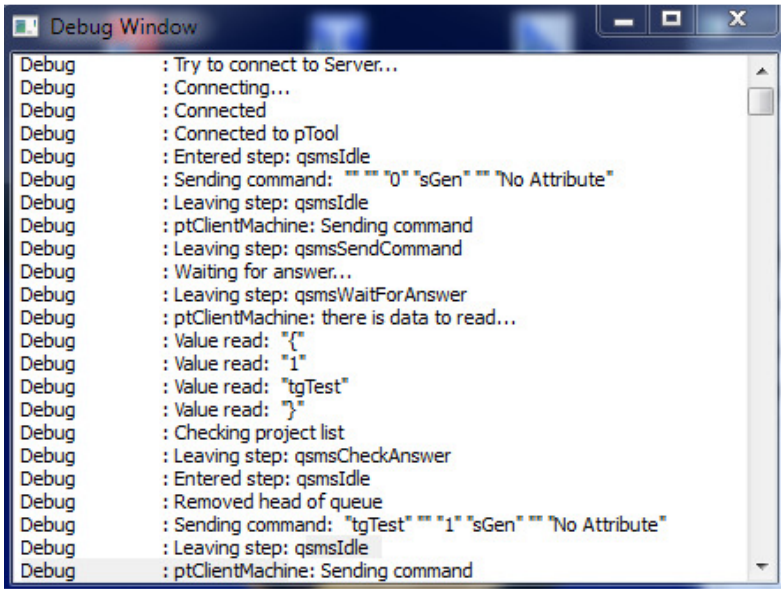

Figure 4.9-2: sGen: Debug window

As can be seen in Listing 4.9-1: The main function of the sGen application, the message handler consists of a message browser window where the different message types produced by the sGen application are displayed. The custom message handler is installed in the main function and can be switched off after testing the application by adding the line:

DEFINES += QT_NO_DEBUG_OUTPUT

in the project file.

4.9.4 The sGen dialog

The control element of the sGen application is the sGen dialog. It consists of three tabs:

1. Data-Tab (Figure 4.9-3: sGen_Dialogue: Data-Tab)
2. Tree-Tab (Figure 4.9-6: sGen-Dialog: Tree tab)
3. Generator-Tab (Figure 4.9-7: sGen-Dialog: Generator tab)

The Data tab is used to establish the connection to the pTool application, to send commands to it and to show the data received from it. In the final version of sGen the connection will be established at startup and the data wil be retrieved automatically. For testing purposes everything is done by hand.

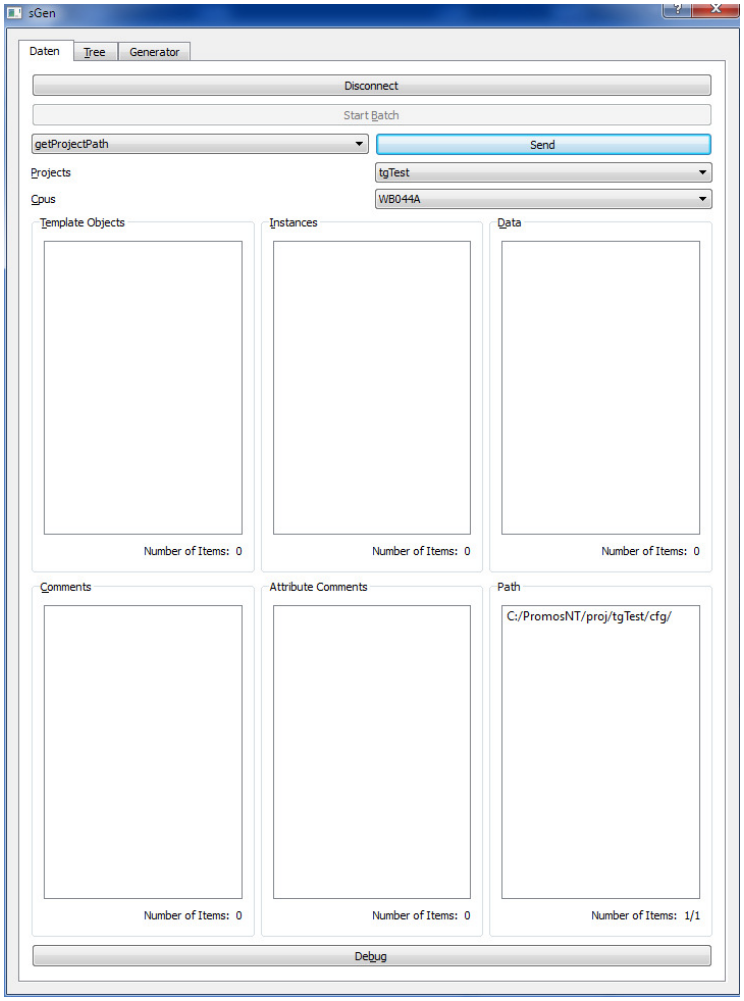

Figure 4.9-3: sGen_Dialogue: Data-Tab

The program flow of the sGen application is shown in Figure 4.9-4: sGen: Program flow.

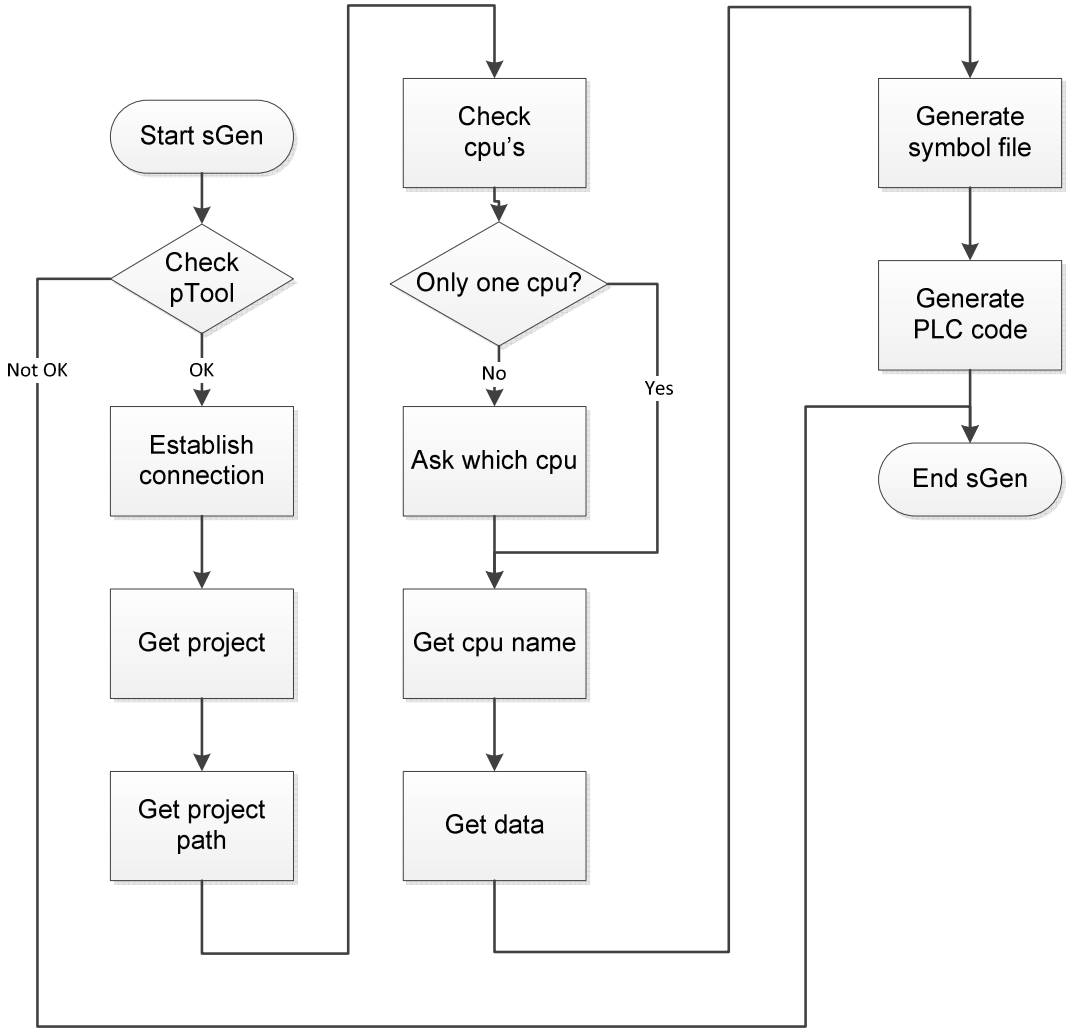

Figure 4.9-4: sGen: Program flow

Figure 4.9-5: sGen: Detail program flow for a command shows the behaviour of the sGen application in case the pTool application can not be reached or a command cannot be executed successfully.

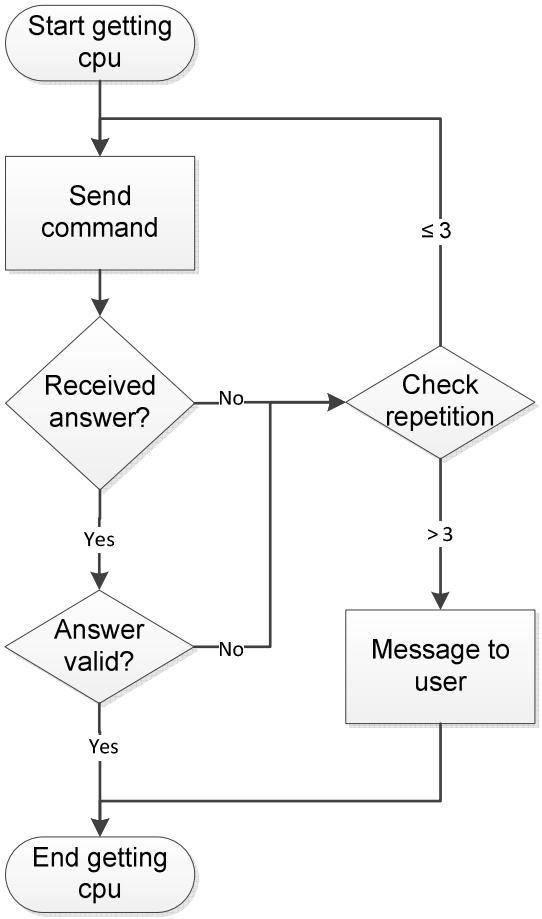

Figure 4.9-5: sGen: Detail program flow for a command

With the command getDataForTree the necessary data to build the dependencies tree can be retrieved from the pTool application. The data is stored in a tree that is visualized in the tree tab of the sGen application. The tree is the basis for the code generator to retrieve the information necessary to generate the symbol file and the PLC code.

Figure 4.9-6: sGen-Dialog: Tree tab

After the dependency tree is built, the PLC software can be generated. The rxp generator is still under construction and is not part of this investigation. The VM generator generates the necessary PLC code to run the LTOs on the VM. The generator is explained in detail in the next section.

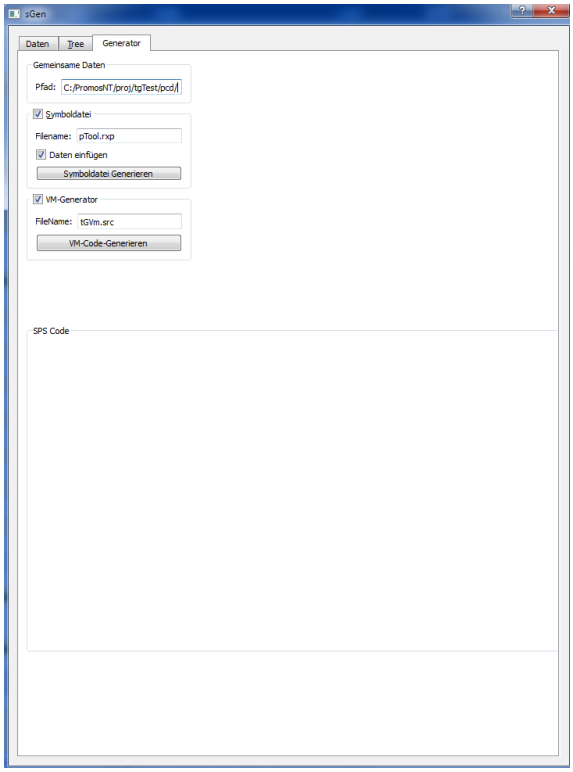

Figure 4.9-7: sGen-Dialog: Generator tab

4.9.5 The VM generator

The VM generator is realised as a finite state machine. The QT framework provides an easy to use state machine framework. The concept of the framework is based on the concepts of Harel (1987) and uses his notations. The semantics for the execution of the state machine are based on State Chart XML (SCXML) W3C (2013). The VM generator state machine consists of five states. It generally uses the *state entered()* function of each step to perform the state tasks. The transition from one state to another is realised with the standard Signal / Slot concept of the QT-Framework Blanchette (2008).

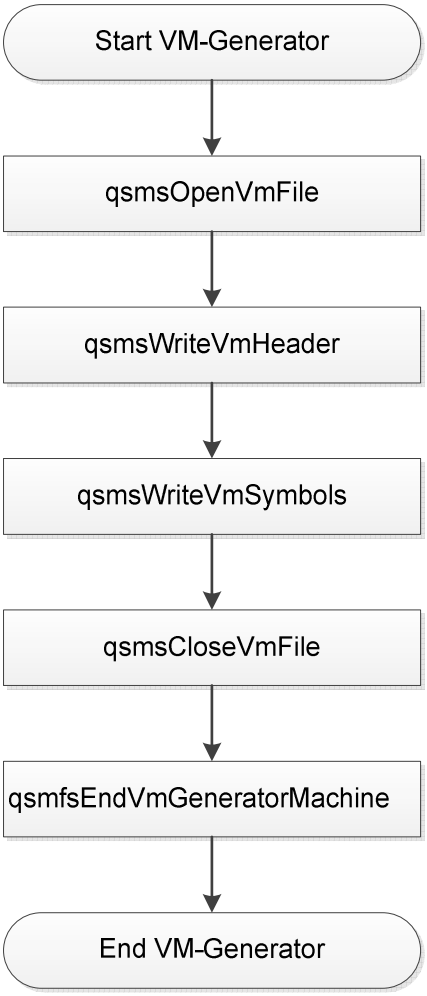

Figure 4.9-8: VM generator state machine

```
void sGen::on_qpbGenerateVm_clicked() {
1       // Locals
2       bool bTest = false;
3
4       // dll einbinden
5       generatorLib = new sGenLib();
6
7       // Generator State machine
8       qsmGeneratorMachine = new QStateMachine(this);
9
10      // Steps of the RXP Generator machine
11      qsmsOpenVmFile = new QState();
12      qsmsWriteVmHeader = new QState();
13      qsmsWriteVmSymbols = new QState();
14      qsmsCloseVmFile = new QState();
15      qsmfsEndVmGeneratorMachine = new QFinalState();
16
17      // Add steps to the data machine
18      qsmGeneratorMachine->addState(qsmsOpenVmFile);
19      qsmGeneratorMachine->addState(qsmsWriteVmHeader);
20      qsmGeneratorMachine->addState(qsmsWriteVmSymbols);
21      qsmGeneratorMachine->addState(qsmsCloseVmFile);
22      qsmGeneratorMachine->addState(qsmfsEndVmGeneratorMachine);
23
24      // Set initial step of the data machine
25      qsmGeneratorMachine->setInitialState(qsmsOpenVmFile);
26
27      // Add transitions to the states of the data machine
28      qsmsOpenVmFile->addTransition(this, SIGNAL(vmFileOpened()), qsmsWriteVmHeader);
29      qsmsWriteVmHeader->addTransition(this, SIGNAL(vmHeaderWritten()),
qsmsWriteVmSymbols);
30      qsmsWriteVmSymbols->addTransition(this, SIGNAL(vmSymbolsWritten()),
qsmsCloseVmFile);
31      qsmsCloseVmFile->addTransition(this, SIGNAL(vmFileClosed()),
qsmfsEndVmGeneratorMachine);
32
33      // Establish connections of the states of the data machine
34      bTest = connect(qsmsOpenVmFile, SIGNAL(entered()), this,
SLOT(if_qsmsOpenVmFile_entered()));
36      bTest = connect(qsmsWriteVmHeader, SIGNAL(entered()), this,
SLOT(if_qsmsWriteVmHeader_entered()));
37      bTest = connect(qsmsWriteVmSymbols, SIGNAL(entered()), this,
SLOT(if_qsmsWriteVmSymbols_entered()));
38      bTest = connect(qsmsCloseVmFile, SIGNAL(entered()), this,
SLOT(if_qsmsCloseVmFile_entered()));
39      bTest = connect(qsmfsEndVmGeneratorMachine, SIGNAL(entered()), this,
SLOT(if_qsmfsEndVmGeneratorMachine_entered()));
40
41      // Start machine
42      qsmGeneratorMachine->start();
}
```

Listing 4.9-2: on_qpbGenerateVm_clicked() function

If the user clicks on the *VM-Code-Generieren* button the on_qpbGenerateVm_clicked() function is called. Here the VM generator state machine is set up.

- In line 8 the state machine is created.
- In lines 11 to 15 the states are created.
- In lines 18 to 22 the states are added to the state machine.
- In line 25 the *qsmsOpenVmFile* state is set as initial step.
- In lines 27 to 31 the transitions are added to the steps, thereby the signal is defined whose emittance causes the source step to switch to the target step.
- In lines 34 to 39 the *entered()* functions of the steps are connected to the appropriate slot (*state_entered()* functions) to perform the state tasks.
- In line 40 the state machine is started with *qsmsOpenVmFile*.

4.9.5.1 Initial state qsmsOpenVmFile

This is the initial state of the VM generator state machine. In this state an existing VM file is opened or a new file is created if no such file exists.

```
void sGen::if_qsmsOpenVmFile_entered() {
1       // Locals
2       QString qsProjectPath;
3
4       // Debug message
5       qDebug() << "Entering step: if_qsmsOpenVmFile_entered";
6
7       // Set path
8       qsProjectPath = ui.qlePlcPath->text();
9
10      if(qsProjectPath.isEmpty() || qsProjectPath.isNull()) {
11              qsProjectPath = QFileDialog::getExistingDirectory(this, tr("Choose the
project path"), "c:/PromosNT/proj/", QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);
12              ui.qlePlcPath->setText(qsProjectPath + "/pcd/");
13              qsProjectPath = ui.qlePlcPath->text();
14      }
15
16      // Set path and filename
17      if(qsProjectPath.isEmpty()) {
18              QMessageBox::warning(this, "sGen", "Please choose a project");
19
20              // Stop machine
21              qsmGeneratorMachine->stop();
22
23              return;
24      }
25      else {
27              // Set input path
28              if(ui.qleVmFileName->text().isEmpty()){
29                      QMessageBox::warning(this, "sGen", "Please insert a filename for
the vm-file");
30
```

```
31            // Stop machine
32            qsmGeneratorMachine->stop();
33
34                  return;
35            }
36            else {
37                  // assemble complete path
38                  qsNameOfOutputFile = qsProjectPath % ui.qcbCpus->currentText() %
"/" % ui.qleVmFileName->text();
39
40                  // Set filename
41                  qfOutput.setFileName(qsNameOfOutputFile);
42
43                  // Open / create new file
44                  qtsOutput.setDevice(&qfOutput);
45                  if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
46                        QMessageBox::warning(0, "sGen", "Error while creating the
vm-file");
47
48                        // Stop machine
49                        qsmGeneratorMachine->stop();
50                  }
51                  else {
52                        // Emite Signal
53                        emit vmFileOpened();
54                  }
55            }
56      }
57      // Return value
58      return;
}
```

Listing 4.9-3: if_qsmsOpenVmFile_entered() function

- In line 2 the local variable qsProjectPath of the type QString is declared.
- In line 5 a debug message is written to the debug window, also see 4.9.3 The debug window.
- In lines 8 to 35 the project path and the file name of the file to generate are checked and, where required, additional information is asked from the user or the state machine is terminated (lines 23 and 34).
- In lines 36 to 50 the output filename is assembled and an attempt to open or create the file is made. If this is unsuccessful the user is notified and the state machine is terminated. If the file is created or opened the *vmFileOpened()* signal is emitted. By emitting this signal the state machine will switch to the next state.

4.9.5.2 qsmsWriteVmHeader

This state writes the file header in the opened file. The design of the header template can be defined in the *srcVmHeader.ini* file. Data that will be filled into the header template is placed in angle brackets (e.q. <DateTime>).

```
void sGen::if_qsmsWriteVmHeader_entered() {
1       // Locals
2       QString qsScratch;
3
4       // Debug message
5       qDebug() << "Entering step: if_qsmsWriteVmHeader_entered";
7
8       // Fill Header data into structure
9       qsSettings->beginGroup("VmTemplates");
10      struHeader.qsHeaderTemplate = qsSettings->value("srcHeader", "No
entry").toString();
11      struHeader.qsPcdType = generatorLib->pcdTyp(ui.qlePlcPath->text(), ui.qcbCpus-
>currentText());
12      struHeader.qsPgVersion = generatorLib->pgVersion(ui.qlePlcPath->text(),
ui.qcbCpus->currentText());
13      struHeader.qsBlockTyp = "COB";
14      struHeader.qsStationNumber = generatorLib->stationNumber(ui.qlePlcPath->text(),
ui.qcbCpus->currentText());
15      struHeader.qsOutputFile  = qsNameOfOutputFile;
16
17      // Write Header
18      if(generatorLib->writeHeader(&struHeader) != 0) {
19              // Show warning
20              QMessageBox::warning(0, "sGen", "Error while writing header, machine will
be stopped");
21
22              // Stop machine
23              qsmGeneratorMachine->stop();
24      }
25
26      // emit signal
27      emit vmHeaderWritten();
28      qsSettings->endGroup();
29
30      // Return value
31      return;
}
```

Listing 4.9-4: The if_qsmsWriteHeader_entered() function

- In line 5 a debug message is written into the debug window indicating that the state has been reached.
- In line 9 the group of the *qsSettings* object is set to *VmTemplates*. The *qsSettings* object uses the *sGen.ini* file to read the necessary settings (see Appendix J II sGen.ini).
- In lines 10 to 15 the necessary data is read from the definition files, from the sGen Gui or from internal variables and is stored in the header structure.

- In lines 18 to 24 an attempt to write the header is made. If this attempt is unsuccessful the user is notified and the state machine is stopped. Otherwise the *vmHeaderWritten()* signal is emitted and the group of the *qsSettings* object is set to default.

Some of the information is retrieved from the configuration files in the Saia-Burgess PLC project. These configuration files are similarly constructed as standard **.ini* files. And therefore a *QSettings* object can be used to retrieve the data.

Filename	Group	Information
CpuName.saia5pc	Device	PG5Version
PCD.SCFG	ConfigNT	PCDType
CpuName.5hw	PersistentSettings	sbus_Station, tcp_DeviceName

Table 4.9-2: Saia-Burgess configuration files

4.9.5.3 qsmsWritteVmSymbols

This is the state where the real work is done. The data tree is searched for the correct data to generate the function block calls and to fill the appropriate data blocks. The processing in this state make heavily use of hashing because of the very fast lookup it is providing and the fact that most of the used data appears in pairs. The generator is divided in two main sections:

1. LTO generation
2. VM generation

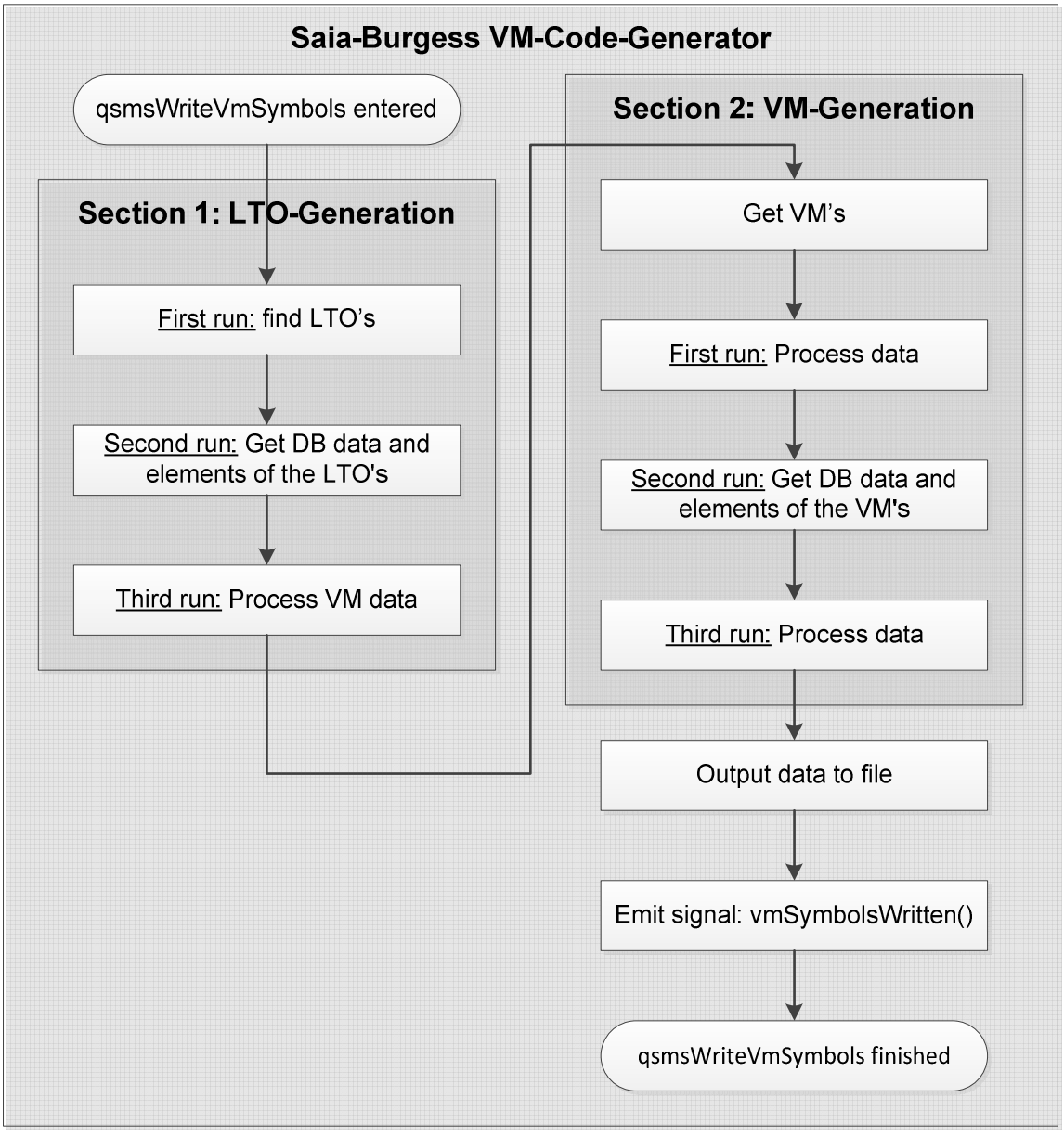

Figure 4.9-9: Saia-Burgess VM code generator

LTO generation: first run

```
////////////////////////////////////////////////////////////////////////////////
1       // Find LTO's
2       qlResult = qhTrees.value("Data-Tree")->findItems("LTO01", Qt::MatchRecursive |
Qt::MatchContains);
3
4       // First Run: Get all LTO's DMS-Names
5       if(!qlResult.isEmpty()) {
6               // Get DMS-Names
7               qslDmsNames = generatorLib->getDmsNames(qlResult);
8       }
9       else {
10              // Debug output
11              qDebug() << "First run: Getting LTO's failed";
12
13              // Return value
14              return;
15      }
////////////////////////////////////////////////////////////////////////////////
```

Listing 4.9-5: LTO generation: first run

- In line 2 all instances of the LTO01 LTO are searched in the tree.
- In lines 5 to 15 the resulting list is checked. If the list is empty a message is posted in the debug window otherwise the DMS-Names of the LTO01-Instances are retrieved from the list.

LTO generation: second run

```
////////////////////////////////////////////////////////////////////////////////
1       // Second run: Get DB data and elements of the LTO's
2       if(!qlResult.isEmpty()) {
3               // Clear List
4               qlHashes2.clear();
5
6               // Get data
7               for(int i = 0; i < qlResult.size(); ++i) {
8                       // Create LTO-Hash
9                       qlHashes2 << generatorLib->createLtoHash(qlResult.at(i)->parent()-
>parent(), qslDmsNames.at(i));
10              }
11      }
12      else {
13              // Debug output
14              qDebug() << "Second run: Getting LTO data failed";
15
16              // Return value
17              return;
18      }
////////////////////////////////////////////////////////////////////////////////
```

Listing 4.9-6: LTO generation: second run

- If the list is not empty (line 2) for each LTO01 instance a list of hashes will be constructed in lines 7 to 10.

- Lines 12 to 18 notify the user that no LTO01 instances have been found.

The *createLtoHash()* function

```
QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
sGenLib::createLtoHash(QStandardItem *qsiResult, QString qsInDmsName) {
1       // Locals
2       QString qsParent, qsScratch1, qsScratch2, qsScratch3;
3       QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
qlReturnHash;
4       QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*> *qmhLtoHash;
5       QMultiHash<QString, QHash<QString, QString>*> *qmhDatapointHash;
6       QHash<QString, QString>     *qhDataHash;
7       QStandardItem *qsiChild1, *qsiChild2, *qsiChild3, *qsiChildAddress;
8       int iRowCount = 0, iRowCount2= 0, iRowCount3= 0;
9
10      // Create a Data-Hash
11      qmhLtoHash = new QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>();
12      qlReturnHash << qmhLtoHash;
13
14      // At first get Parent of the item at i
15      qsParent = qsiResult->text();
16
17      // Check there are children
18      if(qsiResult->hasChildren()) {
19              // Check how many children the item has
20              iRowCount = qsiResult->rowCount();
21              // Check every child to be DB, F or R
22              for(int x = 0; x < iRowCount; ++x) {
23                      // Create Multi-Hashe
24                      qmhDatapointHash = new QMultiHash<QString, QHash<QString,
QString>*>();
25
26                      // Get child on row y, column 0 (allways 0 in tree's)
27                      qsiChild1 = qsiResult->child(x, 0);
28                      qsScratch1 = qsiChild1->text();
29                      // Check if child has children itself
30                      if(qsiChild1->hasChildren()) {
31                              // Check how many children the item has
32                              iRowCount2 = qsiChild1->rowCount();
33                              // Get grand-children
34                              for(int y = 0; y < iRowCount2; ++y) {
35                                      // Get child on row z, column 0 (allways 0 in
tree's)
36                                      qsiChild2 = qsiChild1->child(y, 0);
37                                      qsScratch2 = qsiChild2->text();
38
39                                      // Check if child has children itself
40                                      if(qsiChild2->hasChildren()) {
41                                              // Check how many children the item has
42                                              iRowCount3 = qsiChild2->rowCount();
43                                              // Get grand-children
44                                              for(int z = 0; z < iRowCount3; ++z) {
45                                                      // Get child on row z, column 0
(allways 0 in tree's)
46                                                      qsiChild3 = qsiChild2->child(z, 0);
47                                                      qsScratch3 = qsiChild3->text();
48
49                                                      // Create a Data-Hash
```

```
50                                                  qhDataHash = new QHash<QString,
QString>();
51
52                                                  // Insert data into hash
53                                                  qhDataHash->insert(qsScratch2,
qsScratch3);
54                                          }
55                                  }
56
57                                  // Insert Data-Hash into Datapoint-Hash
58                                  qmhDatapointHash->insert(qsScratch1, qhDataHash);
59                          }
60                  }
61
62                  // Insert Data-Hash into Datapoint-Hash
63                  qmhLtoHash->insert(qsInDmsName, qmhDatapointHash);
64          }
65          // Return value
66          return qlReturnHash;
67      }
68      else {
69          // MessageBox
70          QMessageBox::warning(0, "sGenLib", "Error creating Hash");
71
72          // Clear Hash
73          qlReturnHash.clear();
74
75          // Return value
76          return qlReturnHash;
77      }
78}
```

Listing 4.9-7: The createLtoHash() function

The return value of this function, the constructed list, consists of a complex construct of nested hashes and multi hashes. A multi hash is an extension of a standard hash. A multi hash allows multiple values with the same key.

The function is mapping the necessary parts of the data tree into the generated list. This is done to enable a faster iterating through the data.

LTO generation: third run

All output data is written into a string list and then to the VM file. The code below has been split into several sections to simplify the explanation of the tasks carried out. The code can be reviewed as a whole in Appendix F Sources sGen.

```
////////////////////////////////////////////////////////////////////////////////
       // Third run: process data
       // Generate helper ressources
       qvScratch1 = qlHashes2.size();

       // Fill in Helper register
1      qsSettings->beginGroup("LTO01GEN");
2      qslOutputData << generatorLib->writeHelperRegister(qvScratch1.toInt(),
       qsSettings->value("3", "No entry").toString());
3      qsSettings->endGroup();
```

Listing 4.9-8: Helper registers

- In line 1 the group of the *qsSettings* object is set to *LTO01GEN*
- In line 2 the string that is allocating the necessary amount of helper registers is written to the output string list.
- In line 3 the group of the *qsSettings* object is set to default.

The generated code will look like this:

```
rHrLTO01        EQU     R[4]                                    ; Helper Registers
```

Listing 4.9-9: Generated code for the helper registers

In the next step the DB addresses will be retrieved from the list of hashes in a *for-Loop*.

```
// Get DB-Address
if(!qlHashes2.isEmpty()) {
1      for(int a = 0; a < qlHashes2.size(); ++a) {
2             //
3             qmhLtoHash = qlHashes2.at(a);
4
5             //
6             qvScratch2 = a;
7
8             // get data
9             QMultiHash<QString, QMultiHash<QString, QHash<QString,
              QString>*>*>::const_iterator i;
10            for(i = qmhLtoHash->constBegin(); i != qmhLtoHash->constEnd(); ++i) {
11
12                   // Find DB-Address
```

```
13                  qsSettings->beginGroup("LTO01");
14                  QMultiHash<QString, QHash<QString, QString>*>::const_iterator x =
                    i.value()->find(qsSettings->value("2", "No entry").toString());
15
16                  while (x != i.value()->end() && x.key() == qsSettings->value("2",
                    "No entry").toString()) {
17                          // Get Key
18                          qsScratch2 = x.key();
19
20                          // get data
21                          QHash<QString, QString>::const_iterator y = x.value()-
                            >find("Address");
22
23                          while (y != x.value()->end() && y.key() == "Address") {
24                                  // Get DB Address
25                                  qsAddress = y.value();
26
27                                  // Output data
28                                  qvScratch1 = qlResult.size();
29
30                                  // Insert Data into List
31                                  qslOutputData << "CFG_LTO0" << qvScratch2.toString()
                                    << "\t" << "EQU" << "\t" << "DB " << qsAddress <<
                                    "\t" << "; Datablock for: " << qsScratch1 << "\n" <<
                                    "DB CFG_LTO0" << qvScratch2.toString();
32
33                                  // Move DB Symbol-Name into StringList
34                                  qslLtoDbAdresses << "CFG_LTO0" %
                                    qvScratch2.toString();
35
36                                  // increment iterator
37                                  ++y;
38                          }
39                          // increment iterator
40                          ++x;
41                  }
42                  qsSettings->endGroup();
```

Listing 4.9-10: Find DB address and assemble the allocation string

- In line 1 the for-Loop is started. It will iterate over the list with the LTO01 instances.
- In line 3 the hash for the LTO instance to be examined is retrieved from the list. The control variable is used to determine which LTO instance has to be checked.
- The control variable is saved for further use in line 6. It is saved in a *QVariant* object. A QVariant object holds a single value of a single type at any time but it can be easily converted into numerous types.
- In line 9 an iterator is created to be used to go over the entries of the LTO01 instance hash.
- In line 10 the *for-Loop* that is performing the iterations is started.

- In lines 13 to 42 the data is retrieved from the LTO01 instance hash and the string that allocates the DB in the PLC code is assembled and moved into the output string list (lines 30 – 34).

The generated code will look like this:

```
CFG_LTO00     EQU     DB 4007                         ; Datablock for:
DB CFG_LTO00…..
```

Listing 4.9-11: Generated code for the DB allocation

The next step is involves retrieving the number of elements that are saved in the DB. This is again done by an iterator.

```
// Get number of DB items
1 QMultiHash<QString, QHash<QString, QString>*>::const_iterator d;
2 for(d = i.value()->constBegin(); d != i.value()->constEnd(); ++d) {
3       // Get Key
4       qsScratch2 = d.key();
5
6       // Set Variables
7       qvScratch1 = 0;
8
9       // get data
10      QHash<QString, QString>::const_iterator e = d.value()->find("DBIndex");
11      while (e != d.value()->end() && e.key() == "DBIndex") {
12              if(e.value().toInt() > iDbIndex) {
13                      iDbIndex = e.value().toInt();
14                      qvScratch1 = iDbIndex + 1;
15
16                      // Output data
17                      qsDbItems1 = " [" % qvScratch1.toString() % "] ";
18              }
19              // increment iterator
20              ++e;
21      }
22 }
```

Listing 4.9-12: Retrieving of the number of DB elements

- In line 1 the iterator is created.
- The *for-Loop* that goes over each entry is started in line 2.
- In lines 10 to 21 the hash entries are searched for the key *DBIndex* and a kind of bouble search is performed to find the *DBIndex* with the highest value. The highest value is stored in the *qsDbItems1* variable for further use.

Thereafter the LTO01 instance hash is searched for the configuration data.

```
// Get DB configuration data
1 qsSettings->beginGroup("LTO01GEN");
2 QMultiHash<QString, QHash<QString, QString>*>::const_iterator f = i.value()-
>find(qsSettings->value("1", "No entry").toString());
3 while (f != i.value()->end() && f.key() == qsSettings->value("1", "No
entry").toString()) {
4       // Get Key
5       qsScratch2 = f.key();
6
7       // get data
8       QHash<QString, QString>::const_iterator g = f.value()->find("DatapointValue");
9       while (g != f.value()->end() && g.key() == "DatapointValue") {
10              // Get DB Data
11              qsDbItems2 = g.value();
12
13              // increment iterator
14              ++g;
15      }
16
17      // increment iterator
18      ++f;
19 }
20 qsSettings->endGroup();
```

Listing 4.9-13: Getting the basic digital technology function configured for the LTO01

- In line 1 the *qsSettings* object is set to the appropriate group in the configuration file.
- In line 2 the iterator is created.
- The *while-Loop* that iterates over each entry of the LTO01 instance hash is started in line 3.
- In lines 8 to 19 the value for the data point which is referred to by the *qsSettings* object (1=CFG_Function) is retrieved and stored in the *qsDbItems2* variable for further use.
- Line 20 sets the *qsSettings* object back to the default value.

```
// Get DB configuration data
1 qsSettings->beginGroup("LTO01GEN");
2 QMultiHash<QString, QHash<QString, QString>*>::const_iterator ff = i.value()-3
>find(qsSettings->value("2", "No entry").toString());
3 while (ff != i.value()->end() && ff.key() == qsSettings->value("2", "No
entry").toString()) {
4       // Get Key
5       qsScratch2 = ff.key();
6
7       // get data
8       QHash<QString, QString>::const_iterator gg = ff.value()->find("DatapointValue");
9       while (gg != ff.value()->end() && gg.key() == "DatapointValue") {
10              // Get DB Data
11              qsDbItems3 = gg.value();
```

```
12            qsDbItems3 = ", " % qsDbItems3;
13
14            // increment iterator
15            ++gg;
16      }
17
18      // increment iterator
19      ++ff;
20  }
21  qsSettings->endGroup();
```

Listing 4.9-14: Getting number of used inputs

- In line 1 the *qsSettings* object is set to the appropriate group in the configuration file.
- In line 2 the iterator is created.
- The *while-Loop* that goes over each entry of the LTO01 instance hash is started in line 3.
- In lines 8 to 20 the value for the data point which is referred to by the *qsSettings* object (2=CFG_NbrOfUsedInputs) is retrieved and stored in the *qsDbItems3* variable for further use.
- In line 12 a comma is prepended to the retrieved value.
- Line 21 sets the *qsSettings* object back to the default value.

The pointers to the input addresses are retrieved from the LTO01 instance hashes in the next step.

```
// Insert Pointers to Input addresses
1  if(qsScratch2.contains("IN_Input")) {
2       qsScratch = qsScratch2;
3  }
4
5  QMultiHash<QString, QHash<QString, QString>*>::const_iterator ffff = i.value()-
>find(qsScratch);
6  while (ffff != i.value()->end() && ffff.key() == qsScratch) {
7       // Get Key
8       qsScratch3 = ffff.key();
9
10      // get data
11      QHash<QString, QString>::const_iterator gggg = ffff.value()->find("PAR_IN");
12      while (gggg != ffff.value()->end() && gggg.key() == "PAR_IN") {
13            // Get DB Data
14            qsScratch4 = gggg.value();
15
16            // Check string
17            if(qsScratch4.contains("Logical Input")) {
18                  // insert 0
19                  qsDbItems4.prepend(", 0");
20            }
```

```
21            else {
22                  // Check string
23                  if(qsScratch4.section('.', 0, 0) == "F") {
24                        // Insert flag-address
25                        qsDbItems4.prepend(", " % qsScratch4.section('.', 1, 1));
26                  }
27
28                  else {
29                        // Get last bit of the DMS-Name
30                        qsDmsNameLastPart = qsScratch4.section(':', -1, -1);
```

Listing 4.9-15: Add the pointer to the input addresses to the DB

- In lines 1 to 3 the variable *qsScratch2* is checked for its content and the variable *qsScratch* is set accordingly.
- In line 5 the outer iterator is created.
- In line 6 the *while-Loop* that goes over the entries of the LTO01 instance hash is started.
- In line 11 the retrieval of the values tagged by the search string *PAR_IN* is started.
- In line 14 a value is retrieved from the hash and stored in the variable *qsScratch4*.
- Starting from line 16 the retrieved value is checked for the various possible contents.
- Lines 17 to 20 checking the value to be unused, which means a placeholder (*Logical Input*) has been retrieved.
- Starting from line 21 the content is tested to be a flag address (line 22 to 26) where the flag address is retrieved directly (line 25).
- In line 28 the content is expected to be a DMS-Name (e.g. *WI065:H02:PW:001:Running*), so the last part of the DMS-Name (*Running*) is used to search for its pointer indirectly.

In the next step the data tree is searched for the entries corresponding to the last part of the DMS-Name found above.

```
// Search for address
1 qlResultAddress = qhTrees.value("Data-Tree")->findItems(qsDmsNameLastPart,
Qt::MatchRecursive | Qt::MatchContains);
2
3 if(!qlResultAddress.isEmpty()) {
4      for(int i = 0; i < qlResultAddress.size(); ++i) {
5            // Assemble DMS-Name
```

```
6             // At first get Parent of the item at i
7             qsiResultAddress = qlResultAddress.at(i);
8
9             // Check there is a parent
10            qsiResult = qsiResultAddress;
11                  if(qsiResult != 0) {
12                        while(qsiResult->parent()) {
13                              qsDmsName.prepend(qsiResult->text() % ":");
14                              qsiResult = qsiResult->parent();
15                        }
16                  }
17                  // Move DMS-Name into string list
18                  qsDmsName.remove(qsDmsName.lastIndexOf(":"), 1);
19
20                  // Check DMS-Name
21                  if(qsDmsName == qsScratch4) {
22                        if(qsiResultAddress->hasChildren()) {
23                              //
24                              QString qsTest2 = qsiResultAddress->text();
25                              iRowCount = qsiResultAddress->rowCount();
26
27                              // Get Child
28                              qsiChild1 = qsiResultAddress->child(0, 0);
29                              QString qsTest3 = qsiChild1->text();
30                              if(qsiChild1->text() == "Address") {
31                                    // Check if child has children itself
32                                    if(qsiChild1->hasChildren()) {
33                                    // Get Address, which is the child's text
34                                          qsiChildAddress = qsiChild1->child(0,
                                            0);
35                                          qsAddress = qsiChildAddress->text();
36                                          qslLtoInputAddresses << qsAddress;
37
38                                          // break loops
39                                          i = qlResultAddress.size();
40                                    }
41                              }
42                        }
43                        else {
44                              // insert 0
45                              qsDbItems4.prepend(", 0");
46                        }
47                  }
48
49                  // Clear string
50                  qsDmsName.clear();
51            }
52      }
53
54 // insert Address
55 qsDbItems4.prepend(", " % qsAddress);
56 }
57 }
58 // increment iterator
59 ++gggg;
60 }
61 // increment iterator
62 ++ffff;
63 }
```

Listing 4.9-16: Retrieving the address indirectly via the DMS-Name

- In line 1 the data tree is searched for entries containing the search string retrieved in the previous step.
- In line 3 the result is checked for content.
- In line 4 the retrieval of the pointer is started.
- In lines 5 to 18 the DMS-Name of an entry is assembled.
- In lines 20 to 55 the pointer is retrieved and stored in the *qsDbItems4* variable and a comma is prepended. If no pointer is found a 0 is stored and a comma is prepended.

In the next step the polarities of the input signals are retrieved and stored in the DB.

```
// Insert Polarities of the inputs
1 if(qsScratch2.contains("CFG_LogicInput")) {
2      qsScratch = qsScratch2;
3 }
4
5 QMultiHash<QString, QHash<QString, QString>*>::const_iterator fffff = i.value()-
>find(qsScratch);
6 while (fffff != i.value()->end() && fffff.key() == qsScratch) {
7      // Get Key
8      qsScratch3 = fffff.key();
9
10     // get data
11     QHash<QString, QString>::const_iterator ggggg = fffff.value()-
       >find("DatapointValue");
12     while (ggggg != fffff.value()->end() && ggggg.key() == "DatapointValue") {
13            // Get DB Data
14            qsScratch4 = ggggg.value();
15            qsDbItems5.prepend(", " % qsScratch4);
16
17            // increment iterator
18            ++ggggg;
19     }
20
21     // increment iterator
22     ++fffff;
23 }
24 }
```

Listing 4.9-17: Inserting of the input polarities into the DB

- In lines 1 to 3 the variable *qsScratch2* is checked for its content and the variable *qsScratch* is set accordingly.
- In line 5 the outer iterator is created.
- In line 6 the *while-Loop* that goes over the entries of the LTO01 instance hash is started.

- In line 11 the retrieval of the values tagged by the search string *DatapointValue* is started.
- In line 14 a value is retrieved and stored in the variable *qsScratch4*.
- In line 5 the *qsScratch4* variable is stored in the *qsScratch5* variable and a comma is prepended.

In the last step of the data retrieval for the LTO01 instances the unused configuration data points and the output pointers in the db are filled with 0 values. At the end of that task three *new-line* characters are inserted.

```
// Insert Reserve Configuration Datapoints
for(int i = 0; i < 8; ++i) {
	qsDbItems3.append(", 0");
}

// Insert Output pointers
for(int i = 0; i < 2; ++i) {
	qsDbItems6.append(", 0");
}

// Insert NewLine
qsDbItems6.append("\n\n\n");
```

Listing 4.9-18: Final task of the LTO01 data retrieval

Finally the generated code for the DB allocation is written into the output string list and the variables are cleared so that they can be used again.

```
// Append DB-Data to the StringList
qslOutputData << qsDbItems1;
qslOutputData << qsDbItems2;
qslOutputData << qsDbItems3;
qslOutputData << qsDbItems4;
qslOutputData << qsDbItems5;
qslOutputData << qsDbItems6;

// Clear Strings
qsDbItems1.clear();
qsDbItems2.clear();
qsDbItems3.clear();
qsDbItems4.clear();
qsDbItems5.clear();
qsDbItems6.clear();

// Set Variables
iDbIndex = 0;
```

Listing 4.9-19: Write the generated code into the output string list

The generated code will look like this:

```
CFG_LTO00    EQU    DB 4007                              ; Datablock for:
DB CFG_LTO00 [32] 1, 9, 0, 0, 0, 0, 0, 0, 0, 0, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0

CFG_LTO01    EQU    DB 4008                              ; Datablock for: Register
DB CFG_LTO01 [32] 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0

CFG_LTO02    EQU    DB 4000                              ; Datablock for: Register
DB CFG_LTO02 [32] 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2200, 300, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0

CFG_LTO03    EQU    DB 4001                              ; Datablock for: Register
DB CFG_LTO03 [32] 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2201, 301, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0
```

Listing 4.9-20: Generated DB allocation code

The next step is to generate the functional block calls for each of the found LTO01 instances with the appropriate data.

```
// Insert LTO FB-Call
// Get all keys
1 qsSettings->beginGroup("LTO01");
2 qslKeys = qsSettings->allKeys();
3 qsSettings->endGroup();
4
4 // Get number of keys
5 iKeyCount = qslKeys.size();
6
7 // Insert Data
8 for(int i = 1; i <= iKeyCount; ++i) {
9      //
10     qsSettings->beginGroup("LTO01");
11     qvScratch1 = i;
12     qsScratch1 = qsSettings->value(qvScratch1.toString(), "No entry").toString();
13
14     qsSettings->endGroup();
15
16     // Assemble the data
17     switch(qvScratch1.toInt()) {
```

```
18            // Header
19            case 1:
20                  qsGeneratorTemplate.append(qsScratch1 % "\n");
21            break;
22
23            // DB Address
24            case 2:
25
26            // DB Address (for Pointer)
27            case 3:
28                  qsScratch1 = qsScratch1 %"\t; [" % qvScratch1.toString() % "]" %
                    qsScratch1  % "\n";
29                  qsGeneratorTemplate.append("DB " % qslLtoDbAdresses.at(a) % "; ["
                    % qvScratch1.toString() % "]" % "DB Address"  % "\n");
30            break;
31
32            // Configuration Pointer
33            case 4:
34
35            // Output Address
36            case 5:
37            // OutputInvers Address
38            case 6:
39            // Output Address (for Pointer)
40            case 7:
41            // OutputInvers Address (for Pointer)
42            case 8:
43                  qsScratch1 = qsScratch1 % "\t; [" % qvScratch1.toString() % "]" %
                    qsScratch1  % "\n";
44                  qsScratch3 = generatorLib->makeSymbol(qslDmsNames.at(a),
                    qsScratch1);
45                  qsGeneratorTemplate.append(qsScratch3);
46            break;
47
48            // Helper Register
49            case 9:
50                  qsSettings->beginGroup("LTO01GEN");
51                  qsGeneratorTemplate.append(qsSettings->value("3", "No
                    entry").toString() % "+" % qvScratch2.toString() % "\t; [" %
                    qvScratch1.toString() % "]" % qsScratch1  % "\n\n\n");
52                  qsSettings->endGroup();
53            break;
54
55            // Error
56            default:
57                  qsGeneratorTemplate.append("Something is wrong\n");
58      }
59 }
60 }
```

Listing 4.9-21: Generate the FB calls

To generate the functional block parameters in the correct way, a *switch-statement* is used. This simplifies the code significantly because of a feature omitting the *break-statement* between the *case-statements* of the *switch-statement* in C++. As can be seen in the above listing case 2 and case 3 use the same code to generate their

functional block parameters as do cases 4 to 8. Finally, as shown below, the footer will be generated.

```
// Footer
qsSettings->beginGroup("LTO01FOOTER");
qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString() % "\n\n\n");
qsSettings->endGroup();
}
else {
	// Debug output
	qDebug() << "Third run: processing LTO data failed";

// Return value
	return;
}
////////////////////////////////////////////////////////////////////////////////
```

Listing 4.9-22: LTO01 Generator: Insert footer

The generated code looks like this:

```
; ********** Call FB

	CFB	LTO01				; Call LTO01
		DB CFG_LTO00			; [2]DB Address
		DB CFG_LTO00			; [3]DB Address
		H09.LG.CTRL_001.CFG_Pointer	; [4]CFG_Pointer
		H09.LG.CTRL_001.OUT_Output	; [5]OUT_Output
		H09.LG.CTRL_001.OUT_OutputInvers ; [6]OUT_OutputInvers
		H09.LG.CTRL_001.OUT_Output	; [7]OUT_Output
		H09.LG.CTRL_001.OUT_OutputInvers ; [8]OUT_OutputInvers
		rHrLTO01+0			; [9]Register


; ********** Call FB

	CFB	LTO01				; Call LTO01
		DB CFG_LTO01			; [2]DB Address
		DB CFG_LTO01			; [3]DB Address
		H09.LG.CTRL_002.CFG_Pointer	; [4]CFG_Pointer
		H09.LG.CTRL_002.OUT_Output	; [5]OUT_Output
		H09.LG.CTRL_002.OUT_OutputInvers ; [6]OUT_OutputInvers
		H09.LG.CTRL_002.OUT_Output	; [7]OUT_Output
		H09.LG.CTRL_002.OUT_OutputInvers ; [8]OUT_OutputInvers
```

```
                rHrLTO01+1                          ; [9]Register


; ********** Call FB

        CFB     LTO01                               ; Call LTO01
                DB CFG_LTO02                        ; [2]DB Address
                DB CFG_LTO02                        ; [3]DB Address
                H20.LG.CTRL.LOGIC_001.CFG_Pointer;  [4]CFG_Pointer
                H20.LG.CTRL.LOGIC_001.OUT_Output    ; [5]OUT_Output
                H20.LG.CTRL.LOGIC_001.OUT_OutputInvers ; [6]OUT_OutputInvers
                H20.LG.CTRL.LOGIC_001.OUT_Output    ; [7]OUT_Output
                H20.LG.CTRL.LOGIC_001.OUT_OutputInvers ; [8]OUT_OutputInvers
                rHrLTO01+2                          ; [9]Register


; ********** Call FB

        CFB     LTO01                               ; Call LTO01
                DB CFG_LTO03                        ; [2]DB Address
                DB CFG_LTO03                        ; [3]DB Address
                H21.LG.CTRL.LOGIC_001.CFG_Pointer   ; [4]CFG_Pointer
                H21.LG.CTRL.LOGIC_001.OUT_Output    ; [5]OUT_Output
                H21.LG.CTRL.LOGIC_001.OUT_OutputInvers ; [6]OUT_OutputInvers
                H21.LG.CTRL.LOGIC_001.OUT_Output    ; [7]OUT_Output
                H21.LG.CTRL.LOGIC_001.OUT_OutputInvers ; [8]OUT_OutputInvers
                rHrLTO01+3                          ; [9]Register


END_LTO:
; ****************** Ende LTO's **********************
```

Listing 4.9-23: Code generated by the LTO instance generator

VM generation: first run

The first step in generating the PLC code for the VM is to search the data-tree for instances of the VM001 TO.

```
////////////////////////////////////////////////////////////////////////////////
// Find VM's
1 qlResult = qhTrees.value("Data-Tree")->findItems("VM001", Qt::MatchRecursive |
Qt::MatchContains);
////////////////////////////////////////////////////////////////////////////////
2
3
////////////////////////////////////////////////////////////////////////////////
4       // First run: process vm data
5       if(!qlResult.isEmpty()) {
6               // Get DMS-Names
7               qslDmsNames.clear();
8               qslDmsNames = generatorLib->getDmsNames(qlResult);
9       }
10      else {
11              // Debug output
12              qDebug() << "First run: processing VM data failed";
13
14              // Return value
15              return;
16      }
////////////////////////////////////////////////////////////////////////////////
```

Listing 4.9-24: VM generation: first run

- In line 1 the data-tree is searched for instances of the VM001 TO.
- In lines 4 to 9 the resulting list is checked for content and the DMS-Names of the VM001 instances are retrieved from it.
- Lines 10 to 16 handle the case of the list received as return value in line 1 being empty.

VM generation: second run

```
////////////////////////////////////////////////////////////////////////////////
// Second run: Get DB data and elements of the VM's
// Header
1 qsSettings->beginGroup("VM001HEADER");
2 qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString() % "\n\n");
3 qsSettings->endGroup();
4
5
6 if(!qlResult.isEmpty()) {
7       // Clear List
8       qlHashes2.clear();
9
10      // Get data
11      for(int i = 0; i < qlResult.size(); ++i) {
12              // Create VM-Hash
```

```
13            qlHashes2 << generatorLib->createVmHash(qlResult.at(i)->parent()-
>parent(), qslDmsNames.at(i));
14      }
15 }
16 else {
17      // Debug output
18      qDebug() << "Second run: Getting VM data failed";
19
20      // Return value
21      return;
22}
////////////////////////////////////////////////////////////////////////////////
```

Listing 4.9-25: VM generation: second run

- In lines 1 to 3 the header string is retrieved from the configuration file and moved to the output string.
- If the list contains data (line 6) a list of hashes will be constructed in lines 11 to 14 for each VM001 instance.
- Lines 16 to 21 notify the user that no VM001 instances have been found.

The *createVmHash()* function

```
1 QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
sGenLib::createVmHash(QStandardItem *qsiResult, QString qsInDmsName) {
2       // Locals
3       QString qsParent, qsScratch1, qsScratch2, qsScratch3;
4       QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
qlReturnHash;
5       QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*> *qmhLtoHash;
6       QMultiHash<QString, QHash<QString, QString>*> *qmhDatapointHash;
7       QHash<QString, QString>     *qhDataHash;
8       QStandardItem *qsiChild1, *qsiChild2, *qsiChild3, *qsiChildAddress;
9       int iRowCount = 0, iRowCount2= 0, iRowCount3= 0;
10
11      // Create a Data-Hash
12      qmhLtoHash = new QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>();
13      qlReturnHash << qmhLtoHash;
14
15      // At first get Parent of the item at i
16      qsParent = qsiResult->text();
17
18      // Check there are children
19      if(qsiResult->hasChildren()) {
20            // Check how many children the item has
21            iRowCount = qsiResult->rowCount();
22            // Check every child to be DB, F or R
23            for(int x = 0; x < iRowCount; ++x) {
24                  // Create Multi-Hashe
25                  qmhDatapointHash = new QMultiHash<QString, QHash<QString,
QString>*>();
26
27                  // Get child on row y, column 0 (allways 0 in tree's)
```

```
28                  qsiChild1 = qsiResult->child(x, 0);
29                  qsScratch1 = qsiChild1->text();
30                  // Check if child has children itself
31                  if(qsiChild1->hasChildren()) {
32                        // Check how many children the item has
33                        iRowCount2 = qsiChild1->rowCount();
34                        // Get grand-children
35                        for(int y = 0; y < iRowCount2; ++y) {
36                        // Get child on row z, column 0 (allways 0 in tree's)
37                              qsiChild2 = qsiChild1->child(y, 0);
38                              qsScratch2 = qsiChild2->text();
39
40                              // Check if child has children itself
41                              if(qsiChild2->hasChildren()) {
42                                    // Check how many children the item has
43                                    iRowCount3 = qsiChild2->rowCount();
44                                    // Get grand-children
45                                    for(int z = 0; z < iRowCount3; ++z) {
46                              // Get child on row z, column 0 (allways 0 in tree's)
47                                          qsiChild3 = qsiChild2->child(z, 0);
48                                          qsScratch3 = qsiChild3->text();
49
50                                          // Create a Data-Hash
51                                          qhDataHash = new QHash<QString,
QString>();
52
53                                          // Insert data into hash
54                                          qhDataHash->insert(qsScratch2,
qsScratch3);
55                                    }
56                              }
57
58                              // Insert Data-Hash into Datapoint-Hash
59                              qmhDatapointHash->insert(qsScratch1, qhDataHash);
60                        }
61                  }
62
63                  // Insert Data-Hash into Datapoint-Hash
64                  qmhLtoHash->insert(qsInDmsName, qmhDatapointHash);
65            }
66            // Return value
67            return qlReturnHash;
68      }
69      else {
70            // MessageBox
71            QMessageBox::warning(0, "sGenLib", "Error creating Hash");
72
73            // Clear Hash
74            qlReturnHash.clear();
75
76            // Return value
77            return qlReturnHash;
78      }
79 }
```

Listing 4.9-26: The createVmHash() function

The *createVmHash()* function is similar to the *createLtoHash()* function in Listing 4.9-7: The createLtoHash() function. The return value is a list holding a complex

construct that consists of multi hashes and hashes. The data-tree element of each instance of the VM001 TO is mapped into the VmHash. Therefore the data can be accessed very fast and in a most convenient way.

VM generation: third run

The code of the third run below has been split into various small parts to make the explanations easier.

First the used amount of helper registers is determined and the allocation string is assembled.

```
////////////////////////////////////////////////////////////////////////////////
1 // Third run: process data
2 // Generate helper ressources
3 qsSettings->beginGroup("VM001");
4 // Get keys
5 qslKeys = qsSettings->allKeys();
6 // Get number of keys
7 iKeyCount = qslKeys.size();
8 //
9 for(int i = 1; i <= iKeyCount; ++i) {
10     qvScratch1 = i;
11
12     // Count number of used helper registers
13     if(qsSettings->value(qvScratch1.toString(), "No entry").toString() ==
"Register") {
14             qvScratch3 = qvScratch3.toInt() + 1;
15     }
16 }
17
18 qsSettings->endGroup();
19
20 qvScratch1 = qlHashes2.size();
21
22 // Fill in Helper register
23 qsSettings->beginGroup("VM001GEN");
24 qsGeneratorTemplate.append(generatorLib->writeHelperRegister(qvScratch3.toInt(),
qsSettings->value("4", "No entry").toString()));
25 qsSettings->endGroup();
```

Listing 4.9-27: VM generation: third run step 1

- In lines 1 to 16 the number of used helper registers is determined and stored in the *qvScratch3* variable.
- In line 20 the number of VM001 instances is figured out
- In lines 22 to 25 the string to allocate the correct amount of helper registers is assembled and moved to the generator template string.

The generated code looks like this:

```
; ******************** VM's ************************
VM:
; ********** Resources

rHrVM001        EQU    R[5]                    ; Helper Registers
```

Listing 4.9-28: Generated VM header and helper registers

In the second step the address of the VM001 configuration DB is retrieved from the hash.

```
// Get DB-Address
1 if(!qlHashes2.isEmpty()) {
2      for(int a = 0; a < qlHashes2.size(); ++a) {
3             //
4             qmhLtoHash = qlHashes2.at(a);
5             qvScratch2 = qlHashes2.size();
6
7             // get data
8             QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>::const_iterator i;
9             for(i = qmhLtoHash->constBegin(); i != qmhLtoHash->constEnd(); ++i) {
10
11                   // Find DB-Address
12                   qsSettings->beginGroup("VM001");
13
14                   QString qsTest = qsSettings->value("2", "No entry").toString();
15
16                   QMultiHash<QString, QHash<QString, QString>*>::const_iterator x =
i.value()->find(qsSettings->value("2", "No entry").toString());
17                   while (x != i.value()->end() && x.key() == qsSettings->value("2",
"No entry").toString()) {
18                         // Get Key
19                         qsScratch2 = x.key();
20
21                         // get data
22                         QHash<QString, QString>::const_iterator y = x.value()-
>find("Address");
23                         while (y != x.value()->end() && y.key() == "Address") {
24                               // Get DB Address
25                               qsAddress = y.value();
26
27                               // Output data
28                               qvScratch1 = qlResult.size();
29
30                               // Insert Data into List
31                               qsGeneratorTemplate.append("CFG_VM0" %
qvScratch2.toString() % "\t" % "EQU" % "\t" % "DB " % qsAddress % "\t" % "; Datablock
for: " % qsScratch1 % "\n" % "DB CFG_VM0" % qvScratch2.toString());
32
33                               // Move DB Symbol-Name into StringList
```

```
34                                  qslVmDbAdresses << "CFG_VM0" %
qvScratch2.toString();
35
36                                  // increment iterator
37                                  ++y;
38                          }
39                          // increment iterator
40                          ++x;
41                  }
42                  qsSettings->endGroup();
```

Listing 4.9-29: VM generation: third run step 2

- In line 1 the list is checked for content.
- In line 2 the *for-Loop* going over the list containing the VM001 instances is started.
- From line 8 to 30 the DB address for each VM001 instance is retrieved from the hash.
- In line 31 the allocation string for the VM001 DB is assembled and appended to the VM generator template.
- In line 32 the DB address is stored in the string list for further use.

In the third step the number of DB elements is determined.

```
// Get number of DB items
1 QMultiHash<QString, QHash<QString, QString>*>::const_iterator d;
2 for(d = i.value()->constBegin(); d != i.value()->constEnd(); ++d) {
3       // Get Key
4       qsScratch2 = d.key();
5
6       // Set Variables
7       qvScratch1 = 0;
8
9       // get data
10      QHash<QString, QString>::const_iterator e = d.value()->find("DBIndex");
11      while (e != d.value()->end() && e.key() == "DBIndex") {
12              if(e.value().toInt() > iDbIndex) {
13                      iDbIndex = e.value().toInt();
14                      qvScratch1 = iDbIndex + 1;
15
16                      // Output data
17                      qsDbItems1 = " [" % qvScratch1.toString() % "] ";
18              }
19              // increment iterator
20              ++e;
21      }
22 }
```

Listing 4.9-30: VM generation: third run step 3

The hash is searched for the *DBIndex* keyword to determine the number of DB elements.

- In lines 11 to 21 the highest value for the found *DBIndex entries* is determined by a kind of bubble search. The string is assembled with the necessary parentheses and stored in *qsDbItems1* variable.

In the fourth step the pointers to the LTO01 TOs are retrieved from the VM hash and inserted into the allocation string.

```
1 // Insert Pointers to LTO addresses
2 qsSettings->beginGroup("VM001GEN");
3
4 qsScratch = qsSettings->value("3", "No entry").toString();
5 if(qsScratch2.contains(qsScratch)) {
6      qsScratch = qsScratch2;
7 }
8
9 QMultiHash<QString, QHash<QString, QString>*>::const_iterator ffff = i.value()-
>find(qsScratch);
10 while (ffff != i.value()->end() && ffff.key() == qsScratch) {
11     // Get Key
12     qsScratch3 = ffff.key();
13
14     // get data
15     QHash<QString, QString>::const_iterator gggg = ffff.value()->find("PAR_IN");
16     while (gggg != ffff.value()->end() && gggg.key() == "PAR_IN") {
17           // Get DB Data
18           qsScratch4 = gggg.value();
19
20           // Check string
21           if(qsScratch4.contains("Pointer to the LTO")) {
22                 // insert 0
23                 qsDbItems3.prepend(", 0");
24           }
25           else {
26                 // Check string
27                 if(qsScratch4.section('.', 0, 0) == "F") {
28                       // Insert flag-address
29                       qsDbItems3.prepend(", " % qsScratch4.section('.', 1, 1));
30                 }
31
32                 else {
33                       // Get last bit of the DMS-Name
34                       qsDmsNameLastPart = qsScratch4.section(':', -1, -1);
35
```

Listing 4.9-31: VM generation: third run step 4

- In lines 5 to 7 the variable *qsScratch2* is checked for its content and the variable *qsScratch* is set accordingly.
- In line 9 the outer iterator is created.

- In line 10 the *while-Loop* that goes over the entries of the VM001 instance hash is started.
- In line 15 the retrieval of the values tagged by the search string *PAR_IN* is started.
- In line 18 a value is retrieved from the hash and stored in the variable *qsScratch4*.
- Starting from line 20 the retrieved value is checked for the various possible contents.
- Lines 21 to 24 checking the value to be unused which means a placeholder (*Pointer to the LTO*) has been retrieved.
- Starting from line 21 the content is tested for being a flag address (line 26 to 30) where the flag address is retrieved directly (line 29).
- In line 34 the content is assumed to be a DMS-Name (*WI065:H02:PW:001:Running*), so the last part of the DMS-Name (*Running*) is used to search for its pointer indirectly.

In the next step the data tree is searched for the entries corresponding to the last part of the DMS-Name found above.

```
1 // Search for address
2 qlResultAddress = qhTrees.value("Data-Tree")->findItems(qsDmsNameLastPart,
Qt::MatchRecursive | Qt::MatchContains);
3
4 if(!qlResultAddress.isEmpty()) {
5      for(int i = 0; i < qlResultAddress.size(); ++i) {
6            // Assemble DMS-Name
7            // At first get Parent of the item at i
8            qsiResultAddress = qlResultAddress.at(i);
9
10           // Check there is a parent
11           qsiResult = qsiResultAddress;
12           if(qsiResult != 0) {
13                 while(qsiResult->parent()) {
14                       qsDmsName.prepend(qsiResult->text() % ":");
15                       qsiResult = qsiResult->parent();
16                 }
17           }
18           // Move DMS-Name into string list
19           qsDmsName.remove(qsDmsName.lastIndexOf(":"), 1);
20
21           // Check DMS-Name
22           if(qsDmsName == qsScratch4) {
23                 if(qsiResultAddress->hasChildren()) {
24                       //
25                       QString qsTest2 = qsiResultAddress->text();
26                       iRowCount = qsiResultAddress->rowCount();
```

```
27
28                          // Get Child
29                          qsiChild1 = qsiResultAddress->child(0, 0);
30                          QString qsTest3 = qsiChild1->text();
31                          if(qsiChild1->text() == "Address") {
32                                 // Check if child has children itself
33                                 if(qsiChild1->hasChildren()) {
34                                        // Get Address, which is the child's text
35                                        qsiChildAddress = qsiChild1->child(0, 0);
36                                        qsAddress = qsiChildAddress->text();
37                                        qslVmInputAddresses << qsAddress;
38
39                                        // break loops
40                                        i = qlResultAddress.size();
41                                 }
42                          }
43                   }
44                   else {
45                          // insert 0
46                          qsDbItems3.prepend(", 0");
47                   }
48            }
49
50            // Clear string
51            qsDmsName.clear();
52     }
53 }
54
55 // insert Address
56 qsDbItems3.prepend(", " % qsAddress);
57
```

Listing 4.9-32: VM generation: third run step 4 (continued)

- In line 2 the data tree is searched for entries containing the search string retrieved in the previous step.
- In line 4 the result is checked for content.
- In line 5 the retrieval of the pointer is started.
- In lines 6 to 19 the DMS-Name of an entry is assembled.
- In lines 21 to 56 the pointer is retrieved and stored in the *qsDbItems3* variable and a comma is prepended. If no pointer is found a 0 is stored and a comma is prepended.

In the next step, the number of LTO01 TOs that are executed by the VM001 TO are retrieved, a 0 is prepended as a place holder for the cycle time, the reserve data points are filled with 0s as place holders and everything is added to the configuration DB of the V001 LTO.

```
// Add Number of used LTO's
```

```
1 qvScratch1 = qslVmInputAddresses.size();
2 qsDbItems2.prepend(qvScratch1.toString());
3
4 // Prepend Cycletime
5 qsDbItems2.prepend("0, ");
6
7 // Insert Reserve Configuration Datapoints
8 for(int c = 0; c < 8; ++c) {
9      qsDbItems2.append(", 0");
10 }
11
12 // Insert NewLine
13 qsDbItems3.append("\n\n\n");
14
15 // Append DB-Data to the StringList
16 qsGeneratorTemplate.append(qsDbItems1);
17 qsGeneratorTemplate.append(qsDbItems2);
18 qsGeneratorTemplate.append(qsDbItems3);
19
20 // Clear Strings
21 qsDbItems1.clear();
22 qsDbItems2.clear();
23 qsDbItems3.clear();
24
25 // Set Variables
26 iDbIndex = 0;
```

Listing 4.9-33: VM generation:

- In lines 1 and 2 the number of executed LTO01 TOs is determined and added to a temporary Qstring variable.
- In line 5 the place holder for the cycle time is prepended.
- In lines 8 to 10 the reserve data points are inserted.
- In line 13 three newlines are added to a temporary QString variable.
- In lines 16 to 18 the temporary strings are appended to the generator template string.
- In lines 20 to 26 the temporary Qstring variables are cleared and the *iDBIndex* variable is set to 0.

In the next step the function block call for the VM001 TO is inserted into the generator template string.

```
1 // Insert Data
2 for(int I = 1; I <= iKeyCount; ++i) {
3      //
4      qsSettings->beginGroup("VM001");
5      qvScratch1 = I;
6      qsScratch1 = qsSettings->value(qvScratch1.toString(), "No entry").toString();
7
```

```
8       qsSettings->endGroup();
9
10      // Assemble the data
11      switch(qvScratch1.toInt()) {
12              // Header
13              case 1:
14                      qsGeneratorTemplate.append(qsScratch1 % "\n");
15              break;
16
17              // DB Address
18              case 2:
19              // DB Address (for Pointer)
20              case 3:
21                      qsScratch1 = qsScratch1 %"\t; [" % qvScratch1.toString() % "]" %
qsScratch1  % "\n";
22                      qsGeneratorTemplate.append("DB " % qslVmDbAdresses.at(a) % "; [" %
qvScratch1.toString() % "]" % "DB Address" % "\n");
23              break;
24
25              // Configuration Pointer
26              case 4:
27                      qsScratch1 = qsScratch1 % "\t; [" % qvScratch1.toString() % "]" %
qsScratch1  % "\n";
28                      qsScratch3 = generatorLib->makeSymbol(qslDmsNames.at(a),
qsScratch1);
29                      qsGeneratorTemplate.append(qsScratch3);
30              break;
31
32              // Helper Register
33              case 5:
34              // Helper Register
35              case 6:
36              // Helper Register
37              case 7:
38              // Helper Register
39              case 8:
40              // Helper Register
41              case 9:
42                      qsSettings->beginGroup("VM001GEN");
43                      qvScratch3 = qvScratch1.toInt() - qvScratch2.toInt();
44                      qsGeneratorTemplate.append(qsSettings->value("4", "No
entry").toString() % "+" % qvScratch3.toString() % "\t; [" % qvScratch1.toString() %
"]" % qsScratch1  % "\n");
45                      qsSettings->endGroup();
46              break;
47
48              // Error
49              default:
50                      qsGeneratorTemplate.append("Something is wrong\n");
51      }
52 }
```

Figure 4.9-10: Insert function block call for the VM001 TO

- In line 2 the *for-Loop* that is going over all the keys found in the configuration file is started.
- In line 4 the *qsSettings* object is set to the *VM001* group in the configuration file.

- In line 5 the QVariant variable *qvScratch1* is set to the value of the control variable.
- In line 6 the *qvScratch1* variable is set to the value of the configuration key addressed by the control variable of the *for-Loop*.
- In line 8 the *qsSettings* object is set back to default.
- In lines 11 to 51 the function block parameters are generated and inserted into the generator template string. Again the side effects of the *switch* instruction are used advantageously.

To conclude the assembly of the generator template string some new line characters are inserted for formatting purposes and then the footers are inserted.

```
1 // Add New-lines
2 qsGeneratorTemplate.append("\n\n");
3
4 // VM Footer
5 qsSettings->beginGroup("VM001FOOTER");
6 qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString() %
"\n\n\n");
7 qsSettings->endGroup();
8
9 // COB Footer
10 qsSettings->beginGroup("COBFOOTER");
11 qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString() %
"\n\n\n");
12 qsSettings->endGroup();
```

Figure 4.9-11: Insert new lines, VM footer and COB footer

- In line 2 two new line characters are inserted into the generator template.
- In lines 4 to 7 the VM footer is inserted as a closing statement for the VM001 FB call.
- In lines 9 to 12 the COB-Footer is inserted as a closing statement for the whole generated PLC programm.

In the last step the assembled generator template is written to the output file and the *vmSymbolsWritten()* signal is emitted to indicate the successful completion of the *qsmsWritteVmSymbols* state.

```
////////////////////////////////////////////////////////////////////////////////
1 // Output Data to the file
2 // Set stream to the end of the file
```

```
3 qtsOutput.seek(qfOutput.size());
4
5 // Append data to the StringList
6 qslOutputData << qsGeneratorTemplate;
7
8 // Output Data
9 for(int i = 0; i < qslOutputData.size(); ++i) {
10     qtsOutput << qslOutputData.at(i);
11 }
12 ////////////////////////////////////////////////////////////////////////////
13
14 // emit signal
15 emit vmSymbolsWritten();
16
17     // Return value
18     return;
```

Figure 4.9-12: Writting the generator template string to the file

- In line 3 the file pointer is set to the end of the file.
- In line 6 the generator template string is appended to the *qslOutputData* QStringList.
- In line 9 to 11 the content of the *qslOutputData* QStringList is written to the output file by means of the *qtsOutput* QTextStream.
- In line 15 the *vmSymbolsWritten()* signal is emitted.
- In line 18 the function returns to the call function.

The generated code looks like this:

```
; ******************** VM's ************************
VM:
; ********** Resources

rHrVM001        EQU     R[5]                    ; Helper Registers

CFG_VM01        EQU     DB 4011                 ; Datablock for: Register
DB CFG_VM01 [20] 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2006, 2208, 2209, 2210, 0, 0, 0, 0, 0, 0


; ********** Call FB

        CFB     VM001                   ; Virtual Machine to be used with the LTO01
                DB CFG_VM01             ; [2]DB Address
                DB CFG_VM01             ; [3]DB Address
```

```
                H20.LG.CTRL.VM_001.CFG_Pointer    ; [4]CFG_Pointer
                rHrVM001+0                        ; [5]Register
                rHrVM001+1                        ; [6]Register
                rHrVM001+2                        ; [7]Register
                rHrVM001+3                        ; [8]Register
                rHrVM001+4                        ; [9]Register

END_VM:
; ***************** Ende VM's **********************


END_COB:

        ECOB


; ***************** Ende COB **********************
$ENDGROUP
```

Figure 4.9-13: The generated code for the VM

The generated PLC software can be seen in Appendix I Generated IL-Code.

4.9.5.4 qsmsCloseVmFile

This state is used to close the output file.

```
1 void sGen::if_qsmsCloseVmFile_entered() {
2      // Debug message
3      qDebug() << "Entering step: if_qsmsCloseVmFile_entered";
4
5      // Debug output
6      qDebug() << "closing vm-file...";
7
8      // Close file
9      qfOutput.close();
10
11     // emit signal
12     emit vmFileClosed();
13
14     // Return value
15     return;
16 }
```

Figure 4.9-14: The qsmsCloseVmFile_entered() function

- In line 3 and 6 debug messages are written into the debug window.
- In line 9 the output file is closed.
- In line 12 the *vmFileClosed()* signal is emitted

4.9.5.5 qsmfsEndVmGeneratorMachine

This is the final state of the generator machine. Reaching this state the machine will be stopped. After this step the PLC software is generated and can be compiled by the PLC-System programming tool.

```
1 void sGen::if_qsmfsEndVmGeneratorMachine_entered() {
2      // Debug message
3      qDebug() << "Entering step: if_qsmfsEndVmGeneratorMachine_entered";
4
5      // emit signal
6      emit vmGenerated();
7
8      // Return value
9      return;
10 }
```

Figure 4.9-15: Final state of the generator machine is reached

- In line 3 a debug message is written into the debug window.
- In line 5 the *vmGenerated()* signal is emitted.
- In line 9 the function returns to the call function.

Chapter 5: Testing

5.1 Testing environment

As testing environment a part of a real project has been used. The test project comprises two heating units. A Saia-Burgess PCD3.M5540 CPU is used as PLC-System. ProMoS NT is used as SCADA-System.

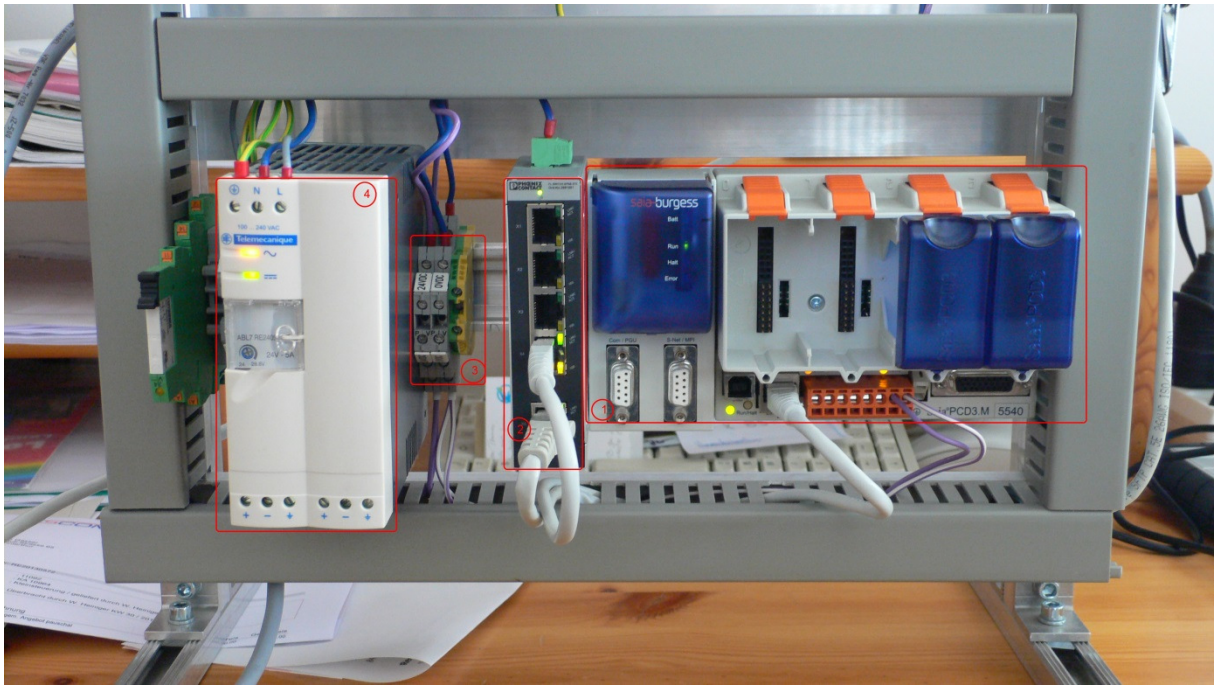

Figure 5.1-1: The used test system

Number.	Description	Typ
1	PLC-System	Saia-Burgess PCD3.M5540
2	Ethernet Switch	Phoenix Contact FL-SWITCH SFNB 5TX
3	Clamps for the voltage distribution	
4	Power Supply	Telemecanique ABL7 RE2405

There are no field instruments connected to the PLC-System. The necessary data is written into the PLC-System via the tools provided by the programming software.

5.1.1 Details PLC-System

Description	Version
PLC Type	Saia PCD3.M5540
Firmware	1.16.69
Hardware	D2 (02h)
Modifications	2 (02h)
Fabrication Date	2007 / 15
Serial Number	00BC3824
MAC-Address	00 50 C2 6F DB A7
PG5	2.0.220.200

Table 5.1-1: Details PLC-System

For the complete list of versions please refer to Appendix K Versions PLC-System.

5.1.2 Details ProMoS NT

Program name	Version
DMS	1.6.8.102
PDBS	1.6.8.24
PET	1.6.8.79
SDriver	1.6.9.39
GE	1.6.8.127

Table 5.1-2: Details ProMoS NT

For the complete list of versions please refer to Appendix L Versions ProMoS NT.

5.2 Testing tools

As main testing tool the Saia-Burgess PLC programming software PG5 was used. PG5 consists of:

- IL editor
- Online debugger
- Watch window
- In code debugger

5.2.1 The online debugger

The online debugger is the oldest tool available. It was introduced with one of the earliest versions of the Saia-Burgess PG programming tool.

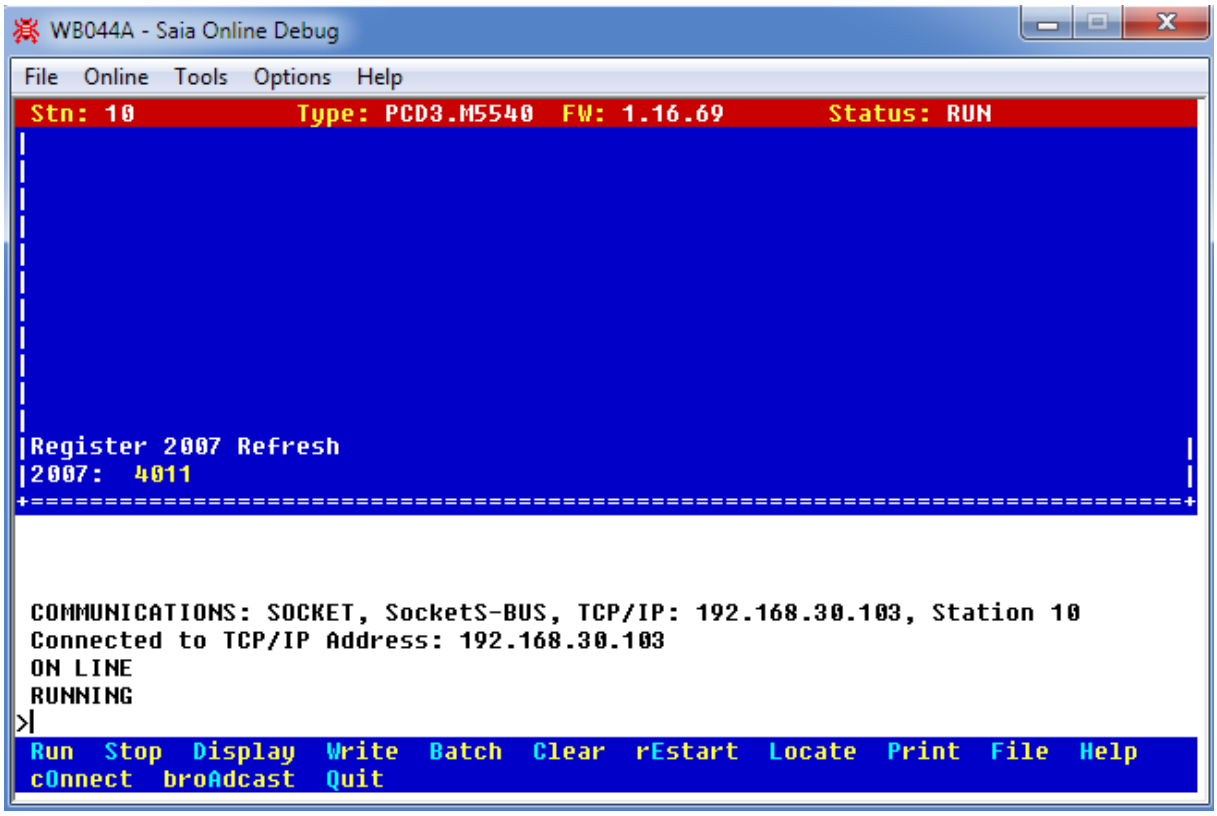

Figure 5.2-1: Saia online debugger

The blue highlighted letters are the shortcuts to carry out the respective commands. The Online Debugger is a very helpful tool if one wants to carry out commands in a batch.

5.2.2 The watch window

The watch window was introduced in the first version of PG5. It is an enhancement of the old online debugger. Since version 2.0.0.0 of the PG5 programming software it is possible to chart the monitored signals.

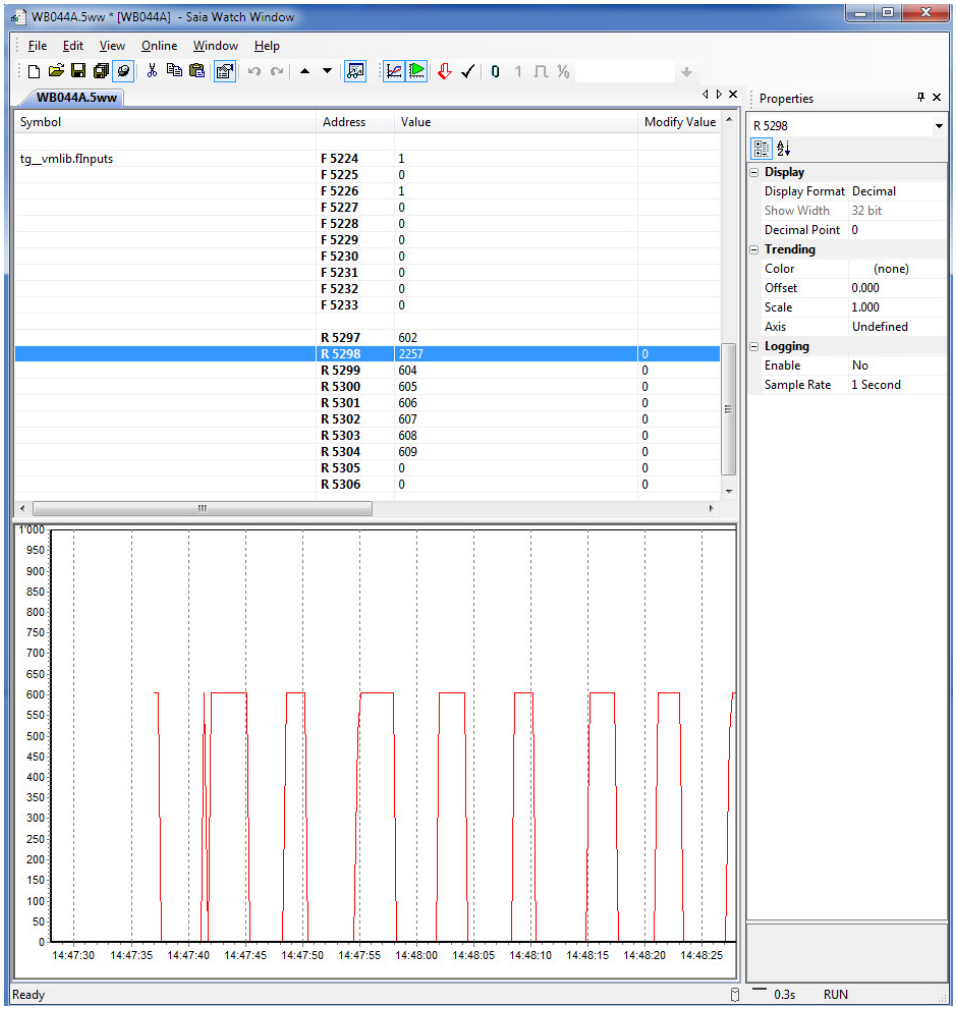

Figure 5.2-2: Saia watch window

5.2.3 In code debugger

The incode debugger was also introduced in the first version of PG5. It allows debugging the PLC-Software directly in the IL code as it is common from the high level programming languages.

```
; ********** Call FB

1       CFB     LTO01                                   ; Call LTO01
2               DB CFG_LTO01                            ; [2]DB Address
3               DB CFG_LTO01                            ; [3]DB Address
4               H09.LG.CTRL_002.CFG_Pointer             ; [4]CFG_Pointer
5               H09.LG.CTRL_002.OUT_Output              ; [5]OUT_Output
6               H09.LG.CTRL_002.OUT_OutputInvers        ; [6]OUT_OutputInvers
7               H09.LG.CTRL_002.OUT_Output              ; [7]OUT_Output
8               H09.LG.CTRL_002.OUT_OutputInvers        ; [8]OUT_OutputInvers
9               rHrLTO01+1                              ; [9]Register

10      010537          CFB     14
11      010538                  DB 4008
12      010539                  4008
13      010540                  R 2210          [4008]
14      010541                  F 2257          [0]
15      010542                  F 2258          [1]
16      010543                  2257
17      010544                  2258
18      010545                  R 5287          [2258]
```

Listing 5.2-1: In code debugging

- In lines 1 to 9 the generated source code is shown.
- In lines 10 to 18 the code downloaded and run in the PLC-System is shown.
- In lines 13, 14, 15 and 18 the actual value stored in the respective register of flag is shown in brackets.

5.2.4 Debugging tools

As in the IDEs of the high level languages there are some debugging tools one can use to check the program under test.

Figure 5.2-3: Debugging tools

1. Asynchronous Data View: In this mode the actual operand data values are refreshed on the screen, but they are not synchronized with the execution of the program.
2. Synchronous Data View: In this mode the values of operand data are only displayed if the code is actually executed.
3. Run
4. Stop
5. Step into
6. Step over
7. Step out
8. Run to cursor
9. Graftec step; run to next ST or TR
10. Set / clear breakpoint

5.2.5 Use of the new system in a real facility

From a real facility two heating units have been extracted and used as test system.

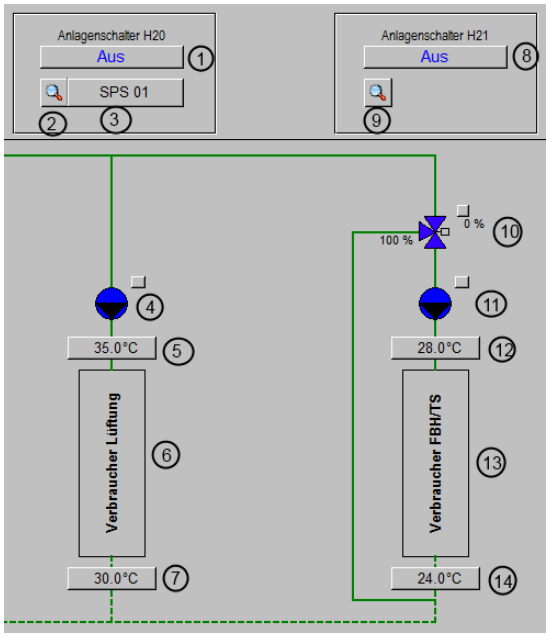

Figure 5.2-4: Plant diagram (Heating system) of the test plant

Number	Description
1	Main software switch of the heating system H20
2	Button to show the detail plant diagram (Figure 5.2-6: System overview in a real plant diagram)
3	LTO to show the state of the PLC-System
4	Circulation pump of the heating system H20
5	Inlet temperature sensor
6	Heat consumer
7	Return temperature sensor
8	Main software switch for the heating system H21
9	Button to show the detail plant diagram (Figure 5.2-6: System overview in a real plant diagram)
10	Control valve
11	Circulation pump of the heating system H21
12	Inlet temperature sensor
13	Heat consumer
14	Return temperature sensor

Table 5.2-1: Legend of the Plant diagram (Heating system) of the test plant

By clicking the button with the magnifying glass on it, one can open the plant diagram with the programming on it (see Figure 5.2-5: Plant diagram before the use of the VM). This plant diagram is used by the plant operator to control the plant.

5.2.6 Plant diagram before the use of the VM

For the real plant the programming has been done on the plant diagram by the use of the previous, existing LTO version. These simple logical conjunctions should be replaced by the newly developed LTO01 and VM001 TOs.

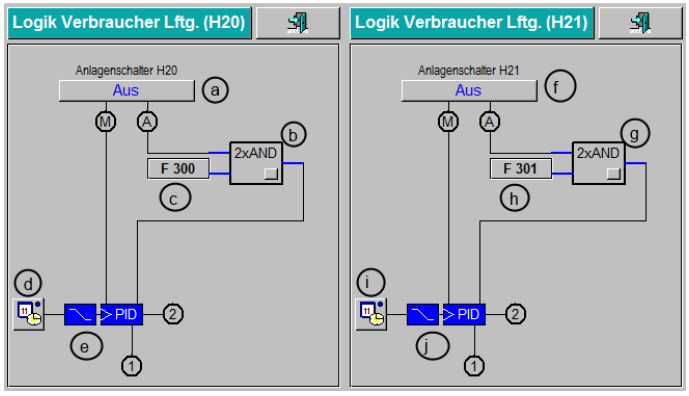

Figure 5.2-5: Plant diagram before the use of the VM

Number	Description
a	Main software switch of the heating system H20
b	LTO
c	Additional required input for the LTO
d	Time switch to change the heating curve
e	PID controller
f	Main software switch of the heating system H21
g	LTO
h	Additional required input for the LTO
i	Time switch to change the heating curve
j	PID controller

Table 5.2-2: Legend of the Plant diagram before the use of the VM

5.2.7 Plant diagram using the VM

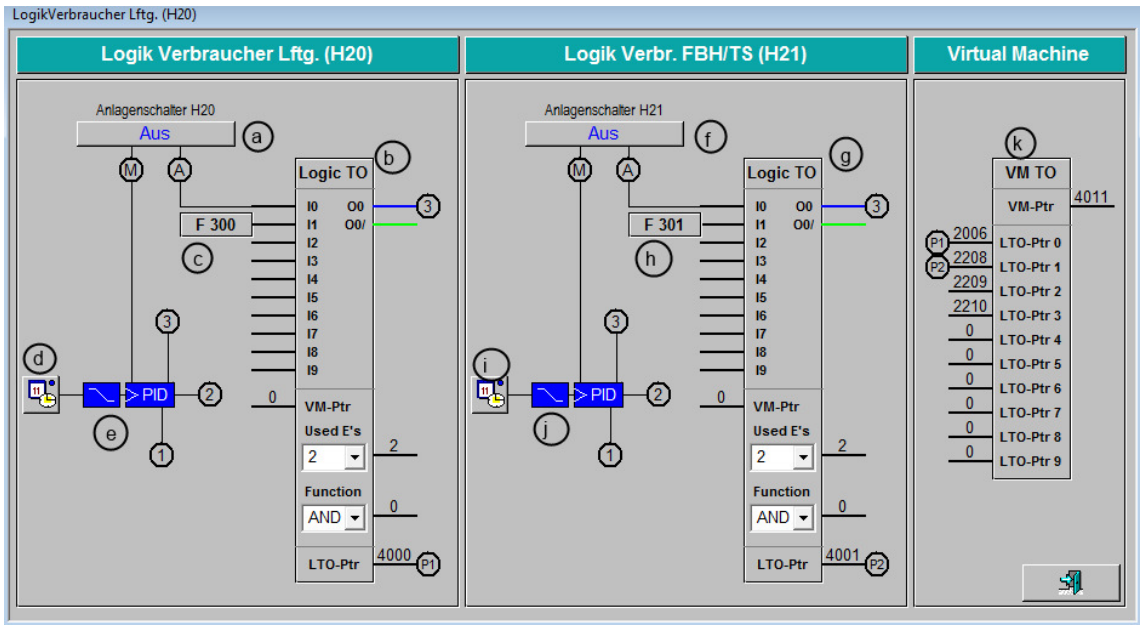

Figure 5.2-6: System overview in a real plant diagram

As can be seen in the figures above, the previous LTOs are replaced with the newly developed LTO (LTO01) and the two instances of the LTO01 TO are linked to the virtual machine (VM001) instance where they are executed.

Number	Description
a	Main software switch of the heating system H20
b	New LTO
c	Additional required input for the LTO
d	Time switch to change the heating curve
e	PID controller
f	Main software switch of the heating system H21
g	New LTO
h	Additional required input for the LTO
i	Time switch to change the heating curve
j	PID controller
k	Virtual machine
P1 / **P2**	Pointers to the LTO data

Table 5.2-3: Legend of the System overview in a real plant diagram

5.3 Testing procedure

To test the developed software and the whole VM / LTO System the testing has been split into two phases:

1. Compiling Test: Is it possible to compile the generated source code without errors or warnings.
2. Functional test:
    a. Test of the plant function
    b. Test of the LTO functions
    c. Test of the VM functions

5.4 Compiling test

The PG5 programming tool is used to compile the PLC-Software and check whether there are errors and warnings or not (see Listing 5.4-1: Compiling messages). In case of errors or warnings, they have to be resolved and the code generator has to be adjusted to generate the code correctly.

```
======================================================
Rebuild All Started: Friday, September 06, 2013 at 08:56:33
Project: pcd
Device: WB044A

Saia PG5 Program Builder V2.0.220.0
Licensed to: Scheco AG
Build command file: WB044A.mak

Assembling: _WB044A.src
Assembling: petcode.src
Generated: Freitag, 30. August 2013 10:51:24
License  : Developement - -
Assembling: tg_main.src
Assembling: tGVm.src
Assembling: ViSiPlus.sy5
Assembling: _TCPIPDBX.src
Assembling: C:\Users\Public\Saia-Burgess\PG5_20\Libs\Std\InitODM.src
Assembly complete.  Errors: 0  Warnings: 0

Linking: _WB044A.obj + petcode.obj + tg_main.obj + tGVm.obj + ViSiPlus.obj + _TCPIPDBX.obj + InitODM.obj
To: WB044A.pcd WB044A.map
Code size: 11068 lines (44272 bytes)
Text/DB size: 416 bytes
Extension memory size: 20988 bytes
Information Block segment size: 1000 bytes
Public symbols: 810
Linkage complete. 0 errors, 0 warnings.

Generating Block Information files...
Block Information Files complete

Build successful.  Total errors: 0  Total warnings: 0
======================================================
```

Listing 5.4-1: Compiling messages

5.5 Functional test

This test has been done by the use of the SCADA-System and the programming software PG5.

5.5.1 The plant overview diagram

To have a better overview a plant overview diagram has been created.

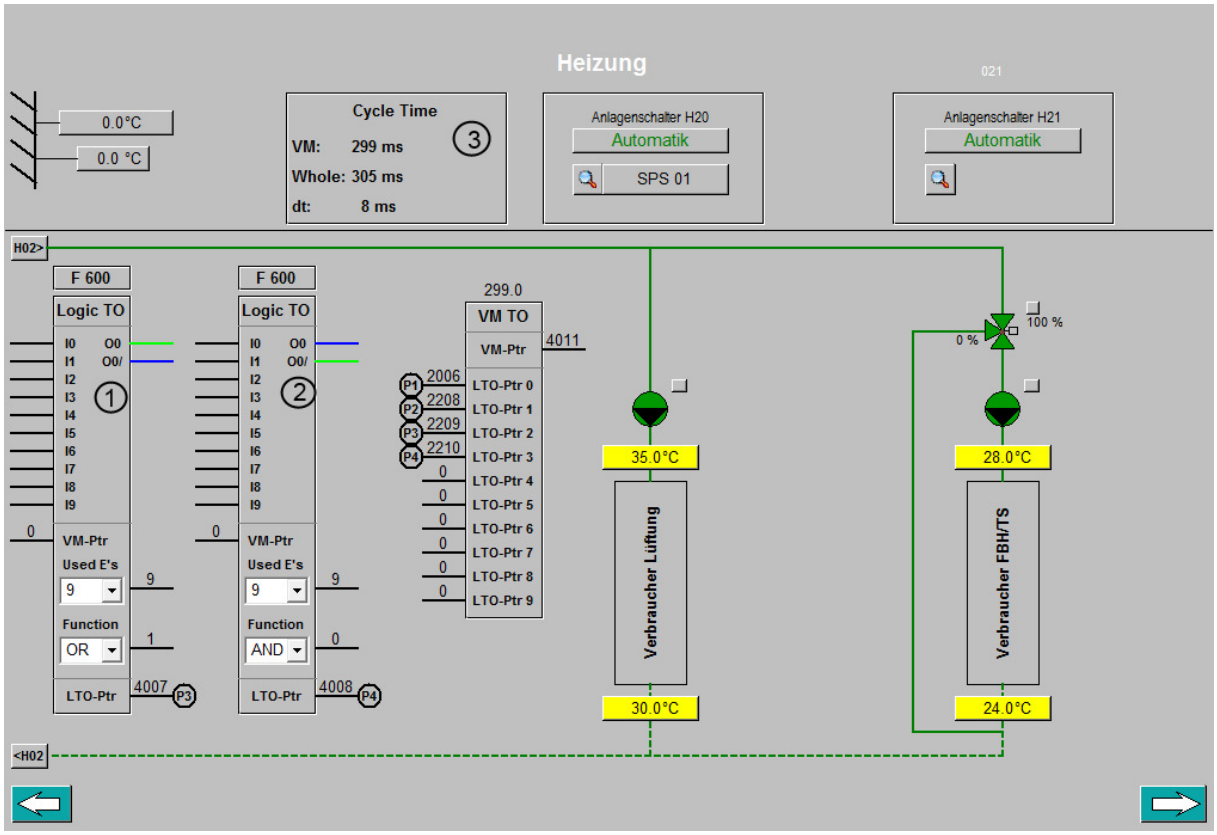

Figure 5.5-1: The plant overview diagram

Number	Description
1	Additional LTO001 instances to test the effect on the virtual machine and the cycle time
2	Additional LTO001 instances to test the effect on the virtual machine and the cycle time
3	Cycle time

Table 5.5-1: Legend of the The plant overview diagram

To test the functionality of the plant an additional plant diagram has been designed.

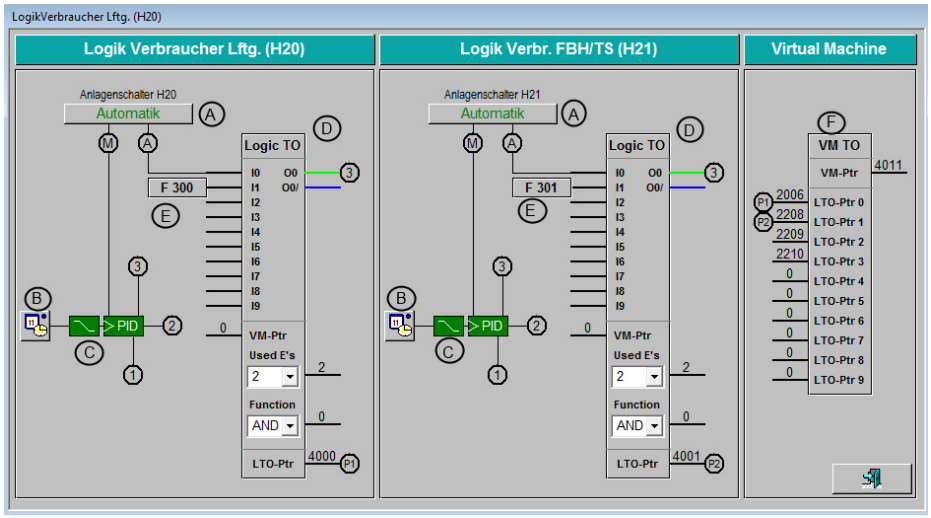

Figure 5.5-2: Testing plant function 01

Number	Description
A	Main software switch of the heating system H20
B	Time switch to change the heating curve
C	PID controller
D	New LTO
E	Additional required input for the LTO
F	Virtual machine
P1 / P2	Pointers to the LTO data

Table 5.5-2: Legend of the Testing plant function 01

Both LTOs are set up to use two inputs, one is the main software switch and the other is the additional input and AND-Gate. The output (3) is set to high state and thus the PID controller is enabled (see Figure 5.5-2: Testing plant function). This implies that both of the used inputs of the LTO001 TO are set to high state. For control the watch window is used.

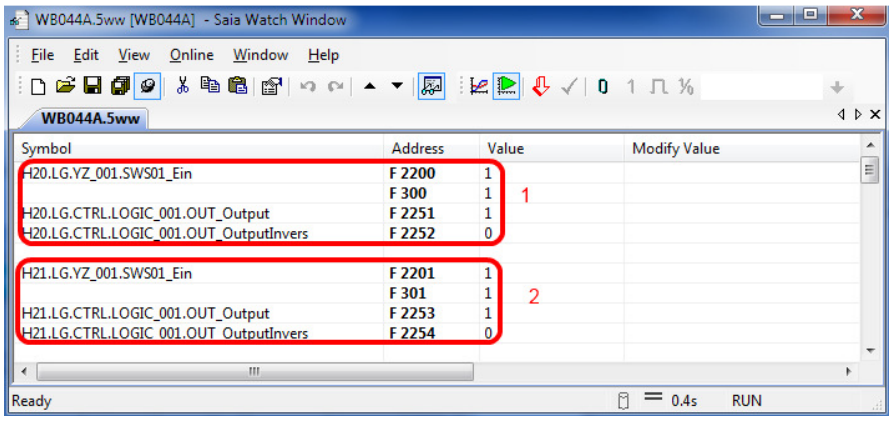

Figure 5.5-3: Watch window 01

The signals labelled 1 are the signals corresponding to the LTO001 instance for the plant part labelled H20 and the signals labelled 2 belonging to the LTO001 instance of the plant part labelled H21.

The order of the signals starting with the top most signal is

- Input 1 of the LTO001 instance
- Input 2 of the LTO001 instance
- Output of the LTO001 instance
- Inverse Output of the LTO001 instance

As both inputs are in high state the output is set to high state too.

If the states of the inputs are changed the state of the output is adapted accordingly

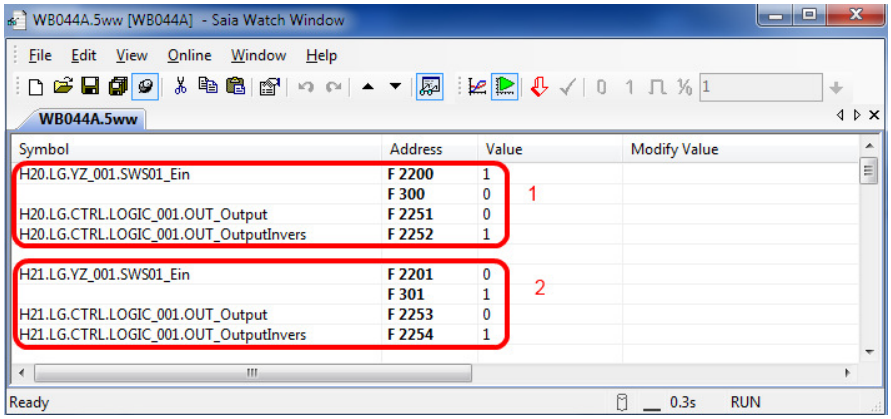

Figure 5.5-4: Watch window 02

This will result in the disabling of the PID controllers

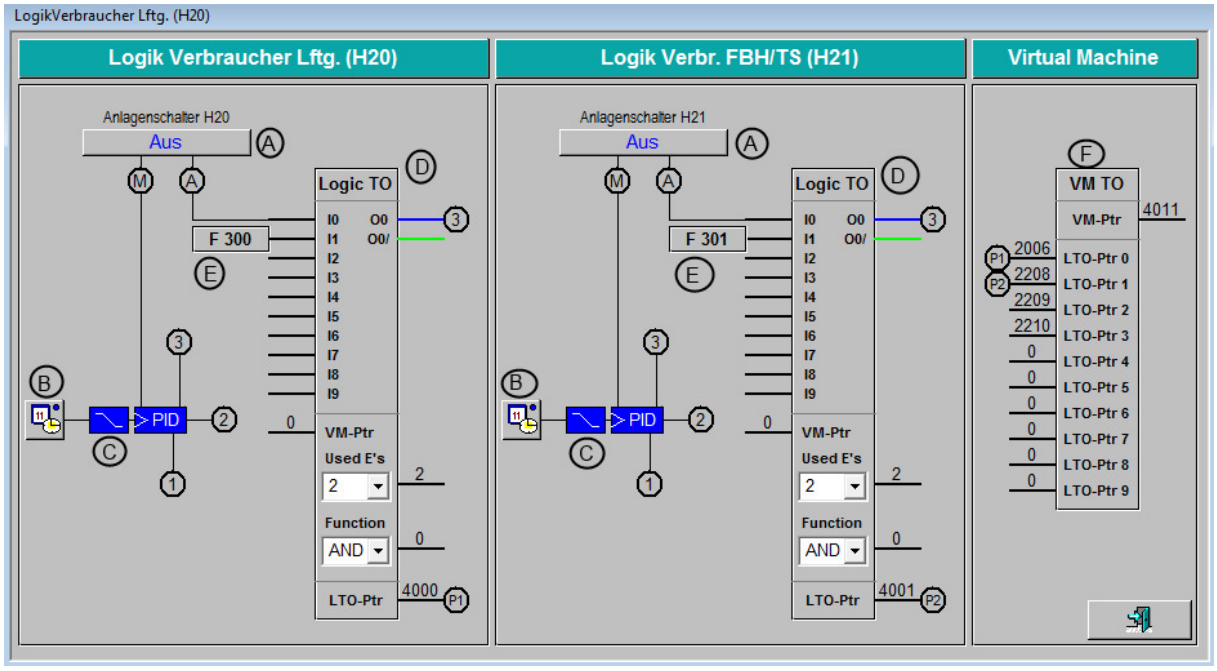

Figure 5.5-5: Testing plant function 02

5.6 Testing results

5.6.1 LTO001 AND-Gate

Device under Test	Passed	Not Passed	Comments
AND-Gate 2 Inputs	X		
AND-Gate 3 Inputs	X		
AND-Gate 4 Inputs		X	
AND-Gate 5 Inputs		X	
AND-Gate 6 Inputs		X	
AND-Gate 7 Inputs		X	
AND-Gate 8 Inputs		X	
AND-Gate 9 Inputs		X	
Change input polarity		X	Polarity changing has no effect

Table 5.6-1: Test results LTO001

5.6.2 LTO001 OR-Gate

Device under Test	Passed	Not Passed	Comments
OR-Gate 2 Inputs	X		
OR-Gate 3 Inputs	X		
OR-Gate 4 Inputs		X	
OR-Gate 5 Inputs		X	
OR-Gate 6 Inputs		X	
OR-Gate 7 Inputs		X	
OR-Gate 8 Inputs		X	
OR-Gate 9 Inputs		X	
Change input polarity		X	Polarity changing has no effect

Table 5.6-2: Test results LTO001

The code for the basic digital technology functions has to be rechecked carefully to ensure the correct function. For the test plant the functionality can be assured with the current software. It is essential that all intended functions are working properly.

5.6.3 VM001

Device under Test	Passed	Not Passed	Comments
Change Nbr of LTO's	X		
Add LTO at runtime	X		
Swapping LTO's	X		

Table 5.6-3: Test results VM001

5.6.4 Cycle time

The cycle time is captured for the vm alone as well as for the whole software. The cycle time has been averaged over a period of 900 sec for seven days. The table below shows the VM adding a huge amount of time onto the cycle time of the whole system whereupon the jitter of the cycle time lies in an acceptable range for a system in building automation.

	VM [ms]	Whole [ms]	Difference [ms]
Maximum	309.000	317.000	8.000
Minimum	288.000	296.000	7.000
Jitter	21.000	21.000	1.000
Average	304.880	313.339	7.870
Median	305.000	313.500	8.000

Table 5.6-4: Cycle time

Chapter 6: Results and Discussion

With this future oriented approach, it is possible for the first time to accomplish all tasks in the realization of a project in building automation with one tool using the object orientated paradigm in a top down approach without any dependency on the PLC hardware.

The user of the SCADA-System FBD will be able to respond to the wishes of the customer regarding the PLC hardware without problems. Projects once built with the SCADA-System FBD can be used with hardware of different manufacturers. The user has to change only the PLC-Manufacturer in his project and to regenerate the code. In this way the user can easily build up libraries of plant templates and achieve a much faster time to market with his products. As the SCADA-System FBD is not developed by a PLC-Manufacturer ensures that no economic interests will reduce the independency from the PLC-Hardware

The approach invented and enhanced in this investigation is particularly useful in the retrofit business. It is possible to use the same plant software on a new generation of hardware. The upgrading of the plant can be done quite fast and there is no need to train the plant operator on anything other than the handling of the new hardware.

The results of this investigation show the usefulness of the SCADA-System FBD. The idea of a virtual machine on the PLC-System has to be looked at very carefully and critically. The VM adds a huge amount of time onto the cycle time and therefore is quite expensive. The merit of the VM is ambivalent. On one hand it offers some advantageous characteristics like the possibility to change the PLC-Software in runtime without the need of downloading any software at all. On the other hand it adds complexity to the system which necessitate a certain expertise in programming.

Chapter 7: Conclusions

This investigation has shown the realisability of the original research questions. The testing of the developed software proved that it is possible to build a virtual machine for PLC-Systems and run it. The investigation has also proved that it is possible to automatically generate the necessary code to run on the virtual machine.

The virtual machine needs a lot of time to run through its codes and adds a considerable amount of time to the cycle time where the LTO are less expensive in reference to cycle time. To save cycle time and have the PLC software running faster it is advisable to drop the idea of a virtual machine and implement the LTO as standalone functions.

Generally it can be said that the merging together of SCADA, HMI development and PLC programming leads to a more natural way in programming PLC-Systems. In a first step the user draws his plant diagram with template objects and afterwards binds the template objects on the plant diagram together. With this approach the user would have to cope with one software tool only. The independency from the PLC hardware is an additional alleviation for the user. So this investigation helps to intensify the cooperation between PLC manufacturers and SCADA-System developers.

7.1 Achievement of objectives

7.1.1 Objective 1: To develop PLC software using the object orientated paradigm

This objective was accomplished completely. All PLC software developed or generated in this investigation complies with the rules of the object orientated paradigm. There are some limitations caused by the technical capabilities of the PLC-Systems. The PLC-Systems do not allow any kind of inheritance therefore the object orientation is achieved by the use of the standard facilities of the PLC-System. The TO are encapsulating their functions and their data as it is known from classes in the high level languages.

7.1.2 Objective 2: To develop a generic SCADA-System FBD programming language

This objective was accomplished completely. The programming of a Plant can be done completely on the Plant-Diagram provided that the TO library contains all necessary functions and the SCADA-System provides a means to connect the TO together.

7.1.3 Objective 3: To develop a virtual machine for PLC-Systems

This objective was accomplished completely. The virtual machine runs on the PLC-System and the plant function is assured. There is much space for improvements on the VM. The prototype is working but it is a very expensive TO to use. The impact on the cycle time is far too big.

7.1.4 Objective 4: To develop a code generator for PLC-Systems which uses the definition files of a SCADA-System as input

This objective was accomplished completely. The code generator generates the correct PLC code to run on the VM and to ensure the correct functionality of the controlled plant. The selected system has proven to be a good choice as it separates the data handling from the process of code generating. The data handler acts as a kind of backbone while the generator is dealing with the PLC-System and its characteristics. Code generators for further PLC-Systems can be developed without the need to change the data handler. The interface between the data handler and the code generators should be carefully checked and be improved to ensure a reliable operating behaviour.

Chapter 8: Recommendation and Future work

Considering the latest improvements in the standard specification for PLC-Systems, IEC (2003), a series of changes in the aims of the research will take place in the future. The new aim should be the development of a generic and object oriented framework of template objects and code generators for the building automation industry. The framework should consist of the following parts.

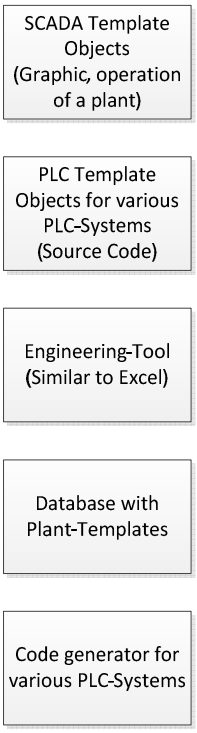

Figure 7.1-1: Future work

The changes should simplify the development process and ensure that the dependencies of the SCADA-Systems can be minimised and the framework can be seen as an autonomous tool. This can be tremendously important in the act of decision making by potential customers.

This work has shown that it is not advisable to depend on only one SCADA system because it restricts the possibilities to act freely. So for future work an effort should be made to establish the framework as an independent, preferably open source tool.

The implementation and development should be done by using open source software wherever possible.

The discussions on improving the energy efficiency of buildings in Switzerland, Energieeffiziente Gebäude Gloor (2009), allows the conclusion that there are great chances for such a framework to achieve success. In the field of building automation there are various SCADA-Systems in service. Some of them are reaching the end of life and have to be replaced. In the majority of cases the whole building automation facilities will be modernised thus a framework that simplifies the engineering process and can work with various modern SCADA-Systems would be a real help for the technicians working on such projects.

Currently the prototypes are working fine in a laboratory environment. There are some known weaknesses in the system. These should be addressed in future work to bring the framework to a point where it can be released to the public.

References

Aho, A.V., Lam, M.S., Sethi, R. and Ullman, J. D. (2007), *Compilers principles, technique & tools,* 2nd edn., Addison Wesley.

Basile, F., Chiacchio, P. and Gerbasio, D. (2013), 'On the implementation of industrial automation systems based on plc'*, IEEE Transactions on Automation Science and Engineering*, 10, pp. 990 - 1003.

Biallas, S., Brauer, J., and Kowalewski, S. (2012), 'Arcade.PLC: a verification platform for programmable logic controllers'*, Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Essen, Germany, 3-7 September, pp 338 - 341. doi: 10.1145/2351676.2351741.

Blanchette, J. and Summerfield, M. (2008), *C++ gui programming with QT 4,* 2nd edn., Prentice Hall.

Bonfè, M., Vignali, M. and Fiorini, M. (2009), 'Plc-based control of a robot manipulator with closed kinematic chain'*, ICRA '09. IEEE International Conference on Robotics and Automation*, Kobe, Japan, 12-17 May, pp 649 - 654. doi: 10.1109/ROBOT.2009.5152281.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2009), *Introduction to algorithms,* 3rd edn.,MIT Press.

Craig, I.D. (2004), *Virtual machines*, Springer.

Czarnecki, K., Eisenecker, U.W. (2000), *Generative programming*, Addison-Wesley

Gay, W.W. (2000), *Linux socket programming by example*, Que.

Gloor, R. (2009), *Energieeffiziente gebäude*, Available at:
http://www.energie.ch/gebaeude / (Accessed: 24.05.2013).

Greenfield, J. and Short, K. (2006), *Software factories: assembling applications with patterns, models, frameworks, and tools*, MITP.

Harel, D. (1987), 'Statecharts: a visual formalism for complex systems'*, Science of Computer Programming*, 8, pp. 231 - 274.

Herrington, J. (2003), *Code generation in action,* Manning Publications Co

IEC (2003), *IEC 61131-3 programmable controllers part 3: programming languages*

ISO (2005), *ISO 16484-3: Systeme der Gebäudeautomation (GA)-Teil3: Funktionen*

Kandare, G., Godena, G. and Strmnik, S. (2003), 'A new approach to plc software design'*, ISA Transactions*, 42, pp. 279 - 288.

Kecher, C. (2006), *Uml 2.0. das umfassende Handbuch,* Galileo Press

Klein, S., Frey, G. and Minas, M. (2003), 'Plc programming with signal interpreted petri nets', *Applications and theory of Petri Nets*, 2679, pp. 440 - 449.

Lee, J., Chun, S. and Kang S. (2002), 'Virtual prototyping of plc-based embedded system using object model of target and behavior model by converting RLL-to-statechart directly*', Journal of systems architecture*, 48 (1-3 September), pp. 17 - 35. doi: 10.1016/S1383-7621(02)00061-9.

Loudon, K. (2000), *Algorithmen in C,* O'Reilly.

MST Systemtechnik AG (2013), *ProMoS NT*, Available at:http://www.promosnt.ch/ / (Accessed: 16. July 2013).

Min Z., Hai W., Chen S., Liangze, Y., Lianyi, Z., Fei, H., and Ming, G. (2013), 'Component-based modeling and code synthesis for cyclic programs'*, IEEE 37th Annual Conference on Computer Software and Applications (COMPSAC),* Kyoto, Japan, 22-26 July, pp 569 - 578. doi: 10.1109/COMPSAC.2013.91.

University of Paderborn (2006), *Hashing*, Available at: http://www2.cs.uni-paderborn.de/cs/ag-monien/LEHRE/SS06/DuA/14.pdf / (Accessed: 24.07.2013).

Rahm, E. (2013), *Hashing*, Available at: http://dbs.uni-leipzig.de/skripte/ADS2/PDF2/kap2.pdf / (Accessed: 24.07.2013).

Sacha, K. (2005), Automatic code generation for plc controllers*, Proceedings of computer safety, realibility, and security,* 3688, pp. 303 - 316.

Saia-Burgess (2013), *S-Engineering*, Available at: http://www.saia-pcd.com/de/technic/s-engineering/Pages/interpreter.aspx / (Accessed: 16. July 2013).

Sedgwick, R. (1992), *Algorithmen in C++,* Addison-Wesley.

Smith, J.E. and Nair, R. (2005), *Virtual machines,* Elsevier Inc.

Stroustrup, B. (2000), *Die C++ Programmiersprache,* Addison-Wesley.

Thieme, J. and Hanisch, H. (2002), 'Model-based generation of modular plc code using IEC61131 function blocks'*, Proceedings of the 2002 IEEE international symposium on industrial electronics, 1*, pp. 199 - 204. doi: 10.1109/ISIE.2002.1026065.

Vogel-Heuser, B., Obermeier, M., Braun, S., Sommer, K., Jobst, F. and Schweizer, K. (2013), 'Evaluation of a uml-based versus an iec 61131-3-based software engineering approach for teaching plc programming'*, IEEE Transactions on Education,* 56(3), pp. 329 - 335. doi: 10.1109/TE.2012.2226035.

W3C (2013), *State chart xml (scxml): state machine notation for control abstraction*, Available at: http://www.w3.org/TR/scxml/ / (Accessed: 17. August 2013).

Weina, H. and Liwei, L. (2011), 'Plc controlling system of transportation manipulator and simulation debugging'*, International Conference on Electronics, Communications and Control*, Zhejiang, China, 9-11 September, pp 957 - 960. doi: 10.1109/ICECC.2011.6066729.

Wikipedia (2013), *Inter-process communication*, Available at: https://en.wikipedia.org/wiki/Inter-process_communication / (Accessed: 18.07.2013).

Wikipedia (2013), *Scada*, Available at: http://en.wikipedia.org/wiki/SCADA / (Accessed: 16. July 2013).

Wikipedia (2013), *Hash table*, Available at: http://en.wikipedia.org/wiki/Hash_table / (Accessed: 23.07.2013).

Wikipedia (2013), *Programmable logic controller*, Available at: https://en.wikipedia.org/wiki/Programmable_logic_controller / (Accessed: 16. July 2013).

Yoo, J., Lee, J. and Lee, J. (2013), 'A research on seamless platform change of reactor protection system from plc to FPGA'*, Nuclear Engineering and Technology*, 45(4, August), pp. 477 - 488. doi: 10.5516/NET.04.2012.078.

Zhou, C. and Chen, H. (2009), 'Development of a plc virtual machine orienting iec 61131-3 standard'*, ICMTMA '09. International Conference on Measuring Technology and Mechatronics Automation*, Zhangjiajie, China, 11-12 April, pp 374 - 379. doi: 10.1109/ICMTMA.2009.422.

Appendix A Source code VM001.src

```
; MODULE: VM001
; -------------

; Version          1 (Parameters generated by PET)
; Date   06.07.12 13:44
; AuthorTg, [Developement]



$SKIP
Ver: 0.5.0

Aenderungen
-----------
06.07.12 TG Erstellt

Kurzbeschreibung
----------------
Virtuelle Maschine zur Bearbeitung der LTO's

$ENDSKIP



; ******************* Includes ***********************
$IFNDEF PETCODE
$INCLUDE ..\frame_code.src
$ENDIF

$IFNDEF AnstPmp
$INCLUDE ..\scoBaseFunc.src
$ENDIF

$IFNDEF tg__vmlib
$INCLUDE ..\VMLib.src
$ENDIF
; **************** Ende Includes ********************



; ******************* XOB 16 ********************** Beim RestartCold ausgeführte Befehle
$INIT
```

```
$ENDINIT
; ***************** Ende XOB 16 *********************


; ************* Statische Definitionen **************
$IFNDEF PETCODE
                EXTN   VM001                    ; external definition (do not change)
                EXTN   __frame.fOnStartUp       ; external definition
                EXTN   __frame.fOnDownLoad      ; external definition
                EXTN   __frame.tObjektAus       ; external definition
                EXTN   __frame.rAlmGroups       ; external definition
                EXTN   __frame.rQuitGroups      ; external definition
$ENDIF


$GROUP sco__VM001                               ; Definition der Gruppe
; Variablen                                     ; Globale Variablen erstellen


; ********** Ende Statische Definitionen ************


; ***************** Functionblock *******************

                FB      VM001                   ; Virtual Machine to be used with the LTO01
                CFG_CycleTime       DEF = 1     ; [01] CFG_CycleTime
                CFG_Address         DEF = 2     ; [02] CFG_Address
                CFG_Pointer         DEF = 3     ; [03] CFG_Pointer
                rLtoAddress         DEF = 4     ; [04]* Address of the actual LTO
                rInputAddresses     DEF = 5     ; [05]* Array of register
                rInputLogics        DEF = 6     ; [06]* Array of registers
                rInputLogicsAdr     DEF = 7     ; [07]* Pointer to an Array of registers
                rOutputAddresses    DEF = 8     ; [08]* Array of registers

; ***************** Lokale Variablen *****************
LOCALS:
WD_Signal               TEQU   F                ; temporary register

rSystemCounter01        TEQU R                  ; temporary register
rSystemCounter02        TEQU R                  ; temporary register
```

```
rSystemCounter03      TEQU  R          ; temporary register
rNumberOfLtos         TEQU  R          ; temporary register
rInputAddress         TEQU  R          ; temporary register
rInputLogic           TEQU  R          ; temporary register
rOutputAddress        TEQU  R          ; temporary register
rLtoFunction          TEQU  R          ; temporary register
rSaveIndexGetLtoAddresses TEQU R       ; temporary register
rSaveIndexPutLto      TEQU  R          ; temporary register
rLtoPointer           TEQU  R          ; temporary register
rHrLtoAddress         TEQU  R          ; temporary register
rSaveIndex            TEQU  R          ; temporary register
rSaveNumberOfLtos     TEQU  R          ; temporary register
rAddressPointer       TEQU  R          ; temporary register
rLogicPointer         TEQU  R          ; temporary register
END_LOCALS:
; ********** Lokale Variablen **********


; ********** Startverhalten **********
O_Enable:

        STL     __frame.tObjektAus      ; Netzwiederkehr?
        JR      H END_O_Enable          ; Timer abgelaufen -> normale Verarbeitung

        LDL     CFG_Pointer             ; Den Pointer mit 0 laden
                K 0
        JR      END                     ; immer ans Ende springen


END_O_Enable:
; ********** Ende Startverhalten **********


; ********* Konfigurationspointer laden **********
LOAD_POINTER:

        LDL     CFG_Pointer             ; Den Pointer
                CFG_Address             ; mit der Adresse des DB laden
```

```
END_LOAD_POINTER:
; ******** Ende Konfigurationspointer laden *********


; ************** Calculate Cycletime ****************
CYCLE_TIME_01:

        SYSRD 7000                          ; Read System-Counter
              rSystemCounter01              ; into Register


END_CYCLE_TIME_01:
; ************ Ende Calculate Cycletime *************


; ************** Konfiguration laden ****************
LOAD_CONFIG:
; ********** Get LTO Count

        TFR   CFG_CycleTime                 ; Get data from DB
              K 1                           ; at element 1
              rNumberOfLtos                 ; into register

        DEC   rNumberOfLtos                 ; Subtract 1


END_LOAD_CONFIG:
; ************ Ende Konfiguration laden *************


; ************ Load LTO Configuration ***************
LOAD_LTO_CONFIG:
; ********** Get LTO's

        SEI   K 0                           ; Set  indexregister to 0
        STI   rSaveNumberOfLtos             ; Save Index

        SEI   K 10                          ; Set  indexregister to 10
        STI   rSaveIndexGetLtoAddresses     ; Save Index
```

```
LOAD_LTO_ADDRESS_LOOP:

        ACC     H                               ; Set accu to high state
        COM     WD_Signal                       ; change logical state
        STH     WD_Signal                       ; if flag is high
        OUT     O 255                           ; set watchdog output to high state


        RSI     rSaveIndexGetLtoAddresses       ; restore index


        TFR     CFG_CycleTime                   ; get from db
                rSaveIndexGetLtoAddresses       ; in element
                rHrLtoAddress           ; the address of the actual address into the helper register


        STI     rSaveIndex                      ; Save Index

        SEI     rHrLtoAddress                   ; Load Index with content of helper register

        GETX    R 0                             ; Get DB-Address from Register
                rHrLtoAddress                   ; ant save it into Register

        COPY    rHrLtoAddress                   ; Copy DB-Address
                rLtoAddress                     ; to Register


        RSI     rSaveIndex                      ; Restore Index


        INI     19                              ; increment index
        STI     rSaveIndexGetLtoAddresses       ; Save Index


; ********** Get LTO configuration
; Input Addresses

        SEI     K 0                             ; Set index register to 0
        STI     rSaveIndexPutLto                ; save index register
```

```
        LD      rAddressPointer             ; load register
                K 10                        ; with constant 10

        LD      rLogicPointer               ; looad register
                K 20                        ; with constant 10


LTO_ADDRESS_LOOP:

        ACC     H                           ; Set accu to high state
        COM     WD_Signal                   ; change logical state
        STH     WD_Signal                   ; if flag is high
        OUT     O 255                       ; set watchdog output to high state

        CSF     S.SF.DBLIB.Library          ; Library number
                S.SF.DBLIB.GetDBItem        ; Read a single DB item
                rHrLtoAddress               ; 1 R|K IN, DB number (any DB number)
                rAddressPointer             ; 2 R|K IN, DB item
                rInputAddress               ; 3 R OUT, Value read

        INC     rAddressPointer             ; inkrement register register

        PUTX    rInputAddress               ; Put the input address
                rInputAddresses             ; into the register


; Input Logic

        CSF     S.SF.DBLIB.Library          ; Library number
                S.SF.DBLIB.GetDBItem        ; Read a single DB item
                rHrLtoAddress               ; 1 R|K IN, DB number (any DB number)
                rLogicPointer               ; 2 R|K IN, DB item
                rInputLogic                 ; 3 R OUT, Value read

        INC     rLogicPointer               ; increment register

        PUTX    rInputLogic                 ; Put the input logic
                rInputLogics                ; into the register
```

```
        INI     K 9                         ; increment index
        JR      H LTO_ADDRESS_LOOP          ; jump if index is less than 10


; Outputs
; Load constants

        LD      rSaveIndexPutLto            ; Load register
                K 0                         ; with constant 0

        LD      rLtoPointer                 ; Load register
                K 30                        ; with constant 30

; get output

        CSF     S.SF.DBLIB.Library          ; Library number
                S.SF.DBLIB.GetDBItem        ; Read a single DB item
                rHrLtoAddress               ; 1 R|K IN, DB number (any DB number)
                rLtoPointer                 ; 2 R|K IN, DB item
                rOutputAddress              ; 3 R OUT, Value read

        RSI     rSaveIndexPutLto            ; restore index

        PUTX    rOutputAddress              ; Put the output address
                rOutputAddresses            ; into the register

        INI     rSaveIndexPutLto            ; increment register
        INI     rLtoPointer                 ; increment register


; get invers output

        CSF     S.SF.DBLIB.Librar            ; Library number
                S.SF.DBLIB.GetDBItem        ; Read a single DB item
                rHrLtoAddress               ; 1 R|K IN, DB number (any DB number)
                rLtoPointer                 ; 2 R|K IN, DB item
                rOutputAddress              ; 3 R OUT, Value read

        RSI     rSaveIndexPutLto            ; rstore index
```

```
        PUTX   rOutputAddress                ; Put the output address
               rOutputAddresses              ; into the register


; ********** Process LTO's
; Check function

        CSF    S.SF.DBLIB.Library            ; Library number
               S.SF.DBLIB.GetDBItem          ; Read a single DB item
               rHrLtoAddress                 ; 1 R|K IN, DB number (any DB number)
               0                             ; 2 R|K IN, DB item
               rLtoFunction                  ; 3 R OUT, Value read

        CMP    rLtoFunction                  ; Check function
               K 0                           ; is it 0?
        JR     Z VM_AND                      ; jump if yes

        CMP    rLtoFunction                  ; Check function
               K 1                           ; is it 1?
        JR     Z VM_OR                       ; jump if yes

        CMP    rLtoFunction                  ; Check function
               K 2                           ; is it 2?
        JR     Z VM_XOR                      ; jump if yes

        JR     CYCLE_TIME_02                 ; jump allways


; Function: AND
VM_AND:
rBaseLogic     TEQU   R                      ; Temporary Register

        LDL    rBaseLogic                    ; Load register
               rInputLogicsAdr               ; with the address of the inpzt logics
```

```
        CFB     fbAnd                   ; Call Function AND
                rLtoAddress             ; Pointer to the actual LTO
                rInputAddresses         ; Pointer to the input addresses of the actual LTO
                rBaseLogic              ; Pointer to the input logic's of the actual LTO
                rOutputAddresses        ; Pointer to the output addresses of the actual LTO

        JR      CHK_FOR_MORE            ; jump allways


; Function: OR
VM_OR:

        LDL     rBaseLogic              ; Load register
                rInputLogicsAdr         ; with the address of the inpzt logics

        CFB     fbOr                    ; Call Function OR
                rLtoAddress             ; Pointer to the actual LTO
                rInputAddresses         ; Pointer to the input addresses of the actual LTO
                rBaseLogic              ; Pointer to the input logic's of the actual LTO
                rOutputAddresses        ; Pointer to the output addresses of the actual LTO

        JR      CHK_FOR_MORE            ; jump allways


; Function: XOR
VM_XOR:

        CFB     fbXor                   ; Call Function XOR
                rLtoAddress             ; Pointer to the actual LTO
                rInputAddresses         ; Pointer to the input addresses of the actual LTO
                rInputLogics            ; Pointer to the input logic's of the actual LTO
                rOutputAddresses        ; Pointer to the output addresses of the actual LTO


; ********** Check if there are more LTO's
CHK_FOR_MORE:

        RSI     rSaveNumberOfLtos       ; restore index

        INI     rNumberOfLtos           ; increment index
```

```
        JR      H LOAD_LTO_ADDRESS_LOOP          ; jump to loop if there are more lto's


END_LOAD_LTO_CONFIG:
; *********** End Load LTO Configuration ************


; ************** Calculate Cycletime *****************
CYCLE_TIME_02:
; ********** Read Systemcounter

        SYSRD 7000                               ; Read System-Counter
              rSystemCounter02                   ; into Register


; ********** Subtract Registers

        SUB   rSystemCounter02                   ; Subtract from value 2
              rSystemCounter01                   ; the value 1
              rSystemCounter03                   ; store the result


; ********** Store the Result

        TFR   rSystemCounter03                   ; Move the Cycletime
              CFG_CycleTime                      ; into DB
              K 0                                ; at element 0


END_CYCLE_TIME_02:
; ************ Ende Calculate Cycletime *************


; *************** End of Functionblock **************
END:

        EFB                                      ; End Functionblock
$ENDGROUP
```

Appendix B Source code VMLib.src

```
; ********************************************************************************
; Library for the virtual machine
; -------------------------------
;
; Datum: 13.07.2012
; SPS            : PCD3.M5540
; Datei          : VMLib.src
; PG5-Version : 2.0.220.0
; Autor          : Thomas Gasser
; ©              : 2012 by Thomas Gasser
;
; Version    : 0.5.0
; Aenderungen :

$SKIP
31.08.12 TG Added function fbLoadAdr
13.07.12 TG Erstellt

$ENDSKIP
;
; ********************************************************************************
$IFNDEF tg__vmlib                                          ; prevent multiple inclusion

; ****************** Includes **********************
$IFNDEF __frame.rAdToken
$INCLUDE ..\frame.src
$ENDIF

$IFNDEF __frameframeframe
$INCLUDE ..\frame_code.src
$ENDIF
; *************** Ende Includes ********************



; ************* Statische Definitionen **************
$IFNDEF PETCODE
                EXTN   __frame.fOnStartUp          ; Import Symbol
                EXTN   __frame.fOnDownLoad         ; Import Symbol
                EXTN   __frame.fEins               ; Import Symbol
```

```
                EXTN   __frame.fNull                    ; Import Symbol
                EXTN   __frame.rNull                    ; Import Symbol
                EXTN   __frame.DD                       ; Import Symbol
$ENDIF

; ********** Functionblocks
fbAnd           PEQU   FB                ; Functionblock for the AND-Function
fbOr            PEQU   FB                ; Functionblock for the OR-Function
fbXor           PEQU   FB                ; Functionblock for the XOR-Function
fbLoadAdr       PEQU   FB                ; Functionblock for the Load Addresses-Function

; ********** Ende Statische Definitionen ************

$GROUP tg__vmlib                                        ; Definition der Gruppe
; ******************* Start Lib *********************


; ***************** Function: AND *******************
; ***************** Functionblock *******************

        FB      fbAnd                                   ; Call Function AND
                rLtoAddress             DEF = 1         ; Pointer to the actual LTO
                rInputAddresses         DEF = 2; Pointer to the input addresses of the actual LTO
                rInputLogics            DEF = 3         ; Pointer to the input logic's of the actual LTO
                rOutputAddresses        DEF = 4;        ;Pointer output addresses of the actual LTO


; ***************** Lokale Variablen *****************
FUNCTION_AND_LOCALS:
fInputs                 TEQU   F [10]                   ; Temporary Flags
fLogics                 TEQU   F [10]                   ; Temporary Flags
fInputsLinked           TEQU   F [10]                   ; Temporary Flags
fHfOutput               TEQU   F                        ; Temporary Flag
fTempLogic              TEQU   F                        ; Temporary Flag

rInputCount             TEQU   R                        ; Temporary Register
rHrLtoAddress  TEQU    R                                ; Temporary Register
rSaveTempIndex          TEQU   R                        ; Temporary Register
rSaveInputAdresses TEQU R                               ; Temporary Register
rSaveInputLogics        TEQU   R                        ; Temporary Register
```

```
FUNCTION_AND_END_LOCALS:
; ********** Lokale Variablen **********


; ********** Get Input Count **********
FUNCTION_AND_GET_CNT:

        COP     rLtoAddress                     ; Copy lto-address
                rHrLtoAddress                   ; to helper register

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.GetDBItem            ; Read a single DB item
                rHrLtoAddress                   ; 1 R|K IN, DB number (any DB number)
                1                               ; 2 R|K IN, DB item
                rInputCount                     ; 3 R OUT, Value read

        SUB     rInputCount                     ; Subtract from InputCount
                1                               ; the constant 1
                rInputCount                     ; and save the value


FUNCTION_AND_END_GET_CNT:
; ********** End Get Input Count **********


; ********** Process function **********
FUNCTION_AND_PROCESS:
; ********** Get Inputs

        SEI     0                               ; Set index to 0
        STI     rSaveTempIndex                  ; save index

        SEI     rInputAddresses             ; Set index register to the base addresse of the input
        STI     rSaveInputAdresses              ; save index

        SEI     rInputLogics            ; Set index register to the base addresse of the input-logics
        STI     rSaveInputLogics                ; save index


FUNCTION_AND_INPUT_LOOP:
```

```
        RSI     rSaveInputAdresses          ; restore index
        STHX    F 0                         ; If input is high

        RSI     rSaveTempIndex              ; restore index
        OUTX    fInputs                     ; set flag to high too


; ********** Get Logics
FUNCTION_AND_LOGIC_LOOP:

        RSI     rSaveInputLogics            ; restore index

        BITOX   1                           ; Get one bit
                R 0                         ; out of the register
                fTempLogic                  ; onto the flag
        STI     rSaveInputLogics            ; save index

        RSI     rSaveTempIndex              ; restore index

        STH     fTempLogic                  ; if temporary flag is high
        OUTX    fLogics                     ; set flag to high too

        STI     rSaveTempIndex              ; save index

        RSI     rSaveInputLogics            ; restore index
        INI     8191                        ; increment index
        STI     rSaveInputLogics            ; save index

        RSI     rSaveTempIndex              ; restore index
        INI     rInputCount                 ; increment index
        STI     rSaveTempIndex              ; save index

        GETX    rInputAddresses             ; copy address of next input
                rSaveInputAdresses          ; to register

        JR      H FUNCTION_AND_INPUT_LOOP   ; jump to loop if there are more inputs


; ********** Link input-logic
```

```
        SEI     0                                   ; Set index to 0

FUNCTION_AND_INPUTLOGIC_LOOP:

        STHX    fInputs                             ; Link inputs
        XORX    fLogics                             ; with logic
        OUTX    fInputsLinked                       ; and save it to a new array

        INI     rInputCount                         ; increment index
        JR      H FUNCTION_AND_INPUTLOGIC_LOOP; jump to loop if more inputs to link


; ********** Process function

        SEI     0                                   ; Set index to 0

        ACC     H                                   ; Set accu to high state
        SET     fHfOutput                           ; Set helper flag to high state

FUNCTION_AND_FUNCTION_LOOP:

        STH     fHfOutput                           ; Link the state of the helper flag with the
        ANHX    fInputsLinked                       ; state of the linked input
        OUT     fHfOutput                           ; set the helper flag to the state of the input

        INI     rInputCount                         ; increment index
        JR      H FUNCTION_AND_FUNCTION_LOOP; jump to loop if more inputs to check

; ********** Set outputs

        SEI     rOutputAddresses            ; Set index to the base address of the output array

        STH     fHfOutput                           ; If the helper flag is high
        OUTX    F 0                                 ; then set the output to high too
        OUTLX   F 1                                 ; and set the inverted output to low or high


FUNCTION_AND_END_PROCESS:
; ************* End Process function ****************
```

```
; *************** End of Functionblock **************
FUNCTION_AND_END:

        EFB                                     ; End Functionblock

; ************** End Function: AND ******************


; ***************** Function: OR ********************

        FB      fbOr                            ; Call Function OR
                rLtoAddress     DEF = 1         ; Pointer to the actual LTO
                rInputAddresses DEF = 2 ; Pointer to the input addresses of the actual LTO
                rInputLogics    DEF = 3         ; Pointer to the input logic's of the actual LTO
                rOutputAddresses DEF = 4        ; Pointer to the output addresses of the actual LTO


; **************** Lokale Variablen *****************
FUNCTION_OR_LOCALS:
fInputs         TEQU   F [10]                   ; Temporary Flags
fLogics         TEQU   F [10]                   ; Temporary Flags
fInputsLinked   TEQU   F [10]                   ; Temporary Flags
fHfOutput       TEQU   F                        ; Temporary Flag
fTempLogic      TEQU   F                        ; Temporary Flag

rInputCount     TEQU   R                        ; Temporary Register
rHrLtoAddress   TEQU   R                        ; Temporary Register
rSaveTempIndex TEQU R                           ; Temporary Register
rSaveInputAdresses TEQU R                       ; Temporary Register
rSaveInputLogics TEQU R                         ; Temporary Register
FUNCTION_OR_END_LOCALS:
; **************** Lokale Variablen *****************


; *************** Get Input Count ******************
FUNCTION_OR_GET_CNT:

        COPY   rLtoAddress                      ; Copy lto-address
```

```
        rHrLtoAddress                           ; to helper register

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.GetDBItem            ; Read a single DB item
                rHrLtoAddress                   ; 1 R|K IN, DB number (any DB number)
                1                               ; 2 R|K IN, DB item
                rInputCount                     ; 3 R OUT, Value read

        SUB     rInputCount                     ; Subtract from InputCount
                1                               ; the constant 1
                rInputCount                     ; and save the value


FUNCTION_OR_END_GET_CNT:
; ************* End Get Input Count *****************


; ***************** Process function *****************
FUNCTION_OR_PROCESS:
; ********** Get Inputs

        SEI     0                               ; Set index to 0
        STI     rSaveTempIndex                  ; save index

        SEI     rInputAddresses         ; Set index register to the base address of the input
        STI     rSaveInputAdresses              ; save index

        SEI     rInputLogics            ; Set index register to the base address of the input-logics
        STI     rSaveInputLogics                ; save index

        FUNCTION_OR_INPUT_LOOP:

        RSI     rSaveInputAdresses              ; restore index
        STHX    F 0                             ; If input is high

        RSI     rSaveTempIndex                  ; restore index
        OUTX    fInputs                 ; set flag to high too


; ********** Get Logics
```

```
FUNCTION_OR_LOGIC_LOOP:

        RSI     rSaveInputLogics                ; restore index

        BITOX   1                               ; Get one bit
                R 0                             ; out of the register
                F fTempLogic                    ; onto the flag
        STI     rSaveInputLogics                ; save index

        RSI     rSaveTempIndex                  ; restore index

        STH     fTempLogic                      ; if temporary flag is high
        OUTX    fLogics                         ; set flag to high too

        STI     rSaveTempIndex                  ; save index

        RSI     rSaveInputLogics                ; restore index
        INI     8191                            ; increment index
        STI     rSaveInputLogics                ; save index

        RSI     rSaveTempIndex                  ; restore index
        INI     rInputCount                     ; increment index
        STI     rSaveTempIndex                  ; save index

        GETX    rInputAddresses                 ; copy address of next input
                rSaveInputAdresses              ; to register

        JR      H FUNCTION_OR_INPUT_LOOP        ; jump to loop if there are more inputs


; ********** Link input-logic

        SEI     0                               ; Set index to 0

FUNCTION_OR_INPUTLOGIC_LOOP:

        STHX    fInputs                 ; Link inputs
        XORX    fLogics                 ; with logic
        OUTX    fInputsLinked           ; and save it to a new array
```

```
        INI     rInputCount                             ; increment index
        JR      H FUNCTION_OR_INPUTLOGIC_LOOP           ; jump to loop if more inputs to link


; ********** Process function

        SEI     0                                       ; Set index to 0

        ACC     H                                       ; Set accu to high state
        RES     fHfOutput                               ; Reset helper flag to high state

FUNCTION_OR_FUNCTION_LOOP:

        STH     fHfOutput                               ; Link the state of the helper flag with
        ORHX    fInputsLinked                           ; state of the linked input
        OUT     fHfOutput                               ; set theflagthe state of the input

        INI     rInputCount                             ; increment index
        JR      H FUNCTION_OR_FUNCTION_LOOP  ; jump to loop if there are more inputs to check

; ********** Set outputs

        SEI     rOutputAddresses                ; Set index to the base address of the output array

        STH     fHfOutput                               ; If the helper flag is high
        OUTX    F 0                                     ; then set the output to high too
        OUTLX F 1                                       ; and set the inverted output to low


FUNCTION_OR_END_PROCESS:
; ************* End Process function ****************


; *************** End of Functionblock **************
FUNCTION_OR_END:

        EFB                                             ; End Functionblock

; ************** End Function: OR ******************
```

```
; ***************** Function: XOR ******************

        FB      fbXor                                   ; Call Function XOR
        rLtoAddress     DEF  = 1                        ; Pointer to the actual LTO
        rInputAddresses         DEF = 2         ; Pointer to the input addresses of the actual LTO
        rInputLogics    DEF = 3                         ; Pointer to the input logic's of the actual LTO
        rOutputAddresses        DEF = 4         ; Pointer to the output addresses of the actual LTO


; ***************** Lokale Variablen *****************
FUNCTION_XOR_LOCALS:
FUNCTION_XOR_END_LOCALS:
; ***************** Lokale Variablen *****************


; ***************** Do something *********************
FUNCTION_XOR:

        COPY  rLtoAddress
              rLtoAddress


        COPY  rInputAddresses
              rInputAddresses


        COPY  rInputLogics
              rInputLogics


        COPY  rOutputAddresses
              rOutputAddresses


END_FUNCTION_XOR:
; *************** End Do something ******************


; *************** End of Functionblock **************
```

```
FUNCTION_XOR_END:

        EFB                                             ; End Functionblock

; ************** End Function: XOR ******************


; ************ Function: Load Addresses *************

        FB      fbLoadAdr                               ; Call Function Load Addresses
                dbLtoAddress    DEF = 1                 ; DB address of the actual LTO
                bInput00        DEF = 2                 ; Address of the input 00
                bInput01        DEF = 3                 ; Address of the input 01
                bInput02        DEF = 4                 ; Address of the input 02
                bInput03        DEF = 5                 ; Address of the input 03
                bInput04        DEF = 6                 ; Address of the input 04
                bInput05        DEF = 7                 ; Address of the input 05
                bInput06        DEF = 8                 ; Address of the input 06
                bInput07        DEF = 9                 ; Address of the input 07
                bInput08        DEF = 10                ; Address of the input 08
                bInput09        DEF = 11                ; Address of the input 09


; **************** Lokale Variablen *****************
LOAD_ADR_LOCALS:
rDbAddr         TEQU   R                                ; Temporary register
rDbItem         TEQU   R                                ; Temporary register
rInputAddr      TEQU   R                                ; Temporary register
rLtoAddr        TEQU   R                                ; Temporary register

kAddressOffset  EQU    10                               ; Constant value (addressoffset into the db)
LOAD_ADR_END_LOCALS:
; **************** Lokale Variablen *****************


; ***************** Load Addresses ******************
LOAD_ADR:
; ********** Load TO Adress

        LDL     rDbAddr                                 ; Load register
```

```
                dbLtoAddress                    ; with db address of the lto


; ********** Load Offset

        LDL     rDbItem                         ; Load item-pointer
                kAddressOffset                  ; with 10 (offset into the db)


; ********** Input 0

        LDL     rInputAddr                      ; Load address of input 0 into register
                bInput00;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 1

        LDL     rInputAddr                      ; Load address of input 1 into register
                bInput01;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 2

        LDL     rInputAddr                      ; Load address of input 2 into register
```

```
                bInput02;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 3

        LDL     rInputAddr                      ; Load address of input 3 into register
                bInput03;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 4

        LDL     rInputAddr                      ; Load address of input 4 into register
                bInput04                        ;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 5
```

```
        LDL     rInputAddr                      ; Load address of input 5 into register
                bInput05;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 6

        LDL     rInputAddr                      ; Load address of input 6 into register
                bInput06                        ;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem        ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 7

        LDL     rInputAddr                      ; Load address of input 7 into register
                bInput07                        ;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem            ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset
```

```
; ********** Input 8

        LDL     rInputAddr                      ; Load address of input 8 into register
                bInput08                        ;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem        ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written

        INC     rDbItem                         ; increment db offset


; ********** Input 9

        LDL     rInputAddr                      ; Load address of input 9 into register
                bInput09                        ;

        CSF     S.SF.DBLIB.Library              ; Library number
                S.SF.DBLIB.SetDBItem        ; Write a single DB item
                rDbAddr                         ; 1 R|K IN, DB number (any DB number)
                rDbItem                         ; 2 R|K IN, DB item
                rInputAddr                      ; 3 R|K IN, Value to be written


END_LOAD_ADR:
; *************** End Load Addresses ****************

; *************** End of Functionblock **************
LOAD_ADR_END:

        EFB                                     ; End Functionblock

; ********** End Function: Load Addresses ***********

; ******************** End Lib **********************
$ENDGROUP
$ENDIF
```

Appendix C Source code LTO01.src

```
; MODULE: LTO01
; -------------

; Version       1 (Parameters generated by PET)
; Date:         06.07.12 10:10
; Author:       Tg, [Developement]

$SKIP
Ver: 1.0

Aenderungen
-----------
31.01.12 TG Erstellt

Kurzbeschreibung
----------------
Logik Vorlagen Objekt

$ENDSKIP

; ******************** XOB 16 ************************
$INIT                                                   ; Beim RestartCold ausgeführte Befehle
$ENDINIT
; ***************** Ende XOB 16 **********************


; ******************** Includes ************************
$IFNDEF __frame.rAdToken
$INCLUDE ..\frame.src
$ENDIF

$IFNDEF __frameframeframe
$INCLUDE ..\frame_code.src
$ENDIF

$IFNDEF AnstPmp
$INCLUDE ..\scoBaseFunc.src
$ENDIF
; ***************** Ende Includes ********************
```

```
; ******************** XOB 16 ************************      ; Beim RestartCold ausgeführte Befehle
$INIT

$ENDINIT
; ***************** Ende XOB 16 **********************


; ************* Statische Definitionen **************
$IFNDEF PETCODE
        EXTN   LTO01                                      ; external definition (do not change)
        EXTN   __frame.fOnStartUp                         ; external definition
        EXTN   __frame.fOnDownLoad                        ; external definition
        EXTN   __frame.tObjektAus                         ; external definition
        EXTN   __frame.rAlmGroups                         ; external definition
        EXTN   __frame.rQuitGroups                        ; external definition
$ENDIF


$GROUP sco__LTO01                                         ; Definition der Gruppe
; Variablen                                               ; Globale Variablen erstellen


; ********** Ende Statische Definitionen *************


; ***************** Functionblock ********************


        FB      LTO01                                     ; Logic Template Object (do not change)
                CFG_Function            DEF = 1                 ; [01] CFG_Function
                CFG_Address             DEF = 2                 ; [02] CFG_Address
                OUT_CFG_Pointer         DEF = 3                 ; [03] IN_VM_PTR
                OUT_Output              DEF = 4                 ; [04] OUT_Output
                OUT_OutputInvers        DEF = 5                 ; [05] OUT_OutputInvers
                OutputPointer           DEF = 6                 ; [06] OUT_Output (Address)
                OutputInversPointer     DEF = 7                 ; [07] OUT_OutputInvers (Address)
                rAddressOutput          DEF = 8                 ; [08]* rAddressOutput
```

```
; ********** Lokale Variablen **********
LOCALS:
END_LOCALS:
; ********** Lokale Variablen **********


; ********** Startverhalten **********
O_Enable:

        STL     __frame.tObjektAus                      ; Netzwiederkehr?
        JR      H END_O_Enable                  ; Timer abgelaufen -> normale Verarbeitung


; ********** Configuration Pointer

        LDL     OUT_CFG_Pointer                         ; Den Pointer mit 0 laden
                K 0


; ********** Output

        LDL     rAddressOutput                          ; Den Pointer
                OutputPointer                   ; mit der Adresse des Ausgangs laden laden


        TFR     rAddressOutput                  ; Save the address of the output
                CFG_Function                    ; into the db
                K 30                            ; into element 30


; ********** Output Invers

        LDL     rAddressOutput                          ; Den Pointer
                OutputInversPointer             ; mit der Adresse des Ausgangs-Invers laden laden


        TFR     rAddressOutput                  ; Save the address of the output
                CFG_Function                    ; into the db
                K 31                            ; into element 31
```

```
; ********** Outputs

        ACC     H                               ; Akku immer auf high
        RES     OUT_Output                      ; den Ausgang auf low
        OUT     OUT_OutputInvers                ; den Inversen Ausgang auf high


; ********** End

        JR      END                             ; immer ans Ende springen


END_O_Enable:
; *************** Ende Startverhalten ***************


; ********* Konfigurationspointer laden *************
LOAD_POINTER:

        LDL     OUT_CFG_Pointer                 ; Den Pointer
                CFG_Address                     ; mit der Adresse des DB laden


END_LOAD_POINTER:
; ******** Ende Konfigurationspointer laden *********


; ************** Outputpointers laden ***************
LOAD_OUTPUT_ADDRESS:
; ********** Output

        LDL     rAddressOutput                  ; Den Pointer
                OutputPointer                   ; mit der Adresse des Ausgangs laden laden


        TFR     rAddressOutput                  ; Save the address of the output
                CFG_Function                    ; into the db
                K 30                            ; into element 30
```

```
; ********** Output Invers

        LDL     rAddressOutput              ; Den Pointer
                OutputInversPointer         ; mit der Adresse des Ausgangs-Invers laden laden


        TFR     rAddressOutput                  ; Save the address of the output
                CFG_Function                    ; into the db
                K 31                            ; into element 31


END_LOAD_OUTPUT_ADDRESS:
; ************ Ende Outputpointers laden ************


; *************** End of Functionblock **************
END:

        EFB                                     ; End Functionblock
$ENDGROUP
```

Appendix D Sources pTool

Appendix D I Source code pTool.h

```
#ifndef PTOOL_H
#define PTOOL_H

#include <QtGui>                          // QT header
#include "ui_ptool.h"                     // ui header
#include "qcProject.h"                    // qcProject header
#include "common.h"                       // common things
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pTSocketLib/ptsocketlib.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pTSocketLib/ptservermachine.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pLib/pLib.h"

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

//!  This is the pTool class, it is the base class oft the pTool application.
/*!
  This class is the base class of the pTool application. This application is used
  to simplify the development with the ProMoS NT SCADA System. It works as starting
  point for the tools to be developed in te future. It works as a starter application.
*/
class pTool : public QSystemTrayIcon {
	Q_OBJECT

public:
	explicit pTool();
	~pTool();

private:
	// dll einbinden
	pLib *libUtility;

	// dll einbinden
	pTSocketLib *socketLib;

	// Client machine
	ptServerMachine *serverMachine;

	// Attributes
	Ui::pToolClass ui;
	QSettings *qsSettings;
	QMenu qmMenu;
	QMenu *qmProjectMenu;
	QMenu *qmSaiaMenu;
	int retryDelaySec;
	QHash<QString, qcProject*> qhProjects;
	QStringList qslProjectNames;
	QFileSystemWatcher *qfswWatcher;

	// Member Functions
	void createContextMenu(void);
	void openProject(void);
	void openLastProject(QString qsProjectPath);

private slots:
	void on_openProject_clicked();
```

```
        void on_closeProject_clicked();
        void on_qfswWatcher_fileChanged(const QString &);
        void on_showDialog_clicked();

        void if_executeCommand(const struct struCommand *struInCmd);
};

#endif // PTOOL_H
```

Appendix D II Source code main.cpp

```
#include "stdafx.h"
#include "ptool.h"
#include <QtGui/QApplication>
#include <QtGlobal>

//! This Function receives the debug Messages and shows them.
    /*!
         \param type A QtMsgType specifing the type of the recceived message.
         \param msg A const char* which holds the message to insert into the message
window.
      \return No return value
    */
void debugOutput(QtMsgType type, const char* msg) {
       // Create static Message Browser and set it up
       static QTextBrowser* qtbMessageBrowser = new QTextBrowser();
       qtbMessageBrowser->setWindowTitle("Debug Window");
       qtbMessageBrowser->move(0, 0);
       qtbMessageBrowser->show();

       // Check which message type it is
       switch(type) {
              // Debug message
              case QtDebugMsg:
                     qtbMessageBrowser->append(QString("Debug : %1").arg(msg));
                     break;
              // Warning
              case QtWarningMsg:
                     qtbMessageBrowser->append(QString("Warning     : %1").arg(msg));
                     break;
              // Critical
              case QtCriticalMsg:
                     qtbMessageBrowser->append(QString("Critical    : %1").arg(msg));
                     break;
              // Fatal
              case QtFatalMsg:
                     qtbMessageBrowser->append(QString("Fatal : %1").arg(msg));
                     break;
       }
}

int main(int argc, char *argv[]) {

       QApplication app(argc, argv);
       #ifdef Q_OS_WIN
              #ifndef QT_NO_DEBUG_OUTPUT
                     qInstallMsgHandler(debugOutput);
              #endif
       #endif
       app.setApplicationName(app.translate("main", "pTool"));
       app.setOrganizationName("tG Soft");
       app.setOrganizationDomain("tGSoft.ch");
       app.setQuitOnLastWindowClosed(false);
       pTool w;
       w.show();
       return app.exec();
}
```

Appendix D III Source code pTool.cpp

```
/*****************************************************************************
 Implementation of the pTool
 ---------------------------

 Date         : 27.01.2011
 File       : pTool.cpp
 Author             : Thomas Gasser
 ©          : 2011 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's ProMoS NT Tool.
/*!
  This Tool working as starter application for the various tools to be developed
  in the future.
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "ptool.h"                      // Project header
#include "qcproject.h"                  // Project header
/****************** Includes ************************/


// Constructor / Destructor
//! This is the Constructor of the pTool application.
    /*!
         Here the on start stuff is made
    */
pTool::pTool(): QSystemTrayIcon(), retryDelaySec(1) {
       // Local's
       bool bTest = false;

       // Set the applications icon
       setIcon(QIcon("Resources/Tools.png"));

       // dll einbinden
       libUtility = new pLib();

       // dll einbinden
       socketLib = new pTSocketLib();

       // Server machine
       serverMachine = new ptServerMachine(this);
       bTest = connect(serverMachine, SIGNAL(executeCommand(const struct struCommand
*)), this,  SLOT(if_executeCommand(const struct struCommand *)));

       // Create the FileSystem Watcher
       qfswWatcher = new QFileSystemWatcher(this);
       // bTest = connect(qfswWatcher, SIGNAL(fileChanged(const QString &)), this,
SLOT(on_qfswWatcher_fileChanged(const QString &)));

       // Create the Context menu
       createContextMenu();
}
```

```
//! This is the Destructor of the pTool application.
    /*!
         Here some stuff is made just before the application is finished
    */
pTool::~pTool() {
       // Write values to pTools.ini File
       qsSettings->setValue("LastProject/Path",
qhProjects.value(qslProjectNames.at(0))->getProjectPath());
       qsSettings->setValue("LastProject/Name",
qhProjects.value(qslProjectNames.at(0))->getProjectName());

       // Delete created project classes
       qDeleteAll(qhProjects);

       // Delete dll
       delete libUtility;

       // Delete FileSystem Watcher
       delete qfswWatcher;
}


// Creator function for the Context menu
//! This is the creator function for the context menu.
    /*!
         This function creates the context menu and populates it with the
      entries for each tool. The tools which are inserted into the menu
         can be configured in the settings file pTool.ini. There are also actions
         to open a project and to exit from the application.
    */
void pTool::createContextMenu() {
       // Locals
       int iqmbRetValue;
       QString qsScratch;
       QStringList qslKeys, qslTools;

       // Insert Project sub-menu
       qmProjectMenu = new QMenu("&Project: ");
       qmMenu.addMenu(qmProjectMenu);
       qmProjectMenu->setIcon(QIcon("Resources/DatabaseBlue.png"));

       // Open a ProMoS NT project
       qmProjectMenu->addAction(QIcon("Resources/FolderOpen.png"), tr("&Open Project"),
this, SLOT(on_openProject_clicked()));

       // Close a ProMoS NT project
       qmProjectMenu->addAction(QIcon("Resources/FolderRemove.png"), tr("&Close
Project"), this, SLOT(on_closeProject_clicked()));

       // Show Project Dialog
       qmProjectMenu->addAction(QIcon("Resources/bullet_wrench.png"), tr("Show
Dialog"), this, SLOT(on_showDialog_clicked()));

       // Insert Saia sub-menu
       qmSaiaMenu = new QMenu("&Saia-Burgess Code-Generatoren");
       qmMenu.addSeparator();
       qmMenu.addMenu(qmSaiaMenu);
       qmSaiaMenu->setIcon(QIcon("Resources/SaiaBurgess.png"));

       // Create settings object
       qsSettings = new QSettings("../cfg/pTool.ini", QSettings::IniFormat, this);

       // Get last project from pTool.ini file
```

```
	qmProjectMenu->setTitle(qmProjectMenu->title() % qsSettings-
>value("LastProject/Name", tr("No Project active")).toString());

	// Check if there is a last project
	if(qmProjectMenu->title() != "No Project active" ) {
		qsScratch = qsSettings->value("LastProject/Path", tr("No Path
found")).toString();
		if(qsScratch != "No Path found") {
			// Ask if user wants to open last project
			iqmbRetValue = QMessageBox::question(0, "pTool", "Open project: "
% qsSettings->value("LastProject/Name", tr("No Project active")).toString(),
QMessageBox::Yes | QMessageBox::No);

			// Do the the right thing
			if(iqmbRetValue == QMessageBox::Yes) {
				// Open last project
				openLastProject(qsScratch);
			}
			else {
				// Set Projectname
				qmProjectMenu->setTitle(tr("Project: ") % tr("No Project
active"));
			}
		}
	}

	// Get tools from pTool.ini file
	qslKeys = qsSettings->allKeys();
	qslKeys = qslKeys.filter("Tools");

	for(int i = 0; i < qslKeys.size(); ++i) {
		qslTools << qsSettings->value(qslKeys.at(i), "No entry").toString();
	}

	// Fill the Tools into the qmenu
	QActionGroup *group = new QActionGroup(this);
	foreach (const QString &qsTools, qslTools) {
		QAction *action = qmSaiaMenu->addAction(qsTools);
		group->addAction(action);
		action->setData(qsTools);
	}

	// Add the exit action
	qmMenu.addSeparator();
	qmMenu.addAction(QIcon("Resources/exit.png"), tr("E&xit"), qApp, SLOT(quit()));
	setContextMenu(&qmMenu);
}


//! This is the Slot to open a project
    /*!
         This ist the slot which opens a project. A new instance of the qcProject class
will
         be created.
    */
void pTool::on_openProject_clicked() {
	// locals
	QString qsFileName, qsToolPath;

	// Create object of pProject
	if(qslProjectNames.isEmpty()) {
		// open the input dialog
		qsToolPath = qsSettings->value("ToolPath/Path", "C:/").toString();
```

```
			// open the input dialog
			qsFileName = QFileDialog::getOpenFileName(0, "Bitte die Projekt Datei
auswählen", qsToolPath, "DMS Files (*.dms)");

			// create object
			qcProject *pProject = new qcProject(0, qsFileName);

			// Project name
			qslProjectNames << qsFileName.section("/", 3, 3);

			// Push new pProject into hash
			qhProjects.insert(qslProjectNames.at(0), pProject);

			// Set QMenu-Title to the active Project Name
			qmProjectMenu->setTitle(tr("Project: ") %
qhProjects.value(qslProjectNames.at(0))->getProjectName());

			// Add Project Path to the FileSystem Watcher
			qfswWatcher->addPath(qsFileName);

			// Debug output: Local's
			QStringList qslDebug;
			QString qsDebugString;

			// Debug output: CPU's
			qslDebug << qhProjects.value(qslProjectNames.at(0))->getCpus();
			/*qsDebugString = qhProjects.value(qslProjectNames.at(0))-
>getCpus().at(0);
			for(int i = 0; i < qslDebug.size(); ++i) {
				qDebug() << qslDebug.at(i);
			}*/
		}
		else {
			// Show Message Box and return
			QMessageBox::warning(0, "pTool", tr("Maximum number of Projects allready
opened"));
			return;
		}
}


//! This is the Slot to close a project
    /*!
         This ist the slot which closes a project. The existing instance of the
qcProject class will
        be deleted.
    */
void pTool::on_closeProject_clicked() {
	if(!qslProjectNames.isEmpty()) {
			// Remove Project Path from the FileSystem Watcher
			qfswWatcher->removePath(qhProjects.value(qslProjectNames.at(0))-
>getProjectPath());

			// Delete created project classes
			qhProjects.remove(qslProjectNames.at(0));
			qslProjectNames.clear();

			// Set QMenu-Title to the active Project Name
			qmProjectMenu->setTitle(tr("Project: No active Project"));
		}
		else {
			// Show Message Box and return
```

```
                QMessageBox::warning(0, "pTool", tr("No Project to close"));
                return;
        }

}


//! This Function opens the ProMoS NT Project given in the first parameter. It is used
to open the last Project at startup.
    /*!
      \param QString qsProjectPath A QString with the cpmplet Project Path (including
Filename)
         \return void
    */
void pTool::openLastProject(QString qsProjectPath) {
        // create object
        qcProject *pProject = new qcProject(0, qsProjectPath);

        // Project name
        qslProjectNames << qsProjectPath.section("/", 3, 3);

        // TEST:debug output
        qDebug() << qslProjectNames.at(0);

        // Push new pProject into hash
        qhProjects.insert(qslProjectNames.at(0), pProject);

        // Set QMenu-Title to the active Project Name
        qmProjectMenu->setTitle(tr("Project: ") %
qhProjects.value(qslProjectNames.at(0))->getProjectName());

        // Add Project Path to the FileSystem Watcher
        qfswWatcher->addPath(qsProjectPath % "promos.dms");

        // Debug output: Local's
        QStringList qslDebug;
        QString qsDebugString;

        // Debug output: CPU's
        qslDebug << qhProjects.value(qslProjectNames.at(0))->getCpus();
/*      qsDebugString = qhProjects.value(qslProjectNames.at(0))->getCpus().at(0);
        for(int i = 0; i < qslDebug.size(); ++i) {
                qDebug() << qslDebug.at(i);
        }*/
}


//! This Function is called if one of the Files in the ProMoS NT Project folder (cfg)
has been changed
    /*!
      \param QString qsInPath A QString with the canged Project's Path (including
Filename)
         \return void
    */
void pTool::on_qfswWatcher_fileChanged(const QString &) {
        // Show Message Box and return
        QMessageBox::information(0, "pTool", "Project has been changed");
        return;
}


//! This Function is called if in the context menu the entry show dialog is clicked.
    /*!
```

```
        \return void
    */
void pTool::on_showDialog_clicked() {
      qhProjects.value(qslProjectNames.at(0))->showDialog();
}


//! This Slot is called if the executeCommand-Signal of the serverMachine is
triggered.
    /*!
        \param iCommand An Integer which holds the command to execute.
        \return void
    */
void pTool::if_executeCommand(const struct struCommand *struInCmd) {
      // Debug message
      qDebug() << "Executing command...";

      // Locals
      int iCommand;
      QVariant qvScratch;
      QStringList qslAnswer;

      // Check Command
      iCommand = struInCmd->qsCommand.toInt();

      // execute commands
      switch(iCommand) {
            // Send Project List
            case 0:
                  // Debug output
                  qDebug() << "Command is: " << iCommand << ", sending Project
list";

                  // Clear Answer list
                  qslAnswer.clear();

                  // Sending Project list
                  qvScratch = qslProjectNames.size();
                  qslAnswer << qvScratch.toString();
                  for(int i = 0; i < qslProjectNames.size(); ++i) {
                        qslAnswer << qslProjectNames.at(i);
                  }
            break;

            // Send CPU List
            case 1:
                  // Clear Reslut list
                  qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                  // Clear Answer list
                  qslAnswer.clear();

                  // Debug output
                  qDebug() << "Command is: " << iCommand << ", sending CPU list";

                  // Execute Command
                  if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                        // Sending CPU list
                        for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                              qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
```

```
                        }
                }
                else {
                        qDebug() << "Getting CPU's doesn't work";
                }
        break;

        // Send TO List
        case 2:
                // Clear Reslut list
                qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                // Clear Answer list
                qslAnswer.clear();

                // Debug output
                qDebug() << "Command is: " << iCommand << ", sending TO list";

                // Execute Command
                if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                        // Sending CPU list
                        for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                        }
                }
                else {
                        qDebug() << "Getting TO's doesn't work";
                }
        break;

        // Send Instance List
        case 3:
                // Clear Reslut list
                qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                // Clear Answer list
                qslAnswer.clear();

                // Debug output
                qDebug() << "Command is: " << iCommand << ", sending Instance
list";

                // Execute Command
                if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                        // Sending CPU list
                        for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                        }
                }
                else {
                        qDebug() << "Getting Instances's doesn't work";
                }
        break;

        // Send Data
        case 4:
                // Clear Reslut list
```

```
                        qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                        // Clear Answer list
                        qslAnswer.clear();

                        // Debug output
                        qDebug() << "Command is: " << iCommand << ", sending instances by
typ";

                        // Execute Command
                        if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                                // Sending CPU list
                                for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                        qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                                }
                        }
                        else {
                                qDebug() << "Getting instances by name doesn't work";
                        }
                break;

                // Send Data
                case 5:
                        // Clear Reslut list
                        qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                        // Clear Answer list
                        qslAnswer.clear();

                        // Debug output
                        qDebug() << "Command is: " << iCommand << ", sending Data list";

                        // Execute Command
                        if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                                // Sending CPU list
                                for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                        qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                                }
                        }
                        else {
                                qDebug() << "Getting Data doesn't work";
                        }
                break;

                // Send Comment
                case 6:
                        // Clear Reslut list
                        qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                        // Clear Answer list
                        qslAnswer.clear();

                        // Debug output
                        qDebug() << "Command is: " << iCommand << ", sending Comment
list";

                        // Execute Command
```

```
                        if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                                // Sending CPU list
                                for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                        qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                                }
                        }
                        else {
                                qDebug() << "Getting Comment doesn't work";
                        }
                break;

                // Send Attribute Comment
                case 7:
                        // Clear Reslut list
                        qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                        // Clear Answer list
                        qslAnswer.clear();

                        // Debug output
                        qDebug() << "Command is: " << iCommand << ", sending Attribute
Comment list";

                        // Execute Command
                        if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                                // Sending CPU list
                                for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                        qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                                }
                        }
                        else {
                                qDebug() << "Getting Attribute Comment doesn't work";
                        }
                break;

                // Send Project path
                case 8:
                        // Clear Reslut list
                        qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                        // Clear Answer list
                        qslAnswer.clear();

                        // Debug output
                        qDebug() << "Command is: " << iCommand << ", sending Project
Path";

                        // Execute Command
                        if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                                // Sending CPU list
                                for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                        qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                                }
                        }
```

```
                        else {
                                qDebug() << "Getting Project Path doesn't work";
                        }
                break;

                // Send Data for tree
                case 9:
                        // Clear Reslut list
                        qhProjects.value(struInCmd->qsProject)->qslResults.clear();

                        // Clear Answer list
                        qslAnswer.clear();

                        // Debug output
                        qDebug() << "Command is: " << iCommand << ", sending data for
tree";

                        // Execute Command
                        if(qhProjects.value(struInCmd->qsProject)-
>executeCommand(struInCmd->qsCpu, iCommand) == 0) {
                                // Sending CPU list
                                for(int i = 0; i < qhProjects.value(struInCmd->qsProject)-
>qslResults.size(); ++i) {
                                        qslAnswer << qhProjects.value(struInCmd->qsProject)-
>qslResults.at(i);
                                }
                        }
                        else {
                                qDebug() << "Getting Project Path doesn't work";
                        }
                break;

                // When others
                default:
                        // Debug output
                        qDebug() << "pTool: Unknow Command";

                        // clear answer list
                        qslAnswer.clear();
                break;
        }

        // Give answer back to the server machine
        serverMachine->setData(qslAnswer);

        // clear answer list
        qslAnswer.clear();

        // Return value
        return;
}
```

Appendix D IV Source code qcproject.h

```
#ifndef QCPROJECT_H
#define QCPROJECT_H

#include <QtGui>
#include <common.h>
#include "standardtreemodel.h"
#include "ui_qcproject.h"                // ui header

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

class qcProject : public QDialog {
	Q_OBJECT

public:
	qcProject(QWidget *parent = 0, QString qsFileName = "");
	~qcProject();

	// Public Functions
	QString getProjectName();
	QString getProjectPath();
	QString getPgVersion();
	QString getPromosVersion();
	QStringList getCpus();
	void showDialog();
	int executeCommand(QString qsInCpu, int iInCommand);

	// Public Attributes
	QStringList qslResults;

private:
	// dll einbinden
	pLib *libUtility;

	// Private Functions
	bool createTrees();
	int loadProject(QString qsProjectPath);

	// Command functions
	void cmdGetCpus();
	void cmdGetTos(QString qsInCpu);
	void cmdGetPlantTos(QString qsInCpu);
	void cmdGetInstancesByType(QString qsInCpu);
	void cmdGetData(QString qsInCpu);
	void cmdGetComment(QString qsInCpu);
	void cmdGetAttribComment(QString qsInCpu);
	void cmdGetProjectPath();
	void cmdgetDataForTree(QString qsInCpu);

	// Private Attributes
	Ui::qcproject ui;
    QStringList qslCpus;
	QFile *qfSystemTree;
	QFile *qfCpuTree;

	QHash<QString, StandardTreeModel*> qhTrees;
	QMultiHash<QString, QHash<QString, QString>*> *qmhInstanceHash;
	QHash<QString, QString>    *qhDataHash;
```

```
        QList<QHash<QString, QString>*> qlHashes;

        QSettings *qsetSettings;
        struct struProjectData struProjData;

        // Standard Model
        StandardTreeModel *qtmSystemTreeModel;
        StandardTreeModel *qtmCpuModel;
        StandardTreeModel *qtmWholeTree;

// Private Slots
private slots:
        void safeLoadProject();
        bool executeCommand();
        int showTree();

};

#endif // QCPROJECT_H
```

Appendix D V Source code qcproject.cpp

```
/*****************************************************************************
 Implementation of the qcProject-Class
 -------------------------------------

 Date         : 27.01.2011
 File       : qcProject.cpp
 Author             : Thomas Gasser
 ©          : 2011 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's ProMoS NT Tool' s qcProject Class.
/*!
  This is the class which holds important information about the project
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "qcproject.h"                  // Project header
/****************** Includes ************************/


// Constructor / Destructor
//! This is the Constructor of the qcProject Class.
    /*!
         Here the on start stuff is made
    */
qcProject::qcProject(QWidget *parent, QString qsFileName): QDialog(parent) {
       // Setup GUI
       ui.setupUi(this);

       // Check Path to be correct
       if(!qsFileName.isEmpty() && !qsFileName.isNull()) {
              // Set the Project Path
              struProjData.qsProjectPath = qsFileName;

              // Set the Project Name
              struProjData.qsProjectName = qsFileName.section("/", 3, 3);
       }
       else {
              // Show Message Box and return
              QMessageBox::warning(0, "pTool", tr("No vaild project path"));
       }

       // dll einbinden
       libUtility = new pLib();

       // Create settings object
       qsetSettings = new QSettings("../cfg/pTool.ini", QSettings::IniFormat, this);

       // Read settings
       struProjData.qsFileNameDms = qsetSettings->value("FileNames/DMS", tr("No entry
found")).toString();
       struProjData.qsFileNameTo = qsetSettings->value("FileNames/TO", tr("No entry
found")).toString();

       // remove filename
```

```
	struProjData.qsProjectPath.remove(struProjData.qsFileNameDms,
Qt::CaseInsensitive);

	// Connect Signals
	connect(ui.qpbLaden , SIGNAL(clicked()), this,  SLOT(showTree()));
	connect(ui.qpbExecute, SIGNAL(clicked()), this, SLOT(executeCommand()));

	// fill in combobox
	QStringList qslItems;
	qslItems << "getProjects" << "getCPUs" << "getTos" << "getInstances" <<
"getObjectsByType" << "getData" << "getComment" << "getAttribComment" <<
"getProjectPath" << "getDataForTree";
	ui.qcbCommands->addItems(qslItems);

	// Create System Tree temp File
	qfSystemTree = new QFile(this);

	// Create Hashes
	qmhInstanceHash = new QMultiHash<QString, QHash<QString, QString>*>();

	// Create Cpu Tree temp File
	qfCpuTree = new QFile(this);

	// Load Project
	QTimer::singleShot(0, this, SLOT(safeLoadProject()));
}
//! This is the destructor of the qcProject Class.
    /*!
         Here some stuff is made before the class is deleted
    */
qcProject::~qcProject() {
	// Locals
	QFile qfTreeFile;

	// löschen dll
	delete libUtility;

	// Delete temp Files
	delete qfSystemTree;

	// Delete Cpu-Hash
	qDeleteAll(qhTrees);

	// Delete System-Tree-File
	qfTreeFile.setFileName("../tmp/pSystemTree.dms");
	qfTreeFile.remove();

	// Delete all CPU-Tree-Files
	for(int i = 0; i < qslCpus.size(); ++i) {
		qfTreeFile.setFileName("../tmp/" % qslCpus.at(i) % ".dms");
		qfTreeFile.remove();
	}
}

//! This Function returns the Project Name of the active project.
    /*!
      \return A QString holding the Project Name of the active project
    */
QString qcProject::getProjectName() {
	// Return value
	return struProjData.qsProjectName;
}
```

```
//! This Function returns the Project Path of the active project.
    /*!
      \return A QString holding the Project Path of the active project
    */
QString qcProject::getProjectPath() {
       // Return value
       return struProjData.qsProjectPath;
}

//! This Function returns the Projects Cpus.
    /*!
      \return A QStringList holding the Cpus used in the active project
    */
QStringList qcProject::getCpus() {
       // return value
       return qslCpus;
}

//! This Function returns the PG5-Version actual project.
    /*!
      \return A QString holding the PG5-Version used in the active project
    */
QString qcProject::getPgVersion() {
       // return value
       return struProjData.qsPgVersion;
}

//! This Function returns the ProMoS NT Version actual project.
    /*!
      \return A QString holding the ProMoS NT Version used in the active project
    */
QString qcProject::getPromosVersion() {
       // return value
       return struProjData.qsPromosVersion;
}

//! This is the Create-Trees slot.
    /*!
        The data is loaded from the files in this slot. The System-Tree and a Tree for
each Cpu is created
        There is also a Tree with the whole Project inside
        \return A bool          true := Reading the file was successful,
                                        false := Reading the file was
unsuccessful.
    */
bool qcProject::createTrees() {
       // Create system tree
       qtmSystemTreeModel = new StandardTreeModel(this);

       // Call load-function of the model
       qtmSystemTreeModel->bLoadData("../tmp/pSystemTree.dms");
       qtmSystemTreeModel->setHorizontalHeaderLabels(QStringList() << "System-Tree");

       // Move address into hash
       qhTrees.insert("System", qtmSystemTreeModel);

       // Create a new Model for each cpu
       for(int i = 0; i < qslCpus.size(); ++i) {
              // Create model
              qtmCpuModel = new StandardTreeModel(this);

              // Call load-function of the model
              qtmCpuModel->bLoadData("../tmp/" % qslCpus.at(i) % ".dms");
```

```
                qtmCpuModel->setHorizontalHeaderLabels(QStringList() << qslCpus.at(i) %
"-Tree");

                // Move address into hash
                qhTrees.insert(qslCpus.at(i), qtmCpuModel);
        }

        // Return value
        return true;
}

//! This is the Execute-Comand slot.
    /*!
         Commands sent to the qcProject Class are processed in this slot.
         \return A bool           true := Command executed correctly,
                                                false := Command not executed.
    */
bool qcProject::executeCommand() {
        // Locals
        int iCommand, iTreeIndex;

        // Check if something is highlighted in the tree combobox
        iTreeIndex = ui.qcbTree->currentIndex();

        if(iTreeIndex < 0) {
                QMessageBox::warning(this, "qcProject", "No choosen tree");
                return false;
        }
        else {
                // Get the model out of the hash
                if(qhTrees.isEmpty()) {
                        return false;
                }
        }

        // Get command
        iCommand = ui.qcbCommands->currentIndex();

        // Execute Command
        switch(iCommand) {
                // Command Nbr. 0
                case 0:
                        // clearing result list
                        qslResults.clear();

                        // Getting Project name
                        qslResults << getProjectName();

                        // Show results
                        ui.qtbResults->insertPlainText("Found this Project: " %
qslResults.at(0) % "\n");
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 1
                case 1:
                        // clearing result list
                        qslResults.clear();

                        // Getting Cpu's
                        cmdGetCpus();
```

```
                        // Show results
                        ui.qtbResults->insertPlainText("Found " % qslResults.at(0) % "
CPU's, named: " % "\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 2
                case 2:
                        // clearing result list
                        qslResults.clear();

                        // Find To's
                        cmdGetTos(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found: " % qslResults.at(0) % "
TO's used on CPU " % ui.qcbTree->currentText() % "\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 3
                case 3:
                        // clearing result list
                        qslResults.clear();

                        // Find Plant TO's
                        cmdGetPlantTos(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found: " % qslResults.at(0) % "
Plant TO's on CPU " % ui.qcbTree->currentText() % "\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 4
                case 4:
                        // clearing result list
                        qslResults.clear();

                        // Getting instances by type
                        cmdGetInstancesByType(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found: " % qslResults.at(0) % "
instances with type. These are the following:\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 5
                case 5:
```

```
                        // clearing result list
                        qslResults.clear();

                        // Getting data
                        cmdGetData(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found " % qslResults.at(0) % "
data items, these are: " % "\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 6
                case 6:
                        // clearing result list
                        qslResults.clear();

                        // Getting data
                        cmdGetComment(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found " % qslResults.at(0) % "
comment items, these are: " % "\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 7
                case 7:
                        // clearing result list
                        qslResults.clear();

                        // Getting data
                        cmdGetAttribComment(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found " % qslResults.at(0) % "
comment items, these are: " % "\n");
                        qslResults.removeFirst();
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Command Nbr. 8
                case 8:
                        // clearing result list
                        qslResults.clear();

                        // Getting Project name
                        qslResults << getProjectPath();

                        // Show results
                        ui.qtbResults->insertPlainText("Found this Project: " %
qslResults.at(0) % "\n");
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
```

```
                break;

                // Command Nbr. 9
                case 9:
                        // clearing result list
                        qslResults.clear();

                        // Getting Project name
                        cmdgetDataForTree(ui.qcbTree->currentText());

                        // Show results
                        ui.qtbResults->insertPlainText("Found: " % qslResults.at(0) %
"\n");
                        for(int i = 0; i < qslResults.size(); ++i) {
                                ui.qtbResults->insertPlainText(qslResults.at(i)  % "\n");
                        }
                break;

                // Unknow command
                default:
                        // Debug output
                        qDebug() << "qcProject: Unknow command";

                        // clearing result list
                        qslResults.clear();
        }

        // Return value
        return true;
}

//! This is the Load-Project slot.
    /*!
         The data is loaded from the *.dms file in this slot. The *.dms file is split
to various files. A file for the System-Tree  and a file for each Cpu is created.
         The data ist set into the ui.
         \return An Integer  0 := Settinging up the project successful,
                                                -1 := Settinging up the project
unsuccessful.
    */
int  qcProject::loadProject(QString qsProjectPath) {
        // Locals
        int iRetValue = 0;
        QStringList qslTempList;

        // Check Path to be correct
        if(!qsProjectPath.isEmpty()) {
                // Read System Tree
                libUtility->readTree(qsProjectPath, "System", "pSystemTree.dms",
*qfSystemTree);

                // Get the Cpu's
                qslCpus = libUtility->getCpus(qsProjectPath);

                // Read Cpu tree's
                for(int i = 0; i < qslCpus.size(); ++i) {
                        // Create a new QFile object
                        qfCpuTree = new QFile;

                        // read Cpu-Tree to file
                        libUtility->readTree(qsProjectPath, qslCpus.at(i), qslCpus.at(i) %
".dms", *qfCpuTree);
                }
```

```
                // Fill CPU's into combobox
                qslTempList = qslCpus;
                qslTempList.prepend("System");
                ui.qcbTree->addItems(qslTempList);

                // Load files into models
                createTrees();

                // Set return value
                iRetValue = 0;
        }
        else {
                iRetValue = -1;
        }

        // Return value
        return iRetValue;
}

//! This is a wrapper slot to safely load the project at startup.
    /*!
         This slot is used to call the necessary functions in a safe way and because
slots can't return anything
         \return An void.
    */
void  qcProject::safeLoadProject() {
        // Call loadProject()
        loadProject(struProjData.qsProjectPath % struProjData.qsFileNameDms);

        // Show System-Tree at first
        ui.qcbTree->setCurrentIndex(0);
        showTree();
}

//! This is the Show-Tree-Function.
    /*!
         This Function is used to change the tree shown in the tree-view witget of the
ui.
         \return An Integer  0 := Changing the tree successful,
                                                  -1 := Changing the tree unsuccessful.
    */
int qcProject::showTree() {
        // Locals
        int iTreeIndex;
        int iRetValue = 0;

        // Check if something is highlighted in the tree combobox
        iTreeIndex = ui.qcbTree->currentIndex();

        if(iTreeIndex < 0) {
                QMessageBox::warning(this, "qcProject", "No tree choosen");
                iRetValue = -1;
        }
        else {
                // Get the model out of the hash
                if(!qhTrees.isEmpty()) {
                        // Get the text out of the current item
                        ui.qtvSystemTree->setModel(qhTrees.value(ui.qcbTree-
>currentText()));
                        ui.qtvSystemTree->show();
                        iRetValue = 0;
                }
```

```
		else {
			QMessageBox::warning(this, "qcProject", "Hash is empty");
			iRetValue = -1;
		}
	}

	// Enable Commands only if not System-Tree is shown
	if(ui.qcbTree->currentText() != "System") {
		// Enable Command parts
		ui.qpbExecute->setEnabled(true);
		ui.qtbResults->setEnabled(true);
		ui.qcbCommands->setEnabled(true);
	}
	else {
		// Disable Command parts
		ui.qpbExecute->setEnabled(false);
		ui.qtbResults->setEnabled(false);
		ui.qcbCommands->setEnabled(false);
	}

	// Return value
	return iRetValue;
}

//! This is the Show-Dialog-Function.
    /*!
         This Function is used to show the qcProject ui if it is hidden.
         \return An void.
    */
void qcProject::showDialog() {
	// Show ui
	this->show();
}
//! This is the Execute-Comand slot for commands send from an other application.
    /*!
         Commands sent to the qcProject Class are processed in this slot.
         \return An integer 0 := Command executed correctly,
                                                -2 := Project Hash empty,
                                                -3 := No vaild command,

    */
int qcProject::executeCommand(QString qsInCpu, int iInCommand) {
	// Check command
	if(iInCommand < 0 || iInCommand > 9) {
		QMessageBox::warning(this, "qcProject", "No vaild command");
		return -3;
	}

	// Check if qsInCpu is empty
	if(qsInCpu.isEmpty()) {
		// Command Nbr. 1, get CPU's
		cmdGetCpus();
	}
	else {
		// Execute Command
		switch(iInCommand) {
			// Command Nbr. 1
			case 1:
				cmdGetCpus();
			break;

			// Command Nbr. 2
			case 2:
```

```
                        // Find To's
                        cmdGetTos(qsInCpu);
                    break;

                    // Command Nbr. 3
                    case 3:
                        // Find Plant TO's
                        cmdGetPlantTos(qsInCpu);
                    break;

                    // Command Nbr. 4
                    case 4:
                        // Getting instances by type
                        cmdGetInstancesByType(qsInCpu);
                    break;

                    // Command Nbr. 5
                    case 5:
                        // Getting data
                        cmdGetData(qsInCpu);
                    break;

                    // Command Nbr. 6
                    case 6:
                        // Getting comments
                        cmdGetComment(qsInCpu);
                    break;

                    // Command Nbr. 7
                    case 7:
                        // Getting Attribute comments
                        cmdGetAttribComment(qsInCpu);
                    break;

                    // Command Nbr. 8
                    case 8:
                        // Getting Project path
                        cmdGetProjectPath();
                    break;

                    // Command Nbr. 9
                    case 9:
                        // Getting Project path
                        cmdgetDataForTree(qsInCpu);
                    break;

                    // Unknow command
                    default:
                        // Debug output
                        qDebug() << "qcProject: Unknow command";
                }
            }
            // Return value
            return 0;
}
```

Appendix D VI Source code qcProjectCommands.cpp

```
/*****************************************************************************
 Implementation of the qcProject-Class
 -------------------------------------

 Date         : 27.01.2011
 File       : qcProject.cpp
 Author             : Thomas Gasser
 ©          : 2011 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's ProMoS NT Tool' s qcProject Class.
/*!
  This is the class which holds important information about the project
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "qcproject.h"                  // Project header
/****************** Includes ************************/


// *************** Command Functions ********************

//! This Function implements the getCpus-Command (cmd nbr. :1).
    /*!
      \param - None.
      \return void
    */
void qcProject::cmdGetCpus() {
       // Locals
       QVariant qvResult;
       QStringList qslTempList;

       // Get all Cpu's out of the qHash
       qslTempList = qhTrees.keys();

       // Remove "System"
       qslTempList.removeOne("System");

       // Get CPU count
       qvResult = qslTempList.size();

       // Copy temp list to result list
       qslResults << qslTempList;

       // insert size of the answer at the head of the list itself
       qslResults.prepend(qvResult.toString());

       // Return value
       return;
}

//! This Function implements the getTos-Command (cmd nbr. :2).
    /*!
      \param a QString holding the cpu the command is for.
      \return void
```

```
    */
void qcProject::cmdGetTos(QString qsInCpu) {
	// Locals
	QString qsScratch;
	QList<QStandardItem*> qlResult;
	QStringList qslTempList, qslTempList2;
	QVariant qvResult;

	// Clear lists
	qlResult.clear();
	qslTempList.clear();
	qslTempList2.clear();

	// Find TO's
	qlResult = qhTrees.value(qsInCpu)->findItems("OBJECT:", Qt::MatchRecursive |
Qt::MatchContains);

	// Put Elements into StringList
	for(int i = 0; i < qlResult.size(); ++i) {
		qsScratch = qlResult.at(i)->text();
		qslTempList << qsScratch;
	}

	// Remove duplicates
	qslTempList.removeDuplicates();

	// Shape strings
	for(int i = 0; i < qslTempList.size(); ++i) {
		qsScratch = qslTempList.at(i);
		qsScratch = qsScratch.section('(', 1, 1);
		qsScratch = qsScratch.remove(")");
		qslTempList2 << qsScratch;
	}

	// Get size
	qvResult = qslTempList2.size();

	// Copy temp list to result list
	qslResults << qslTempList2;

	// insert size of the answer at the head of the list itself
	qslResults.prepend(qvResult.toString());
}

//! This Function implements the getPlantTos-Command (cmd nbr. :3).
    /*!
      \param a QString holding the cpu the command is for.
      \return void
    */
void qcProject::cmdGetPlantTos(QString qsInCpu) {
	// Locals
	QString qsScratch;
	QList<QStandardItem*> qlResult;
	QStandardItem* qsiResult;
	QStringList qslTempList;
	QVariant qvResult;

	// Clear lists
	qlResult.clear();
	qslTempList.clear();

	// Find plant TO's
```

```
	qlResult = qhTrees.value(qsInCpu)->findItems("OBJECT:", Qt::MatchRecursive |
Qt::MatchContains );

	// Get Plant Objects
	if(!qlResult.isEmpty()) {
		for(int i = 0; i < qlResult.size(); ++i) {
			// Assemble DMS-Name
			// At first get Parent of the item at i
			qsiResult = qlResult.at(i)->parent();

			// Check there is a parent
			if(qsiResult != 0) {
				//
				while(qsiResult->text() != qsInCpu) {
					qsScratch.prepend(qsiResult->text() % ":");
					qsiResult = qsiResult->parent();
				}
			}
			// Move DMS-Name into string list
			qslTempList << qsScratch;

			// Clear string
			qsScratch.clear();
		}
	}
	else {
		// Debug output
		qDebug() << "Getting Plant To's failed";

		// Return value
		return;
	}

	// Get size
	qvResult = qslTempList.size();

	// Copy temp list to result list
	qslResults << qslTempList;

	// insert size of the answer at the head of the list itself
	qslResults.prepend(qvResult.toString());

	// Return value
	return;
}

//! This Function implements the getInstancesByType-Command (cmd nbr. :4).
    /*!
      \param a QString holding the cpu the command is for.
      \return void
    */
void qcProject::cmdGetInstancesByType(QString qsInCpu) {
	// Locals
	QString qsScratch;
	QList<QStandardItem*> qlResult;
	QStandardItem* qsiResult;
	QStringList qslTempList;
	QVariant qvResult;

	// Clear lists
	qlResult.clear();
	qslTempList.clear();
```

```
	// Find plant TO's
	qlResult = qhTrees.value(qsInCpu)->findItems("OBJECT:", Qt::MatchRecursive |
Qt::MatchContains );

	// Get Plant Objects
	if(!qlResult.isEmpty()) {
		for(int i = 0; i < qlResult.size(); ++i) {
			// Get TO Typ from string
			qsScratch = qlResult.at(i)->text();
			qsScratch = qsScratch.section('(', 1, 1);
			qsScratch = qsScratch.remove(")");

			// Assemble DMS-Name
			// At first get Parent of the item at i
			qsiResult = qlResult.at(i)->parent();

			// Check there is a parent
			if(qsiResult != 0) {
				//
				while(qsiResult->text() != qsInCpu) {
					qsScratch.prepend(qsiResult->text() % ":");
					qsiResult = qsiResult->parent();
				}
			}
			// Move DMS-Name into string list
			qslTempList << qsScratch;

			// Clear string
			qsScratch.clear();
		}
	}
	else {
		// Debug output
		qDebug() << "Getting Plant To's failed";

		// Return value
		return;
	}

	// Get size
	qvResult = qslTempList.size();

	// Copy temp list to result list
	qslResults << qslTempList;

	// insert size of the answer at the head of the list itself
	qslResults.prepend(qvResult.toString());

	// Return value
	return;
}

//! This Function implements the getData-Command (cmd nbr. :5).
    /*!
      \param a QString holding the cpu the command is for.
      \return void
    */
void qcProject::cmdGetData(QString qsInCpu) {
	// Locals
	QString qsScratch, qsScratch2;
	QList<QStandardItem*> qlResult;
	QStandardItem* qsiResult;
	QStringList qslTempList;
```

```
	QVariant qvResult;
	int iRowCount = 0;

	// Clear lists
	qlResult.clear();
	qslTempList.clear();

	// Find addresses
	qlResult = qhTrees.value(qsInCpu)->findItems("Address", Qt::MatchRecursive |
Qt::MatchContains);

	// Get adresses
	if(!qlResult.isEmpty()) {
		for(int i = 0; i < qlResult.size(); ++i) {
			// Get TO Typ from string
			qsScratch = qlResult.at(i)->text();

			// Assemble DMS-Name
			// At first get Parent of the item at i
			qsiResult = qlResult.at(i)->parent();

			// Check there is a parent
			if(qsiResult != 0) {
				//
				while(qsiResult->text() != qsInCpu) {
					qsScratch.prepend(qsiResult->text() % ":");
					qsiResult = qsiResult->parent();
				}
			}
			// Remove unneccesary things
			qsScratch.remove(qsScratch.section(":", -2, -1));
			qsScratch2 = qsScratch.section(":", -4, -4);
			qsScratch.remove(qsScratch2);
			qsScratch.remove(":PLC:");
			qsScratch.remove(qsInCpu);
			qsScratch.append(qsScratch2);

			// Move DMS-Name into string list
			if(!qsScratch.section(":", -2, -2).contains("D",
Qt::CaseInsensitive)) {
				qslTempList << qsScratch;
			}

			// Clear string
			qsScratch.clear();
		}
	}
	// Get size
	qvResult = qslTempList.size();

	// Copy temp list to result list
	qslResults << qslTempList;

	// insert size of the answer at the head of the list itself
	qslResults.prepend(qvResult.toString());

	// Return value
	return;
}

	//! This Function implements the getComment-Command (cmd nbr. :6).
    /*!
      \param a QString holding the cpu the command is for.
```

```
      \return void
    */
void qcProject::cmdGetComment(QString qsInCpu) {
       // Locals
       QString qsScratch;
       QList<QStandardItem*> qlResult;
       QStandardItem* qsiResult;
       QStringList qslTempList;
       QVariant qvResult;
       int iRowCount = 0;

       // Clear lists
       qlResult.clear();
       qslTempList.clear();

       // Find comments
       qlResult = qhTrees.value(qsInCpu)->findItems("NAME", Qt::MatchRecursive |
Qt::MatchStartsWith);

       // Get adresses
       if(!qlResult.isEmpty()) {
              for(int i = 0; i < qlResult.size(); ++i) {
                     // Get TO Typ from string
                     qsScratch = qlResult.at(i)->text();

                     // Assemble DMS-Name
                     // At first get Parent of the item at i
                     qsiResult = qlResult.at(i)->parent();

                     // Check there is a parent
                     if(qsiResult != 0) {
                            //
                            while(qsiResult->text() != qsInCpu) {
                                   qsScratch.prepend(qsiResult->text() % ":");
                                   qsiResult = qsiResult->parent();
                            }
                     }

                     // Move DMS-Name into string list
                     qslTempList << qsScratch;

                     // Clear string
                     qsScratch.clear();
              }
       }
       // Get size
       qvResult = qslTempList.size();

       // Copy temp list to result list
       qslResults << qslTempList;

       // insert size of the answer at the head of the list itself
       qslResults.prepend(qvResult.toString());

       // Return value
       return;
}

//! This Function implements the getAttribComment-Command (cmd nbr. :7).
    /*!
      \param a QString holding the cpu the command is for.
      \return void
    */
```

```
void qcProject::cmdGetAttribComment(QString qsInCpu) {
	// Locals
	QString qsScratch;
	QList<QStandardItem*> qlResult;
	QStandardItem* qsiResult;
	QStringList qslTempList1, qslTempList2;
	QVariant qvResult;
	int iRowCount = 0;

	// Clear lists
	qlResult.clear();
	qslTempList1.clear();

	// Find addresses
	qlResult = qhTrees.value(qsInCpu)->findItems("Address", Qt::MatchRecursive |
Qt::MatchContains);

	// Get adresses
	if(!qlResult.isEmpty()) {
		for(int i = 0; i < qlResult.size(); ++i) {
			// Get TO Typ from string
			qsScratch = qlResult.at(i)->text();

			// Assemble DMS-Name
			// At first get Parent of the item at i
			qsiResult = qlResult.at(i)->parent();

			// Check there is a parent
			if(qsiResult != 0) {
				//
				while(qsiResult->text() != qsInCpu) {
					qsScratch.prepend(qsiResult->text() % ":");
					qsiResult = qsiResult->parent();
				}
			}
			// Remove unneccesary things
			qsScratch.remove(qsScratch.section(":", -2, -1));
			qsScratch.remove(qsScratch.section(":", -4, -4));
			qsScratch.remove(":PLC:");
			qsScratch.remove(qsInCpu);

			// Move DMS-Name into string list
			if(!qsScratch.section(":", -2, -2).contains("D",
Qt::CaseInsensitive)) {
				// Remove unneccesary things
				qsScratch.remove(qsScratch.section(":", -2, -2));
				qsScratch.remove("::");

				// Move DMS-Name into string list
				qsScratch.append(":");
				qslTempList1 << qsScratch;
			}

			// Clear string
			qsScratch.clear();
		}
	}

	// Clear lists
	qlResult.clear();
	qslTempList2.clear();

	// Find addresses
```

```
	qlResult = qhTrees.value(qsInCpu)->findItems("Comment", Qt::MatchRecursive |
Qt::MatchStartsWith);

	// Get adresses
	if(!qlResult.isEmpty()) {
		for(int i = 0; i < qlResult.size(); ++i) {
			// Get TO Typ from string
			qsScratch = qlResult.at(i)->text();

			// Assemble DMS-Name
			// At first get Parent of the item at i
			qsiResult = qlResult.at(i)->parent();

			// Check there is a parent
			if(qsiResult != 0) {
				//
				while(qsiResult->text() != qsInCpu) {
					qsScratch.prepend(qsiResult->text() % ":");
					qsiResult = qsiResult->parent();
				}
			}

			// Check if templist1 contains the string
			QString qsScratch2 = qsScratch.section(":",-3 , -1);
			QString qsScratch3 = qsScratch;
			qsScratch3 = qsScratch3.remove(qsScratch2);

			for(int i = 0; i < qslTempList1.size(); ++i) {
				if(qslTempList1.at(i).contains(qsScratch3)) {
					// Remove unneccesary things
					qsScratch.remove(qsScratch.section(":", -3, -2));
					qsScratch.replace(":: ", ": ");

					// Move DMS-Name into string list if so
					qslTempList2 << qsScratch;

					// Clear string
					qsScratch.clear();
					break;
				}
			}

			// Clear string
			qsScratch.clear();
		}
	}
	// Get size
	qvResult = qslTempList2.size();

	// Copy temp list to result list
	qslResults << qslTempList2;

	// insert size of the answer at the head of the list itself
	qslResults.prepend(qvResult.toString());

	// Return value
	return;
}

//! This Function implements the getProjectPath-Command (cmd nbr. :8).
    /*!
      \param - none
      \return void
```

```
    */
void qcProject::cmdGetProjectPath() {
	// Locals
	QVariant qvResult;
	QStringList qslTempList;

	// Get all Cpu's out of the qHash
	qslTempList << getProjectPath();

	// Get CPU count
	qvResult = qslTempList.size();

	// Copy temp list to result list
	qslResults << qslTempList;

	// insert size of the answer at the head of the list itself
	qslResults.prepend(qvResult.toString());

	// Return value
	return;
}

//! This Function implements the getDataForTree-Command (cmd nbr.: 9).
    /*!
      \param - none
      \return void
    */
void qcProject::cmdgetDataForTree(QString qsInCpu) {
	// Locals
	QString qsScratch, qsScratch2, qsKeys, qsValues, qsToDmsName, qsToDmsName2,
qsDatapointValue;
	QList<QStandardItem*> qlResult;
	QStandardItem* qsiResult;
	QStandardItem* qsiChild;
	QStandardItem* qsiChild2;
	QStringList qslTempList, qslAttributesPlc, qslHashKeys, qslKeys;
	QVariant qvResult;
	QSet<QString> qsetParData, qsetParIn, qsetObject, qsetName;
	int iRowCount = 0, iRowCount2 = 0, iHashSize = 0, iScratch = 0;

	// Clear lists
	qlResult.clear();
	qslTempList.clear();

	// Clear Sets
	qsetParData.clear();
	qsetParIn.clear();

	// Fill attributes lists
	qslAttributesPlc << "Address" << "Type" << "PLC_Hi" << "PLC_Lo" << "Unit_Hi" <<
"Unit_Lo" << "DBIndex" << "BIT" << "FLT" << "DWU" << "Comment";

	// Find plant TO's address
	qlResult = qhTrees.value(qsInCpu)->findItems("Address", Qt::MatchRecursive |
Qt::MatchContains);

	// Get Plant Objects
	if(!qlResult.isEmpty()) {
		for(int i = 0; i < qlResult.size(); ++i) {
			// Assemble DMS-Name of the TO (WB044A:H09:MT:500)
			// At first get Parent of the item at i
			qsiResult = qlResult.at(i)->parent()->parent()->parent();
```

```
// Check if there is a parent
if(qsiResult != 0) {
	// Remove unnecessary things
	qsScratch2 = qsiResult->text();
	qsScratch2 = qsScratch2.section("(", 0, 0);
	qsScratch2 = qsScratch2.remove(": ");
	qsToDmsName = qsScratch2;

	// Get Next Parent
	qsiResult = qsiResult->parent();
	qsScratch2 = qsiResult->text();

	// Assemble DMS-Name
	while(qsiResult->parent()) {
		qsToDmsName.prepend(qsScratch2 % ":");
		qsiResult = qsiResult->parent();
		qsScratch2 = qsiResult->text();
	}
}

// At first get Parent of the item at i (WB044A:H09:MT:500:Eing)
qsiResult = qlResult.at(i)->parent()->parent();

// Check if there is a parent
if(qsiResult != 0) {
	// Remove unnecessary things
	qsScratch2 = qsiResult->text();
	qsScratch2 = qsScratch2.section("(", 0, 0);
	qsScratch2 = qsScratch2.remove(": ");
	qsToDmsName2 = qsScratch2;

	// Get Next Parent
	qsiResult = qsiResult->parent();
	qsScratch2 = qsiResult->text();

	// Assemble DMS-Name
	while(qsiResult->parent()) {
		qsToDmsName2.prepend(qsScratch2 % ":");
		qsiResult = qsiResult->parent();
		qsScratch2 = qsiResult->text();
	}
}

// At first get Parent of the item at i (WB044A:H09:MT:500:Eing)
qsiResult = qlResult.at(i)->parent()->parent();

// Check if there are children and get them
if(qsiResult != 0) {
	// Check if there are children
	if(qsiResult->hasChildren()) {

		// Create a MultiHash
		qhDataHash = new QHash<QString, QString>();
		qlHashes << qhDataHash;

		// Count rows
		iRowCount = qsiResult->rowCount();

		// Get Data
		for(int y = 0; y < iRowCount; ++y) {
			if(qsiResult->child(y) != 0) {
				// Get Child
				qsiChild = qsiResult->child(y);
```

```
0);

1, 1);

0);

qsValues);

>rowCount();

++x) {

0) {

qsiChild->child(x);

>text();

qsKeys.section("(", 1, 1);

qsKeys.section("(", 0, 0);

");

attribute is needed

to data-hash
```

```
// Get key-text
qsScratch2 = qsiChild->text();
qsScratch2 = qsScratch2.section("(", 0,

qsScratch2.remove(": ");

// Check which parameter it is
if(qsScratch2 == "Comment") {
	// Get key- and value-text
	qsKeys = qsiChild->text();

	// Get value-text
	qsValues = qsKeys.section("(",

	qsValues.remove(")");

	// Get key-text
	qsKeys = qsKeys.section("(", 0,

	qsKeys.remove(": ");

	// Move data to data-hash
	qhDataHash->insert(qsKeys,

}

else if(qsScratch2 == "PLC") {
	// Count rows
	iRowCount2 = qsiChild-


	// Get Data
	for(int x = 0; x < iRowCount2;

		if(qsiChild->child(x) !=

			// Get Child
			qsiChild2 =

			// Get key-text
			qsKeys = qsiChild2-

			// Get value-text
			qsValues =

			// Get key-text
			qsKeys =

			qsKeys.remove(":

			// check if the

				// Move data
```

```
>insert(qsKeys, qsValues);
                                                        qhDataHash-
                                                                }
                                                        }
                                                }
                                        }

                                        else if(qsScratch2 == "PAR_IN") {
                                                // Get key- and value-text
                                                qsKeys = qsiChild->text();

                                                // Get value-text
                                                qsValues = qsKeys.section("(",
1, 1);
                                                qsValues.remove(")");

                                                // Get key-text
                                                qsKeys = qsKeys.section("(", 0,
0);
                                                qsKeys.remove(": ");

                                                // Put DMS-Name2 into set
                                                qsetParIn.insert(qsToDmsName2);

                                                // Move data to data-hash
                                                qhDataHash->insert(qsKeys,
qsValues);
                                        }

                                        else if(qsScratch2 == "PAR_OUT") {
                                                // Get key- and value-text
                                                qsKeys = qsiChild->text();

                                                // Get value-text
                                                qsValues = qsKeys.section("(",
1, 1);
                                                qsValues.remove(")");

                                                // Get key-text
                                                qsKeys = qsKeys.section("(", 0,
0);
                                                qsKeys.remove(": ");

                                                // Move data to data-hash
                                                qhDataHash->insert(qsKeys,
qsValues);
                                        }

                                        else if(qsScratch2 == "PAR_DATA") {
                                                // Get key- and value-text
                                                qsKeys = qsiChild->text();

                                                // Get value-text
                                                qsValues = qsKeys.section("(",
1, 1);
                                                qsValues.remove(")");

                                                // Get key-text
                                                qsKeys = qsKeys.section("(", 0,
0);
                                                qsKeys.remove(": ");

                                                // Put DMS-Name2 into set
```

```
	qsetParData.insert(qsToDmsName2);

                                                // Move data to data-hash
                                                qhDataHash->insert(qsKeys,
qsValues);
                                        }

                                        // Get Datapoint values
                                        qsDatapointValue = qsiResult->text();
                                        qsDatapointValue =
qsDatapointValue.section("(", 1, 1);
                                        qsDatapointValue.remove(")");
                                        qsDatapointValue =
qsDatapointValue.section(".", 0, 0);

                                        // Move data to data-hash
                                        qhDataHash->insert("DatapointValue",
qsDatapointValue);
                                    }
                                }

                                // Move data-hash to instance-hash
                                qmhInstanceHash->insert(qsToDmsName2, qhDataHash);
                            }
                            else {
                                // Show messagebox
                                QMessageBox::warning(0, "pTool/cmd: 9", "No Data
found");

                                // Return value
                                return;
                            }
                        }

                        // Clear string
                        qsToDmsName2.clear();
                        qsToDmsName.clear();
                    }

                    // Clear Result-List
                    qlResult.clear();

	}
	else {
                    // Debug output
                    qDebug() << "Getting Plant To's failed";

                    // Return value
                    return;
	}

	// Get additional PAR_DATA
	qlResult = qhTrees.value(qsInCpu)->findItems("PAR_DATA", Qt::MatchRecursive |
Qt::MatchContains);

	// Get Plant Objects
	if(!qlResult.isEmpty()) {
                    for(int i = 0; i < qlResult.size(); ++i) {
                        // Assemble DMS-Name of the TO (WB044A:H09:MT:500)
                        // At first get Parent of the item at i
                        qsiResult = qlResult.at(i)->parent()->parent();
```

```
// Check if there is a parent
if(qsiResult != 0) {
	// Remove unnecessary things
	qsScratch2 = qsiResult->text();
	qsScratch2 = qsScratch2.section("(", 0, 0);
	qsScratch2 = qsScratch2.remove(": ");
	qsToDmsName = qsScratch2;

	// Get Next Parent
	qsiResult = qsiResult->parent();
	qsScratch2 = qsiResult->text();

	// Assemble DMS-Name
	while(qsiResult->parent()) {
		qsToDmsName.prepend(qsScratch2 % ":");
		qsiResult = qsiResult->parent();
		qsScratch2 = qsiResult->text();
	}
}

// At first get Parent of the item at i (WB044A:H09:MT:500:Eing)
qsiResult = qlResult.at(i)->parent();

// Check if there is a parent
if(qsiResult != 0) {
	// Remove unnecessary things
	qsScratch2 = qsiResult->text();
	qsScratch2 = qsScratch2.section("(", 0, 0);
	qsScratch2 = qsScratch2.remove(": ");
	qsToDmsName2 = qsScratch2;

	// Get Next Parent
	qsiResult = qsiResult->parent();
	qsScratch2 = qsiResult->text();

	// Assemble DMS-Name
	while(qsiResult->parent()) {
		qsToDmsName2.prepend(qsScratch2 % ":");
		qsiResult = qsiResult->parent();
		qsScratch2 = qsiResult->text();
	}
}

// At first get Parent of the item at i (WB044A:H09:MT:500:Eing)
qsiResult = qlResult.at(i);

// Check if there are children and get them
if(qsiResult != 0) {
	// Get Data
	if(!qsetParData.contains(qsToDmsName2)) {
		// Create a MultiHash
		qhDataHash = new QHash<QString, QString>();
		qlHashes << qhDataHash;

		// Get key-text
		qsScratch2 = qsiResult->text();
		qsScratch2 = qsScratch2.section("(", 0, 0);
		qsScratch2.remove(": ");

		// Check which parameter it is
		if(qsScratch2 == "PAR_DATA") {
			// Get key- and value-text
			qsKeys = qsiResult->text();
```

```
                                        // Get value-text
                                        qsValues = qsKeys.section("(", 1, 1);
                                        qsValues.remove(")");

                                        // Get key-text
                                        qsKeys = qsKeys.section("(", 0, 0);
                                        qsKeys.remove(": ");

                                        // Put DMS-Name2 into set
                                        qsetParData.insert(qsToDmsName2);

                                        // Move data to data-hash
                                        qhDataHash->insert(qsKeys, qsValues);

                                        // Move data-hash to instance-hash
                                        qmhInstanceHash->insert(qsToDmsName2,
qhDataHash);
                                }
                        }
                }

                // Clear string
                qsToDmsName2.clear();
                qsToDmsName.clear();
        }

        // Clear Result-List
        qlResult.clear();

    }
    else {
        // Debug output
        qDebug() << "Getting additional PAR_DATA failed";

        // Return value
        return;
    }


    // Get additional PAR_IN
    qlResult = qhTrees.value(qsInCpu)->findItems("PAR_IN", Qt::MatchRecursive |
Qt::MatchContains);

    // Get Plant Objects
    if(!qlResult.isEmpty()) {
        for(int i = 0; i < qlResult.size(); ++i) {
            // Assemble DMS-Name of the TO (WB044A:H09:MT:500)
            // At first get Parent of the item at i
            qsiResult = qlResult.at(i)->parent()->parent();

            // Check if there is a parent
            if(qsiResult != 0) {
                // Remove unnecessary things
                qsScratch2 = qsiResult->text();
                qsScratch2 = qsScratch2.section("(", 0, 0);
                qsScratch2 = qsScratch2.remove(": ");
                qsToDmsName = qsScratch2;

                // Get Next Parent
                qsiResult = qsiResult->parent();
                qsScratch2 = qsiResult->text();
```

```
		// Assemble DMS-Name
		while(qsiResult->parent()) {
			qsToDmsName.prepend(qsScratch2 % ":");
			qsiResult = qsiResult->parent();
			qsScratch2 = qsiResult->text();
		}
	}

	// At first get Parent of the item at i (WB044A:H09:MT:500:Eing)
	qsiResult = qlResult.at(i)->parent();

	// Check if there is a parent
	if(qsiResult != 0) {
		// Remove unnecessary things
		qsScratch2 = qsiResult->text();
		qsScratch2 = qsScratch2.section("(", 0, 0);
		qsScratch2 = qsScratch2.remove(": ");
		qsToDmsName2 = qsScratch2;

		// Get Next Parent
		qsiResult = qsiResult->parent();
		qsScratch2 = qsiResult->text();

		// Assemble DMS-Name
		while(qsiResult->parent()) {
			qsToDmsName2.prepend(qsScratch2 % ":");
			qsiResult = qsiResult->parent();
			qsScratch2 = qsiResult->text();
		}
	}

	// At first get Parent of the item at i (WB044A:H09:MT:500:Eing)
	qsiResult = qlResult.at(i);

	// Check if there are children and get them
	if(qsiResult != 0) {
		// Get Data
		if(!qsetParIn.contains(qsToDmsName2)) {
			// Create a MultiHash
			qhDataHash = new QHash<QString, QString>();
			qlHashes << qhDataHash;

			// Get key-text
			qsScratch2 = qsiResult->text();
			qsScratch2 = qsScratch2.section("(", 0, 0);
			qsScratch2.remove(": ");

			// Check which parameter it is
			if(qsScratch2 == "PAR_IN") {
				// Get key- and value-text
				qsKeys = qsiResult->text();

				// Get value-text
				qsValues = qsKeys.section("(", 1, 1);
				qsValues.remove(")");

				// Get key-text
				qsKeys = qsKeys.section("(", 0, 0);
				qsKeys.remove(": ");

				// Put DMS-Name2 into set
				qsetParIn.insert(qsToDmsName2);
```

```
                                        // Move data to data-hash
                                        qhDataHash->insert(qsKeys, qsValues);

                                        // Move data-hash to instance-hash
                                        qmhInstanceHash->insert(qsToDmsName2,
qhDataHash);
                                }
                        }
                }

                // Clear string
                qsToDmsName2.clear();
                qsToDmsName.clear();
        }

        // Clear Result-List
        qlResult.clear();

    }
    else {
        // Debug output
        qDebug() << "Getting additional PAR_IN failed";

        // Return value
        return;
    }

    // Get additional OBJECT and NAME
    qlResult = qhTrees.value(qsInCpu)->findItems("OBJECT", Qt::MatchRecursive |
Qt::MatchContains);

    // Get Plant Objects
    if(!qlResult.isEmpty()) {
        for(int i = 0; i < qlResult.size(); ++i) {
            // At first get Parent of the item at i (WB044A:H09:MT:500)
            qsiResult = qlResult.at(i)->parent();

            // Check if there is a parent
            if(qsiResult != 0) {
                // Remove unnecessary things
                qsScratch2 = qsiResult->text();
                qsScratch2 = qsScratch2.section("(", 0, 0);
                qsScratch2 = qsScratch2.remove(": ");
                qsToDmsName = qsScratch2;

                // Get Next Parent
                qsiResult = qsiResult->parent();
                qsScratch2 = qsiResult->text();

                // Assemble DMS-Name
                while(qsiResult->parent()) {
                    qsToDmsName.prepend(qsScratch2 % ":");
                    qsiResult = qsiResult->parent();
                    qsScratch2 = qsiResult->text();
                }
            }

            // At first get Parent of the item at i (WB044A:H09:MT:500:OBJECT)
            qsiResult = qlResult.at(i)->parent();

            // Check if there are children
            if(qsiResult->hasChildren()) {
```

```
                        // Count rows
                        iRowCount = qsiResult->rowCount();

                        // Create a Hash
                        qhDataHash = new QHash<QString, QString>();
                        qlHashes << qhDataHash;

                        // Clear QSet
                        qsetObject.clear();

                        // Get Data
                        for(int y = 0; y < iRowCount; ++y) {
                               if(qsiResult->child(y) != 0) {
                                      // Get Child
                                      qsiChild = qsiResult->child(y);
                                      if(!qsiChild->hasChildren()) {
                                             // Get key-text
                                             qsScratch2 = qsiChild->text();
                                             qsScratch2 = qsScratch2.section("(", 0,
0);
                                             qsScratch2.remove(": ");

                                             // assembly string
                                             qsScratch = qsToDmsName % ":" %
qsScratch2;

                                             // Check which parameter it is
                                             if(qsScratch2 == "OBJECT" || qsScratch2
== "NAME") {
                                                    // Get key- and value-text
                                                    qsKeys = qsiChild->text();

                                                    // Get value-text
                                                    qsValues = qsKeys.section("(",
1, 1);
                                                    qsValues.remove(")");

                                                    // Get key-text
                                                    qsKeys = qsKeys.section("(", 0,
0);
                                                    qsKeys.remove(": ");

                                                    // Put DMS-Name into set
                                                    qsetObject.insert(qsScratch2);

                                                    // Move data to data-hash
                                                    qhDataHash->insert(qsKeys,
qsValues);

                                                    if(qsetObject.count() == 2) {
                                                           // Check whether key is
allready in hash
                                                           if(qmhInstanceHash-
>contains(qsToDmsName)) {
                                                                  // Delete key if it
is in hash
                                                                  qmhInstanceHash-
>remove(qsToDmsName);
                                                                  // Move data-hash
to instance-hash
                                                                  qmhInstanceHash-
>insert(qsToDmsName, qhDataHash);
                                                           }
```

```
                                                                        else {
                                                                                // Move data-hash
to instance-hash
                                                                                qmhInstanceHash-
>insert(qsToDmsName, qhDataHash);
                                                                        }
                                                                }
                                                        }

                                                }
                                        }
                                }
                        }

                        // Clear string
                        qsToDmsName.clear();
                }
                // Clear Keys ans Values
                qsKeys.clear();
                qsValues.clear();

                // Clear Result-List
                qlResult.clear();
        }
        else {
                // Debug output
                qDebug() << "Getting OBJECT failed";

                // Return value
                return;
        }

        // Get size of hash QList<QString>
        iHashSize = qmhInstanceHash->size();
        qslHashKeys = qmhInstanceHash->keys();

        // Serialize
        for(int i = 0; i < qslHashKeys.size(); ++i) {
                // Insert Key into list
                qsScratch = "Key: " % qslHashKeys.at(i);
                qslTempList << qsScratch;

                // Insert data into list after key
                QList<QHash<QString, QString>*> qlData;
                qlData.clear();
                qlData = qmhInstanceHash->values(qslHashKeys.at(i));

                for(int x = 0; x < qlData.size(); ++x) {
                        if(!qlData.isEmpty()) {
                                // Get keys
                                qslKeys = qlData.at(x)->keys();

                                // get values
                                while(!qslKeys.isEmpty()) {
                                        // get keys
                                        qsKeys = qslKeys.takeFirst();
                                        qslTempList << qsKeys;

                                        // get value
                                        qsValues = qlData.at(x)->value(qsKeys);
                                        qslTempList << qsValues;
                                }
                        }
```

```
			}
		}

		// Clear Hash
		qmhInstanceHash->clear();

		while(!qlHashes.isEmpty()) {
			// Delete Hashes
			delete qlHashes.takeFirst();
		}

		// Get size of qslTempList
		qvResult = qslTempList.size();

		// Copy temp list to result list
		qslResults << qslTempList;

		// insert size of the answer at the head of the list itself
		qslResults.prepend(qvResult.toString());

		// Return value
		return;
}
// ************** End of Command Functions ****************
```

Appendix D VII Source code standardtreemodel.h

```
#ifndef STANDARDTREEMODEL_H
#define STANDARDTREEMODEL_H

#include <QStandardItemModel>
#include <QtGui>                                    // QT header

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

class StandardTreeModel : public QStandardItemModel {
       Q_OBJECT

public:
       StandardTreeModel(QObject *parent);
       ~StandardTreeModel();

       bool bLoadData(QString qsInPath);

private:
       void initialize();

       QList<QStandardItem*> qlTextItems;
       QList<QStandardItem*> qlDataItems;
       QList<QStandardItem*> qlParentItems;

       QStandardItem *qsiTextItem;
       QStandardItem *qsiDataItem;
       QStandardItem *qsiParentItem;
};

#endif // STANDARDTREEMODEL_H
```

Appendix D VIII Source code standardtreemodel.cpp

```
#include "standardtreemodel.h"
#include "stdafx.h"                        // Precompiled headers

StandardTreeModel::StandardTreeModel(QObject *parent): QStandardItemModel(parent) {
	// Inizialize tree
	initialize();
}

StandardTreeModel::~StandardTreeModel() {

}

void StandardTreeModel::initialize() {
	// Set Header Labels
	setHorizontalHeaderLabels(QStringList() << tr("Tree"));

	// Set alignments
	for (int column = 1; column < columnCount(); ++column) {
	 horizontalHeaderItem(column)-
>setTextAlignment(Qt::AlignVCenter|Qt::AlignRight);
	}
}

bool StandardTreeModel::bLoadData(QString qsInPath) {
	// Locals
	QFile qfInputFile;
	QTextStream qtsInput;
	QString qsLine, qsOnlyTextItems, qsDataItem, qsScratch;
	QStringList qslTextItems, qslDataItems;
	bool qsRetValue;
	int iSemicolonCount;

	if(qsInPath.isEmpty()) {
		QMessageBox::warning(0, "treeTest", "Bitte geben Sie ein Quellen-File
an");
	}
	else {
		// Create Items
		qsiParentItem = new QStandardItem();

		// Set Filename
		qfInputFile.setFileName(qsInPath);
		qtsInput.setDevice(&qfInputFile);

		if(!qfInputFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "treeTest", "Kein File im Ordner
gefunden");
			// Set Return value
			qsRetValue = false;
		}
		else {
			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfInputFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Scanning file: " % qsInPath);
			qpdProgress->setValue(0);
			qpdProgress->show();

			// Clear tree
```

```
                        clear();

                        // Initialize tree
                        initialize();

                        // find Typ
                        while (!qtsInput.atEnd()) {
                               // Search String
                               qsLine = qtsInput.readLine();

                               // Set progress bar
                               qpdProgress->setValue(qfInputFile.pos());

                               // Extract Data
                               iSemicolonCount = qsLine.count(";");

                               // Get Items
                               qsOnlyTextItems = qsLine.section(';', 0, 0);
                               qsDataItem = qsLine.section(";", 1, iSemicolonCount);

                               // Extract CPU or System
                               qslTextItems = qsOnlyTextItems.split(":");
                               qslDataItems = qsDataItem.split(";");

                               // Get Data
                               for(int i = 0; i < qslDataItems.size(); ++i) {
                                      // Create data item
                                      qsiDataItem = new QStandardItem(qslDataItems.at(i));

                                      qlDataItems.append(qsiDataItem);
                               }

                               // Create Tree
                               qlTextItems.clear();
                               qlParentItems.clear();
                               qlParentItems.append(invisibleRootItem());
                               for(int i = 0; i < qslTextItems.size(); ++i) {
                                      // Check if Item exists
                                      qsScratch = qslTextItems.at(i);
                                      qsiTextItem = new QStandardItem(qslTextItems.at(i));

                                      if(!qlParentItems.isEmpty()) {
                                             if(qlParentItems.at(0) != qsiTextItem-
>parent()) {
                                                    // Check parent text
                                                    QString qsTest01 = qlParentItems.at(0)-
>text();
                                                    int iRowCount = qlParentItems.at(0)-
>rowCount();
                                                    bool bMatch = false;

                                                    if(iRowCount > 0) {

                                                           for(int y = 0; y < iRowCount;
++y){
                                                                  QString qsTest02 =
qlParentItems.at(0)->child(y, 0)->text();
                                                                  QString      qsTest03 =
qslTextItems.at(i);

                                                                  if(qsTest02 == qsTest03)
{
                                                                         bMatch = true;
```

```
                                                                        qsiTextItem =
qlParentItems.at(0)->child(y, 0);

        qlParentItems.clear();

        qlParentItems.append(qsiTextItem);
                                                                                }
                                                                        }
                                                                        if(!bMatch) {
                                                                                // Insert item
                                                                                qlParentItems.at(0)-
>appendRow(qsiTextItem);
                                                                                qlParentItems.clear();

        qlParentItems.append(qsiTextItem);
                                                                        }
                                                                        else {
                                                                                qlParentItems.clear();

        qlParentItems.append(qsiTextItem);
                                                                        }
                                                                }
                                                                else {
                                                                        // Insert item
                                                                        QString qsTest04 =
qlParentItems.at(0)->text();
                                                                        qlParentItems.at(0)-
>appendRow(qsiTextItem);
                                                                        qlParentItems.clear();

        qlParentItems.append(qsiTextItem);
                                                                }
                                                        }
                                                        else {
                                                                qlParentItems.clear();
                                                                qlParentItems.append(qsiTextItem);
                                                        }
                                                }
                                                else {
                                                        qlParentItems.clear();
                                                        qlParentItems.append(qsiTextItem);
                                                }

                                        }
                                        // Insert Data
                                        qlParentItems.at(0)->setText(qlParentItems.at(0)->text() %
": (" + qslDataItems.at(1) % ")");

                        }
                        // Delete objects
                        delete qpdProgress;

                        // Close file
                        qfInputFile.close();

                        // Sortieren
                        sort(0, Qt::AscendingOrder);
                }
        }
        // Return value
        return qsRetValue;
}
```

Appendix E Sources pLib

Appendix E I Source code pLib.h

```
#ifndef PLIB_H
#define PLIB_H

#include <QtGui>                          // QT header
#include "D:\Thoemus_Stuff\Projekte\C++\pTool\global\globals.h"

/****************** Defines ************************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ************************/


//!  This is the pLib Class, it is used to extract data from the ProMoS NT
configuration Files offline.
/*!
  This Library is designed to extract data from the promos.dms and the bmo.dms Files.
The access to this
  Files will be an offline one. The Library will be used to simplify the development
of ProMoS NT Tools for
  the daily use.
*/
class pLib {
/******************* Publics ***********************/
public:
	pLib();
	~pLib();

	// CPU-Functions
	QString getSingleCpu(QString qsFileName);
	QStringList getCpus(QString qsFileName);

	// Tree-Functions
	int readTree(QString qsInProjectPath, QString qsSearchString, QString
qsFileName, QFile &qfTree);

	// TO-Functions
	QStringList getTos(const struct struGeneratorData *struGenData, QString
qsOutputTo);
	QStringList getTos(QString qsFileName);
	QStringList getToDescription(QString qsFileName, QStringList qslDmsNames);
	QStringList getToInstances(QString qsFileName, QString qsInToName);
	QStringList getInstanceDescription(QString qsFileName, QStringList qslDmsNames);
	QStringList getAttributes(QString qsFileName, QString qsInToName);
	QStringList getAttributeDescriptions(QString qsFileName, QString qsInToName,
QStringList qslAttributes);

	// DMS Name Functions
	int searchDmsName(QString qsDmsName, QString qsFileName);
	QStringList findDmsNames(const struct struGeneratorData *struGenData,
QStringList qslAttributes);

	// Export Functions
	QString getToDmsData(QString qsFileName, QString qsToName);
	int writeToDmsDataToFile(QString qsFileName, QString qsOutputData);

	// Ressources Functions
	QString checkTyp(QString qsInDmsName, QString qsInFilename);
```

```
	QStringList checkTyp(QStringList qslInDmsNames, QString qsInFilename); // not
implementet yet


/****************** End Publics **********************/

/******************** Privates **********************/
private:

/****************** End Privates ********************/
};

#endif // PLIB_H
```

Appendix E II Source code pLib.cpp

```
/*****************************************************************************
 Implementation of the pLib DLL
 ------------------------------

 Date         : 25.01.2010
 File       : plib.cpp
 Author             : Thomas Gasser
 ©          : 2010 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's ProMoS NT Functions Library.
/*!
  This Library is designed to use with the SCADA-System ProMoS NT from MST
Systemtechnik AG.
  It provides functions to retrive Data from the promos.dms and the bmo.dms Files. It
is
  designed to work offline.
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "plib.h"                       // Project header
/****************** Includes ************************/

// Constructor / Destructor
//! This is the Constructor of the pLib Library.
    /*!
      At the moment the Constructor of the pLib Library is not used at all.
    */
pLib::pLib() {

}

//! This is the Destructor of the pLib Library.
    /*!
      At the moment the Destructor of the pLib Library is not used at all.
    */
pLib::~pLib() {

}

// p-Functions

//! This Function searches the given promos.dms File and returns the Cpu's used in it
in a List and lets the user chooses one.
    /*!
      \param qsFileName A QString which holds the Path and the Filename of the
promos.dms File.
      \return A QString with the choosen Cpu
    */
QString pLib::getSingleCpu(QString qsFileName) {
       // Locals
       QFile qfFile;
       QTextStream qtsInput;
       QStringList qslRetList;
       QString qsline, qsCpuName, qsSearchString;
```

```
	bool ok;

	// Pfad setzen
	if(qsFileName.isEmpty()) {
		QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
		qsCpuName = "";
	}
	else {
		// Set Filename
		qfFile.setFileName(qsFileName);
		qtsInput.setDevice(&qfFile);

		if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
			qsCpuName = "";
		}
		else {
			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Suche CPU's");
			qpdProgress->setValue(0);
			qpdProgress->show();

			// Set search string
			qsSearchString = "System:Prg:PCD:";

			// find CPU's
			while (!qtsInput.atEnd()) {
				// find entries
				qsline = qtsInput.readLine();
				if(qsline.section(':', 0, 0) == "System") {
					qpdProgress->setValue(qfFile.pos());
					if(qsline.contains(qsSearchString)) {
						if(qsline.section(':', 3, 3) != qsCpuName) {
							qslRetList << qsline.section(':', 3,
3);

							qsCpuName = qsline.section(':', 3, 3);
						}
					}
					// Check the Progress Dialog Cancel Button
					if (qpdProgress->wasCanceled()) {
						break;
					}
				}
				else {
					break;
				}
			}
			//Remove some entries because of the error messages in the system
branche of the promos.dms file
			qslRetList.removeDuplicates();
			qslRetList.removeAt(qslRetList.indexOf("Default"));
			qslRetList.sort();

			// Get Cpu name
			if(qslRetList.size() > 1) {
				qsCpuName = QInputDialog::getItem(0, "Cpu auswählen",
"Gefundene Cpu's", qslRetList, 0, false, &ok);
				if(qsCpuName.isEmpty()) {
```

```
                                        QMessageBox::warning(0, "pLib", "Bitte eine Cpu
auswählen");
                                }
                        }
                        else {
                                qsCpuName = qslRetList.at(0);
                        }
                        // Delete objects
                        delete qpdProgress;

                        // Close file
                        qfFile.close();
                }
        }
        // Return value
        return qsCpuName;
}

// p-Functions

//! This Function searches the given promos.dms File and returns the Cpu's used in it
in a List.
    /*!
      \param qsFileName A QString which holds the Path and the Filename of the
promos.dms File.
      \return A QStringList with the found Cpus
    */
QStringList pLib::getCpus(QString qsFileName) {
        // Locals
        QFile qfFile;
        QTextStream qtsInput;
        QStringList qslRetList;
        QString qsline, qsCpuName, qsSearchString;
        bool ok;

        // Pfad setzen
        if(qsFileName.isEmpty()) {
                QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
                qsCpuName = "";
        }
        else {
                // Set Filename
                qfFile.setFileName(qsFileName);
                qtsInput.setDevice(&qfFile);

                if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                        QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                        qsCpuName = "";
                }
                else {
                        // Setup Progress Dialog
                        QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
                        qpdProgress->setWindowModality(Qt::WindowModal);
                        qpdProgress->setLabelText("Suche CPU's");
                        qpdProgress->setValue(0);
                        qpdProgress->show();

                        // Set search string
                        qsSearchString = "System:Prg:PCD:";

                        // find CPU's
```

```
                        while (!qtsInput.atEnd()) {
                                // find entries
                                qsline = qtsInput.readLine();
                                if(qsline.section(':', 0, 0) == "System") {
                                        qpdProgress->setValue(qfFile.pos());
                                        if(qsline.contains(qsSearchString)) {
                                                if(qsline.section(':', 3, 3) != qsCpuName) {
                                                        qslRetList << qsline.section(':', 3,
3);
                                                        qsCpuName = qsline.section(':', 3, 3);
                                                }
                                        }
                                        // Check the Progress Dialog Cancel Button
                                        if (qpdProgress->wasCanceled()) {
                                                break;
                                        }
                                }
                                else {
                                        break;
                                }
                        }
                        //Remove some entries because of the error messages in the system
branche of the promos.dms file
                        qslRetList.removeDuplicates();
                        qslRetList.removeAt(qslRetList.indexOf("Default"));
                        qslRetList.sort();

                        // Delete objects
                        delete qpdProgress;

                        // Close file
                        qfFile.close();
                }
        }
        // Return value
        return qslRetList;
}

//! This Function searches the file given in the second parameter for the String given
in the first parameter.
    /*!
      \param qsDmsName A QString holding the string to be searched.
        \param qsFileName A QString holding the Path and the Filename to the File to
be searched.
      \return An Integer    0 := String found,
                                                -1 := File not found,
                                                -2 := Error opening the File,
                                                -3 := Searching was aborted,
                                                -4 := String not found
    */
int pLib::searchDmsName(QString qsDmsName, QString qsFileName) {
        // Locals
        int iRetValue;
        QFile qfFile;
        QTextStream qtsInput;
        QString qsline;

        // Set Filename
        qfFile.setFileName(qsFileName);
        if(qfFile.exists()) {

                // Set device fot the textstream
                qtsInput.setDevice(&qfFile);
```

```
		// Check file
		if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {

			iRetValue = -2;
		}
		// Search Dms Name
		else {
			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Suche Symbol... (" + qsDmsName + ")");
			qpdProgress->setValue(0);
			qpdProgress->show();

			while (!qtsInput.atEnd()) {
				// find entries
				qsline = qtsInput.readLine();
				qpdProgress->setValue(qfFile.pos());

				if(qsline.contains(qsDmsName)) {
					iRetValue = 0;
					break;
				}

				// Check the Progress Dialog Cancel Button
				if (qpdProgress->wasCanceled()) {
					iRetValue = -3;
					break;
				}

			}
			// Set return value
			iRetValue = -4;

			// Delete objects
			delete qpdProgress;

			// Close file
			qfFile.close();
		}
	}
	else {
		// No File found
		iRetValue = -1;
	}

	// Return value
	return iRetValue;
}


//! This Function copies the DMS-Data of the To given in the second parameter from the
bmo.dms File given in the first parameter.
    /*!
      \param qsFileName A QString holding the Path and the Filename to the bmo.dms
File.
        \param qsToName A QString holding the To whose DMS-Data should be copied.
      \return A QString holding the DMS-Data of the To searched for.
    */
QString pLib::getToDmsData(QString qsFileName, QString qsToName) {
	// Locals
```

```
	QFile qfInput;
	QTextStream qtsInput;
	QString qsScratch, qsLine, qsToDmsData;
	bool bFound = false;

	// Pfad setzen
	if(qsFileName.isEmpty()) {
		QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
		qsToDmsData.clear();
	}
	else {
		// Set Filename
		qfInput.setFileName(qsFileName);
		qtsInput.setDevice(&qfInput);

		if (!qfInput.open(QIODevice::ReadOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
			qsToDmsData.clear();
		}
		else {
			// Set Progress
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfInput.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Suche DMS-Daten...");
			qpdProgress->show();

			// Assemble searced String
			qsScratch = "BMO:" % qsToName;

			// Read the whole File into a QString
			while(!qtsInput.atEnd() && bFound == false) {
				qsLine = qtsInput.readLine();
				qpdProgress->setValue(qfInput.pos());

				while(qsLine.contains(qsScratch)) {

					qsToDmsData.append(qsLine % "\n");
					qsLine = qtsInput.readLine();
					qpdProgress->setValue(qfInput.pos());
					bFound = true;

					// Check the Progress Dialog Cancel Button
					if (qpdProgress->wasCanceled()) {
						break;
					}
				}
				// Check the Progress Dialog Cancel Button
				if (qpdProgress->wasCanceled()) {
					break;
				}
			}
			// Delete objects
			delete qpdProgress;
		}
	}
	// Close Files
	qfInput.close();

	// Return value
	return qsToDmsData;
```

```
}
//! This Function writes the DMS-Data given in the second parameter to the File given
in the first parameter.
    /*!
      \param qsFileName A QString holding the Path and the Filename to the output
File.
        \param qsOutputData A QString holding the DMS-Data to be written
      \return An Integer:   0 := File was written correctly,
                                                -1 := The output File could not be
opened or created
    */
int pLib::writeToDmsDataToFile(QString qsFileName, QString qsOutputData) {
       // Locals
       QFile qfOutput;
       QTextStream qtsOutput;
       int iRetValue;

       // Set Filename
       qfOutput.setFileName(qsFileName);

       // Create new file
       qtsOutput.setDevice(&qfOutput);
       if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
              QMessageBox::warning(0, "pLib", "Die Datei konnte geöffnet werden");
              iRetValue = -1;
       }
       else {
              // Write String to File
              qtsOutput << qsOutputData;
              iRetValue = 0;
       }
       // Close file
       qfOutput.close();

       // Return value
       return iRetValue;
}

//! This Function writes the DMS-Data given in the second parameter to the File given
in the first parameter.
    /*!
      \param *struGenData A Pointer to a struGeneratorData structure which holds the
needed Information.
        \param qslAttributes A QStringList holding To Attributes to be searched.
      \return A QStringList holding the DMS-Names of the To's matching one of the
given To Attributes
    */
QStringList pLib::findDmsNames(const struct struGeneratorData *struGenData,
QStringList qslAttributes) {
       // Locals
       QFile qfInputFile;
       QTextStream qtsInput;
       QString qsline, qsScratch, qsDmsName;
       QStringList qslRetList, qslTempList;

       // Find all DMS-Names with the given Attributes
       // Setup file and stream
       qfInputFile.setFileName(struGenData->qsInputFile);
       qtsInput.setDevice(&qfInputFile);

       // Check file
       if (!qfInputFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
```

```
			qslRetList.clear();
		}
		// Search Dms Name
		else {

			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("", "Abbrechen", 0,
qfInputFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Suche DMS-Namen...");
			qpdProgress->setValue(0);
			qpdProgress->show();

			while (!qtsInput.atEnd()) {
				// find entries
				qsline = qtsInput.readLine();
				qpdProgress->setValue(qfInputFile.pos());

				// Check CPU-Name
				if(!struGenData->bAdditionalConfiguration) {
					if(qsline.section(':', 0, 0) == struGenData->qsCpuName &&
qsline.contains(":PLC:Address;")) {
						// Check if line contains one of the given
attributes
						for(int i = 0; i < qslAttributes.size(); ++i) {
							if(qsline.contains(qslAttributes.at(i) +
":PLC:Address;" )) {
								qsScratch = qsline.section(":", 1,
(qsline.count(":") - 2));
								if(qsScratch != qsDmsName) {
									qsDmsName = qsScratch;
									qslTempList << qsScratch;

								}
							}
						}
					}
				}
				else {
					if(qsline.section(':', 0, 0) == struGenData->qsCpuName &&
qsline.section(':', 1, 1) == struGenData->qsAdditionalSeperator &&
qsline.contains(":PLC:Address;")) {
						// Check if line contains one of the given
attributes
						for(int i = 0; i < qslAttributes.size(); ++i) {
							if(qsline.contains(qslAttributes.at(i) +
":PLC:Address;" )) {
								qsScratch = qsline.section(":", 1,
(qsline.count(":") - 2));
								if(qsScratch != qsDmsName) {
									qsDmsName = qsScratch;
									qslTempList << qsScratch;

								}
							}
						}
					}
					if(struGenData->bCommonStuff) {
						if(qsline.section(':', 0, 0) == struGenData-
>qsCpuName && qsline.section(':', 1, 1) == struGenData->qsCommonStuff &&
qsline.contains(":PLC:Address;")) {
							// Check if line contains one of the given
attributes
```

```
                                        for(int i = 0; i < qslAttributes.size(); ++i)
{
                                                if(qsline.contains(qslAttributes.at(i)
+ ":PLC:Address;" )) {
                                                        qsScratch = qsline.section(":",
1, (qsline.count(":") - 2));
                                                        if(qsScratch != qsDmsName) {
                                                                qsDmsName = qsScratch;
                                                                qslTempList << qsScratch;

                                                        }
                                                }
                                        }
                                }
                        }
                }

                // Check the Progress Dialog Cancel Button
                if (qpdProgress->wasCanceled()) {
                        qslRetList.clear();
                        break;
                }
        }
        // Delete objects
        delete qpdProgress;

        // Make PLC-Symbols
        qslRetList << qslTempList;
    }
    // Close file
    qfInputFile.close();

    // Return value
    return qslRetList;
}



//! This Function reads the different Cpu tree's from the promos.dms file.
    /*!
      \param *struGenData A Pointer to a struGeneratorData structure which holds the
needed Information.
        \param qslAttributes A QStringList holding To Attributes to be searched.
      \return An int holding the DMS-Names of the To's matching one of the given To
Attributes
    */
int pLib::readTree(QString qsInProjectPath, QString qsSearchString, QString
qsFileName, QFile &qfTree) {
    // Locals
    QFile qfInputFile;
    QTextStream qtsInput, qtsOutput;
    int iRetValue;
    QString qsline;
    bool bStringFound = false;

    // Pfad setzen
    if(qsInProjectPath.isEmpty()) {
        QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
        // Set Return value
        iRetValue = -1;
    }
    else {
```

```
		// Set Filename
		qfInputFile.setFileName(qsInProjectPath);
		qfTree.setFileName("../tmp/" % qsFileName);
		qtsInput.setDevice(&qfInputFile);
		qtsOutput.setDevice(&qfTree);

		if (!qfInputFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
			// Set Return value
			iRetValue = -2;
		}
		else if (!qfTree.open(QIODevice::WriteOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "pLib", "Temporary File can't be opened");
			// Set Return value
			iRetValue = -3;
		}
		else {
			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfInputFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Searching " % qsSearchString % "
Tree...");
			qpdProgress->setValue(0);
			qpdProgress->show();

			// find Tree
			while (!qtsInput.atEnd()) {
				// Search String
				qsline = qtsInput.readLine();
				if(qsline.section(':', 0, 0) == qsSearchString) {
					// Set Flag
					bStringFound = true;

					// Write line to outputfile
					qtsOutput << qsline << "\n";

					// Set Progress dialog tText
					qpdProgress->setLabelText("Extracting " %
qsSearchString % " Tree...");
				}
				else {
					if(bStringFound) {
						// Reset Flag
						bStringFound = false;

						// break while-loop
						break;

					}
					else {
						// Reset Flag
						bStringFound = false;

						// Set progress bar
						qpdProgress->setValue(qfInputFile.pos());
					}
				}

				while(bStringFound) {
					qsline = qtsInput.readLine();
					if(qsline.section(':', 0, 0) == qsSearchString) {
						// Set progress bar
```

```
                                        qpdProgress->setValue(qfInputFile.pos());

                                        // Write line to outputfile
                                        qtsOutput << qsline << "\n";

                                        // Check the Progress Dialog Cancel Button
                                        if (qpdProgress->wasCanceled()) {
                                               break;
                                               // Set Return value
                                               iRetValue = -4;
                                        }
                                 }
                                 else {
                                        // break while-loop
                                        break;

                                        // Set Return value
                                        iRetValue = 0;
                                 }
                          }
                   }
                   // Delete objects
                   delete qpdProgress;

                   // Close file
                   qfInputFile.close();
                   qfTree.close();
            }
      }
      // Return value
      return iRetValue;
}
```

Appendix E III Source code pLibRessources.cpp

```
/*****************************************************************************
 Implementation of the pLib Functions for the ProMoS PLC-Ressource handling
 ---------------------------------------------------------------------------

 Date         : 12.08.2011
 File        : plib.cpp
 Author              : Thomas Gasser
 ©           : 2010 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version     : 1.0.0.0
 Changes      :

*****************************************************************************/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "plib.h"                       // Project header
/****************** Includes ************************/

//! This Function checks the Typ of the Ressource in the first parameter in the file
given in the second parameter .
    /*!
      \param qsInDmsName A QString holding the DMS-Name of the ressource to be type
checked.
         \param qsInFilename A QString holding the filename of the file where to search
the DMS-Name.
      \return A QString holding the Ressource addressed by the DMS-Names Data-Typ
    */
QString pLib::checkTyp(QString qsInDmsName, QString qsInFilename) {
       // Locals
       QFile qfInputFile;
       QTextStream qtsInput;
       QString qsRetValue;
       QString qsline, qsSearchString;

       // Pfad setzen
       if(qsInFilename.isEmpty() || qsInFilename.isNull()) {
              QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
              // Set Return value
              qsRetValue = "";
       }
       else {
              // Set Filename
              qfInputFile.setFileName(qsInFilename);
              qtsInput.setDevice(&qfInputFile);

              if (!qfInputFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                     QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                     // Set Return value
                     qsRetValue = "";
              }
              else {
                     // Setup Progress Dialog
                     QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfInputFile.size());
                     qpdProgress->setWindowModality(Qt::WindowModal);
                     qpdProgress->setLabelText("Searching Data-Typ for " %
qsInDmsName);
                     qpdProgress->setValue(0);
```

```
                    qpdProgress->show();

                    // Prepare search String
                    qsSearchString = qsInDmsName % ":PLC:Type;STR;";

                    // find Typ
                    while (!qtsInput.atEnd()) {
                           // Search String
                           qsline = qtsInput.readLine();

                           qsline = qtsInput.readLine();
                           if(qsline.contains(qsSearchString, Qt::CaseInsensitive)) {
                                  // Set progress bar
                                  qpdProgress->setValue(qfInputFile.pos());

                                  // Extract Data Typ;
                                  //
WB002C:K1:C09:YZ:009:Freigabe:PLC:Type;STR;Flag;RW
                                  qsRetValue = qsline.section(';', 2, 2);
                                  qpdProgress->setValue(qfInputFile.size());

                                  break; // terminate the while-loop if typ found and
extracted


                                  // Check the Progress Dialog Cancel Button
                                  if (qpdProgress->wasCanceled()) {
                                         break;
                                         // Set Return value
                                         qsRetValue = "";
                                  }
                           }
                    }
                    // Delete objects
                    delete qpdProgress;

                    // Close file
                    qfInputFile.close();
             }
       }
       // Return value
       return qsRetValue;
}
```

Appendix E IV Source code pLibTos.cpp

```
/*****************************************************************************
 Implementation of the pLib Functions for the ProMoS TO handling
 ---------------------------------------------------------------

 Date         : 12.08.2011
 File       : plib.cpp
 Author             : Thomas Gasser
 ©          : 2010 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "plib.h"                       // Project header
/****************** Includes ************************/
//! This Function searches the given promos.dms File and returns all To's given in the
second parameter in a QStringList.
    /*!
      \param *struGenData A Pointer to a struGeneratorData structure which holds the
needed Information.
        \param qsOutputTo A QString which holds To to be searched.
      \return A QStringList with all occurances of the To given in the second
parameter.
    */
QStringList pLib::getTos(const struct struGeneratorData *struGenData, QString
qsOutputTo){
       // Locals
       QFile qfFile;
       QTextStream qtsInput;
       QStringList qslRetList;
       QString qsline;

       // Pfad setzen
       if(struGenData->qsInputFile.isEmpty()) {
              QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
              qslRetList.clear();
       }
       else {
              // Set Filename
              qfFile.setFileName(struGenData->qsInputFile);
              qtsInput.setDevice(&qfFile);

              if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                     QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                     qslRetList.clear();
              }
              else {
                     // Setup Progress Dialog
                     QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
                     qpdProgress->setWindowModality(Qt::WindowModal);
                     qpdProgress->setLabelText("Suche VLO's vom Typ " + qsOutputTo);
                     qpdProgress->setValue(0);
                     qpdProgress->show();

                     // Search all To's with the same type as the OutputVLO
```

```
                        while (!qtsInput.atEnd()) {
                                // find entries
                                qsline = qtsInput.readLine();
                                qpdProgress->setValue(qfFile.pos());

                                if(!struGenData->bAdditionalConfiguration) {
                                        if(qsline.section(':', 0, 0) == struGenData-
>qsCpuName){
                                                if(qsline.contains(qsOutputTo)) {
                                                        qslRetList << qsline.section(':', 0,
(qsline.count(":") - 1));
                                                }
                                        }
                                }
                                else {
                                        if(qsline.section(':', 0, 0) == struGenData-
>qsCpuName && qsline.section(':', 1, 1) == struGenData->qsAdditionalSeperator){
                                                if(qsline.contains(qsOutputTo)) {
                                                        qslRetList << qsline.section(':', 0,
(qsline.count(":") - 1));
                                                }
                                        }
                                }

                                // Check the Progress Dialog Cancel Button
                                if (qpdProgress->wasCanceled()) {
                                break;
                                }
                        }
                        // Delete objects
                        delete qpdProgress;
                }
        }
        // Close file
        qfFile.close();

        // Return value
        return qslRetList;
}

//! This Function searches the given bmo.dms File and returns all To's found.
    /*!
      \param qsPath A QString holding the Path and the Filename to the bmo.dms File.
      \return A QStringList with all To's found.
    */
QStringList pLib::getTos(QString qsFileName){
        // Locals
        QFile qfFile;
        QTextStream qtsInput;
        QStringList qslRetList;
        QString qsline, qsObjectType = "";

        // Pfad setzen
        if(qsFileName.isEmpty()) {
                QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
                qslRetList.clear();
        }
        else {
                // Set Filename
                qfFile.setFileName(qsFileName);
                qtsInput.setDevice(&qfFile);
```

```
		if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
			qslRetList.clear();
		}
		else {
			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Suche VLO's....");
			qpdProgress->setValue(0);
			qpdProgress->show();

			// Search all To's
			while (!qtsInput.atEnd()) {
				// find entries
				qsline = qtsInput.readLine();
				qpdProgress->setValue(qfFile.pos());

				if(qsline.section(':', 1, 1) != qsObjectType){
					if(qsline.contains("OBJECT;STR;")) {
						qsObjectType = qsline.section(';', 2, 2);
						qslRetList << qsObjectType;
					}
				}
				if(!qsObjectType.isEmpty()) {
					if(qsline.contains(qsObjectType + ";STR;")) {
						qslRetList << qsline.section(';', 2, 2);
					}
				}

			// Check the Progress Dialog Cancel Button
				if (qpdProgress->wasCanceled()) {
				break;
				}
			}
			// Delete objects
			delete qpdProgress;
		}
	}
	// Close file
	qfFile.close();

	// Return value
	return qslRetList;
}

//! This Function searches the tree file of the Cpu given in the first parameter and
returns all instances of the To given in the secon parameter.
    /*!
      \param qsInCpuName A QString holding the name of the Cpu where the active TO is
used.
        \param qsInToName A QString holding the name of the active TO.
      \return A QStringList with all instances found.
    */
QStringList pLib::getToInstances(QString qsFileName, QString qsInToName){
	// Locals
	QFile qfCpuTree;
	QTextStream qtsInput;
	QStringList qslRetList;
	QString qsline, qsInstanceDmsName;

	// Pfad setzen
```

```
	if(qsFileName.isEmpty() || qsFileName.isNull() || qsInToName.isEmpty() ||
qsInToName.isNull()) {
		QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
		qslRetList.clear();
	}
	else {
		// Set Filename
		qfCpuTree.setFileName(qsFileName);
		qtsInput.setDevice(&qfCpuTree);

		if (!qfCpuTree.open(QIODevice::ReadOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
			qslRetList.clear();
		}
		else {
			// Setup Progress Dialog
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfCpuTree.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Suche Instanzen des VLO: " % qsInToName
% "...");
			qpdProgress->setValue(0);
			qpdProgress->show();

			// Search all Instances
			while (!qtsInput.atEnd()) {
				// find entries
				qsline = qtsInput.readLine();
				qpdProgress->setValue(qfCpuTree.pos());

				// Search Object Typ
				if(qsline.contains(":OBJECT;STR;", Qt::CaseSensitive)) {
					if(qsline.section(';', 2, 2) == qsInToName) {
						qsInstanceDmsName = qsline.section(':', 1,
(qsline.lastIndexOf(':', 0) - 1));
						qslRetList << qsInstanceDmsName;
					}
				}

				// Check the Progress Dialog Cancel Button
				if (qpdProgress->wasCanceled()) {
					// Clear list
					qslRetList.clear();

					// Terminate while-Loop
					break;
				}
			}
			// Delete objects
			delete qpdProgress;
		}
	}
	// Close file
	qfCpuTree.close();

	// Return value
	return qslRetList;
}

//! This Function searches the given CpuTree File and returns the Description of the
To's given in the second parameter.
    /*!
```

```
      \param qsFileName A QString holding the Path and the Filename to the promos.dms
File.
        \param qslDmsNames A QStringList holding the To's whose description should be
searched
      \return A QStringList with the descriptions.
    */
QStringList pLib::getToDescription(QString qsFileName, QStringList qslDmsNames) {
       // Locals
       QFile qfFile;
       QTextStream qtsInput;
       QStringList qslDescription;
       QString qsline, qsScratch1, qsScratch2, qsScratch3;

       // Pfad setzen
       if(qsFileName.isEmpty()) {
              QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
              qslDmsNames.clear();
       }
       else {
              // Set Filename
              qfFile.setFileName(qsFileName);
              qtsInput.setDevice(&qfFile);

              if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                     QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                     qslDmsNames.clear();
              }
              else {
                     // Set Progress
                     QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
                     qpdProgress->setWindowModality(Qt::WindowModal);
                     qpdProgress->setLabelText("Suche Bezeichnungen...");
                     qpdProgress->show();


                     // take item from string list
                     while (!qslDmsNames.isEmpty()) {
                            qsScratch1 = qslDmsNames.takeAt(0);
                            qsScratch2 = "BMO:" % qsScratch1;
                            qsScratch3 = "BMO:" % qsScratch1 % ";STR;";
                            qpdProgress->setLabelText("Suche Bezeichnungen für TO: " %
qsScratch1 % "...");

                            // find description
                            while (!qtsInput.atEnd()) {
                                   // find entries
                                   qsline = qtsInput.readLine();
                                   qpdProgress->setValue(qfFile.pos());
                                   if(qsline.contains(qsScratch2)) {
                                          if(qsline.contains(qsScratch3)) {
                                                 qslDescription << qsline.section(';',
2, 2);
                                                 break;
                                          }
                                   }
                                   // Check the Progress Dialog Cancel Button
                                   if (qpdProgress->wasCanceled()) {
                                   break;
                                   }
                            }
```

```
                        // Rewinde file
                        qtsInput.seek(0);
                    }
                    // Delete objects
                    delete qpdProgress;
                }
        }
        // Close file
        qfFile.close();

        // Return value
        return qslDescription;
}


//! This Function searches the given CpuTree File and returns the Instances
Description of the To's given in the second parameter.
    /*!
      \param qsFileName A QString holding the Path and the Filename to the promos.dms
File.
        \param qslDmsNames A QStringList holding the To's whose description should be
searched
      \return A QStringList with the descriptions.
    */
QStringList pLib::getInstanceDescription(QString qsFileName, QStringList qslDmsNames)
{
        // Locals
        QFile qfFile;
        QTextStream qtsInput;
        QStringList qslDescription;
        QString qsline, qsScratch1, qsScratch2;

        // Pfad setzen
        if(qsFileName.isEmpty()) {
                QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
                qslDmsNames.clear();
        }
        else {
                // Set Filename
                qfFile.setFileName(qsFileName);
                qtsInput.setDevice(&qfFile);

                if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                        QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                        qslDmsNames.clear();
                }
                else {
                        // Set Progress
                        QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
                        qpdProgress->setWindowModality(Qt::WindowModal);
                        qpdProgress->setLabelText("Suche Bezeichnungen...");
                        qpdProgress->show();


                        // take item from string list
                        while (!qslDmsNames.isEmpty()) {
                                qsScratch1 = qslDmsNames.takeAt(0);
                                qsScratch2 = qsScratch1 % ":NAME;STR;";
                                qpdProgress->setLabelText("Suche Bezeichnungen für TO-
Instanz: " % qsScratch1 % "...");
```

```
                        // find description
                        while (!qtsInput.atEnd()) {
                                // find entries
                                qsline = qtsInput.readLine();
                                qpdProgress->setValue(qfFile.pos());
                                if(qsline.contains(qsScratch1)) {
                                        if(qsline.contains(qsScratch2)) {
                                                qslDescription << qsline.section(';',
2, 2);

                                                break;
                                        }
                                }
                                // Check the Progress Dialog Cancel Button
                                if (qpdProgress->wasCanceled()) {
                                break;
                                }
                        }

                        // Rewinde file
                        qtsInput.seek(0);
                }
                // Delete objects
                delete qpdProgress;
        }
    }
    // Close file
    qfFile.close();

    // Return value
    return qslDescription;
}


//! This Function searches the given CpuTree File and returns the Description of the
To's given in the second parameter.
  /*!
    \param qsFileName A QString holding the Path and the Filename to the bmo.dms
File.
      \param qsInToName A QString holding the TO whose attributes should be searched
    \return A QStringList with the attributes.
  */
QStringList pLib::getAttributes(QString qsFileName, QString qsInToName){
    // Locals
    QFile qfFile;
    bool bToFound = false;
    QTextStream qtsInput;
    QStringList qslAttributes, qslSearchStrings;
    QString qsline;

    // Pfad setzen
    if(qsFileName.isEmpty() && !qsFileName.isNull()) {
            QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
            qslAttributes.clear();
    }
    else {
            // Set Filename
            qfFile.setFileName(qsFileName);
            qtsInput.setDevice(&qfFile);

            if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                    QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                    qslAttributes.clear();
```

```
		}
		else {
			// Set Progress
			QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
			qpdProgress->setWindowModality(Qt::WindowModal);
			qpdProgress->setLabelText("Searching attributes for TO: " %
qsInToName % "...");
			qpdProgress->show();

			// Setup search strings
			qslSearchStrings << "BMO:" % qsInToName %":";
			qslSearchStrings << ":PLC:Address;";
			qslSearchStrings << ":PAR_DATA;STR";
			qslSearchStrings << ";STR;O.";
			qslSearchStrings << ";STR;I.";
			qslSearchStrings << ";STR;F.";

			// search entries
			while (!qtsInput.atEnd()) {
				// find entries
				qsline = qtsInput.readLine();
				qpdProgress->setValue(qfFile.pos());
				if(qsline.contains(qslSearchStrings.at(0))) {
					bToFound = true;
					if(qsline.contains(qslSearchStrings.at(1))) {
						qslAttributes << qsline.section(':', -3, -3);
					}
					else if(qsline.contains(qslSearchStrings.at(2))) {
						qslAttributes << qsline.section(':', -2, -2);
					}
					else if(qsline.contains(qslSearchStrings.at(3))) {
						qslAttributes << qsline.section(';', 0, 0);
					}
					else if(qsline.contains(qslSearchStrings.at(4))) {
						qslAttributes << qsline.section(';', 0, 0);
					}
					else if(qsline.contains(qslSearchStrings.at(5))) {
						qslAttributes << qsline.section(';', 0, 0);
					}
				}
				else {
					// Check if TO was found
					if(bToFound) {
						break; // break while loop if TO was found but
the line read doesn't contains the TO-Nmae anymore
					}
				}
				// Check the Progress Dialog Cancel Button
				if (qpdProgress->wasCanceled()) {
				break;
				}
			}
			// Delete objects
			delete qpdProgress;
		}
	}
	// Close file
	qfFile.close();
	// Return value
	return qslAttributes;
}
```

```
//! This Function searches the given CpuTree File and returns the Description of the
To's given in the second parameter.
    /*!
      \param qsFileName A QString holding the Path and the Filename to the bmo.dms
File.
        \param qsInToName A QString holding the TO whose attribute descriptions should
be searched
        \param qslAttributes A QStringList holding the TO's attributes
      \return A QStringList with the attribute descriptions.
    */
QStringList pLib::getAttributeDescriptions(QString qsFileName, QString qsInToName,
QStringList qslAttributes){
       // Locals
       QFile qfFile;
       bool bToFound = false;
       QTextStream qtsInput;
       qint64 qi64FirstFound;
       QStringList qslAttributeDescriptions;
       QString qsline, qsSearchString, qsScratch;

       // Pfad setzen
       if(qsFileName.isEmpty() && !qsFileName.isNull()) {
              QMessageBox::warning(0, "pLib", "Bitte wählen Sie eine Quellen-Datei
aus");
              qslAttributes.clear();
       }
       else {
              // Set Filename
              qfFile.setFileName(qsFileName);
              qtsInput.setDevice(&qfFile);

              if (!qfFile.open(QIODevice::ReadOnly | QIODevice::Text)) {
                     QMessageBox::warning(0, "pLib", "Kein File im Ordner gefunden");
                     qslAttributes.clear();
              }
              else {
                     // Set Progress
                     QProgressDialog *qpdProgress = new QProgressDialog("",
"Abbrechen", 0, qfFile.size());
                     qpdProgress->setWindowModality(Qt::WindowModal);
                     qpdProgress->setLabelText("Searching attribute descriptions for
TO: " % qsInToName % "...");
                     qpdProgress->show();

                     // For each entry in the Attribute List
                     while(!qslAttributes.isEmpty()) {
                            qsScratch = qslAttributes.takeAt(0);
                            qpdProgress->setLabelText("Suche Bezeichnungen für
Attribut: " % qsScratch % "...");

                            // Setup search strings
                            qsSearchString ="BMO:" % qsInToName % ":" %  qsScratch %
":Comment;STR;";

                            // find description
                            while (!qtsInput.atEnd()) {
                                   // find entries
                                   qsline = qtsInput.readLine();
                                   qpdProgress->setValue(qfFile.pos());
                                   if(qsline.contains(qsSearchString)) {
                                          bToFound = true;
                                          qslAttributeDescriptions <<
qsline.section(';', -2, -2);
```

```
                                        }
                                        else {
                                                // Check if TO was found
                                                if(bToFound) {
                                                        // Rewinde file to first found position
                                                        qtsInput.seek(0);

                                                        // Reset Flag
                                                        bToFound = false;

                                                        break; // break inner while loop if TO
was found but the line read doesn't contains the TO-Nmae anymore
                                                }
                                        }
                                        // Check the Progress Dialog Cancel Button
                                        if (qpdProgress->wasCanceled()) {
                                        break;
                                        }
                                }
                        }
                        // Delete objects
                        delete qpdProgress;
                }
        }
        // Close file
        qfFile.close();

        // Return value
        return qslAttributeDescriptions;
}
```

Appendix F Sources sGen

Appendix F I Source code sGen.h

```
#ifndef SGEN_H
#define SGEN_H

#include <QtGui>
#include <QLocalSocket>                 // QLocalSocket
#include "qcdatatree.h"
#include "ui_sgen.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pTSocketLib/ptsocketlib.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pTSocketLib/ptclientmachine.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pTSocketLib/common.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/sGenLib/sgenlib.h"
#include "D:/Thoemus_Stuff/Projekte/C++/pTool/pLib/pLib.h"


/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

//!  This is the sGen class, it implements the Code-Generator for PLC-Systems of Saia-
Burgess.
/*!
  This is the sGen class, it implements the Code-Generator for PLC-Systems of Saia-
Burgess.
*/
class sGen : public QDialog {
	Q_OBJECT

public:
	sGen(QWidget *parent = 0, Qt::WFlags flags = 0);
	~sGen();

private:
	Ui::sGenClass ui;

	// dll's einbinden
	pTSocketLib *socketLib;
	sGenLib *generatorLib;
	pLib *libUtility;

	// Settings
	QSettings *qsSettings;

	// Structures
	struct struHeaderData struHeader;

	// Start Data Exchange timer
	QTimer *qtStartDataExchange;

	// Client machine
	ptClientMachine *clientMachine;

	// Answer
	QStringList qslAnswer;
	QQueue<QStringList> qqAswerQueue;

	// Sender
```

```
QString qsSender;

// Structures
struct struCommand struCmd;

// File
QFile qfOutput;
QFile qfInput;

// Data-Stream
QTextStream qtsOutput;
QTextStream qtsInput;

// Filenames
QString qsNameOfOutputFile;

// Data machine
QStateMachine *qsmDataMachine;
QStateMachine *qsmGeneratorMachine;

// Data Steps
QState *qsmsDataMachineIdle;;
QState *qsmsGetProjectList;
QState *qsmsGetCpuList;
QState *qsmsGetToList;
QState *qsmsGetInstanceList;
QState *qsmsGetInstancesByType;

QState *qsmsGetData;
QFinalState *qsmfsEndDataMachine;

// Data machine step counter
int iDatamachineStep;

//Data machine functions
void checkProject();
void checkCpus();
void checkTos();
void checkInstances();
void checkInstancesByType();
void checkData();
void checkComment();
void checkAttributeComment();
void checkPath();
void checkDataForTree();
void dataMachineErrorHandler();

// Generator Steps (SPS-Code)
QState *qsmsOpenFile;
QState *qsmsWriteHeader;
QState *qsmsWriteAdminStuff;
QState *qsmsWriteToCalls;
QState *qsmsWriteFrame;
QState *qsmsWriteToCode;
QState *qsmsWriteFooter;
QState *qsmsCloseFile;
QFinalState *qsmfsEndGeneratorMachine;

// Generator Steps (RXP)
QState *qsmsOpenRxpFile;
QState *qsmsWriteRxpHeader;
QState *qsmsWriteSymbols;
QState *qsmsCloseRxpFile;
```

```
        QFinalState *qsmfsEndRxpGeneratorMachine;

        // Generator Steps (VM)
        QState *qsmsOpenVmFile;
        QState *qsmsWriteVmHeader;
        QState *qsmsWriteVmSymbols;
        QState *qsmsCloseVmFile;
        QFinalState *qsmfsEndVmGeneratorMachine;

        // TO-List
        QStringList qslToList;
        QStringList qslpTInstanceList;
        QStringList qslpTDataList;
        QStringList qslpTCommentList;
        QStringList qslpTAttributeCommentList;
        QStringList qslDataForTree;

        // Hashes
        QHash<QString, qcdatatree*> qhTrees;
        QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*> *qmhLtoHash;
        QHash<QString, QString>    *qhDataHash;
        QMultiHash<QString, QHash<QString, QString>*> *qmhDatapointHash;

        QList<QHash<QString, QString>*> qlHashes;
        QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
qlHashes2;
        QList<QMultiHash<QString, QHash<QString, QString>*>*> qlMultiHashes;


        // Maybe obsolet
        QHash<QString, QStringList> qhDmsNames;
        QHash<QString, QStringList> qhAttributes;
        QHash<QString, QStringList> qhToTypes;

        QHash<QString, QStringList> qhGroup;
        QHash<QString, QHash<QString, QStringList>> qhDmsNamesByGroupAndType;

        // Standard Model
        qcdatatree *qtmDataTree;

        // Private Functions
        bool createTrees(QStringList qslInDataForTree);

private slots:
        // GUI slots
        void on_qpbConnect_clicked();
        void on_qpbDebug_clicked();
        void on_qpbSendCommand_clicked();
        void on_qpbStartBatch_clicked();
        void on_qpbGenerateRxp_clicked();
        void on_qpbGenerateVm_clicked();
        void if_connectedToServer();
        void if_disconnectedFromServer();
        void if_answerIsReady();

        // Data machine slots
        void if_qsmsDataMachineIdle_entered();
        void if_qsmsGetProjectList_entered();
        void if_qsmsGetCpuList_entered();
        void if_qsmsGetToList_entered();
        void if_qsmsGetInstanceList_entered();
        void if_qsmsGetInstancesByType_entered();
```

```
        void if_qsmsGetData_entered();
        void if_qsmfsEndDataMachine_entered();
        void if_qsmfsEndDataMachine_stopped();
        void if_qtStartDataExchange_timeout();

        // RXP Generator slots
        void if_qsmsOpenRxpFile_entered();
        void if_qsmsWriteRxpHeader_entered();
        void if_qsmsWriteSymbols_entered();
        void if_qsmsCloseRxpFile_entered();
        void if_qsmfsEndRxpGeneratorMachine_entered();

        // VM Generator slots
        void if_qsmsOpenVmFile_entered();
        void if_qsmsWriteVmHeader_entered();
        void if_qsmsWriteVmSymbols_entered();
        void if_qsmsCloseVmFile_entered();
        void if_qsmfsEndVmGeneratorMachine_entered();

signals:
        // Data machine Signals
        void doDataExchange();
        void projectRecceived();
        void cpusRecceived();
        void tosRecceived();
        void instancesRecceived();
        void instancesByTypeRecceived();
        void dataRecceived();
        void dataMachineStopped();
        void dataMachineError();

        // Generator Signals
        void fileOpen();
        void fileOpenError();
        void headerWritten();
        void adminStuffWritten();
        void toCallsWritten();
        void frameWritten();
        void toCodeWritten();
        void footerWritten();
        void fileClosed();
        void generatorMachineStopped();

        // RXP Generator Signals
        void rxpFileOpened();
        void rxpHeaderWritten();
        void rxpSymbolsWritten();
        void rxpFileClosed();
        void rxpGenerated();

        // VM Generator Signals
        void vmFileOpened();
        void vmHeaderWritten();
        void vmSymbolsWritten();
        void vmFileClosed();
        void vmGenerated();
};

#endif // SGEN_H
```

Appendix F II Source code main.cpp

```
#include "stdafx.h"
#include "sgen.h"
#include <QtGui/QApplication>
#include <QtGlobal>

//! This Function receives the debug Messages and shows them.
    /*!
         \param type A QtMsgType specifing the type of the recceived message.
         \param msg A const char* which holds the message to insert into the message
window.
      \return No return value
    */
void debugOutput(QtMsgType type, const char* msg) {
       // Create static Message Browser and set it up
       static QTextBrowser* qtbMessageBrowser = new QTextBrowser();
       qtbMessageBrowser->setWindowTitle("Debug Window");
       qtbMessageBrowser->move(500, 0);
       qtbMessageBrowser->show();

       // Check which message type it is
       switch(type) {
              // Debug message
              case QtDebugMsg:
                     qtbMessageBrowser->append(QString("Debug : %1").arg(msg));
                     break;
              // Warning
              case QtWarningMsg:
                     qtbMessageBrowser->append(QString("Warning     : %1").arg(msg));
                     break;
              // Critical
              case QtCriticalMsg:
                     qtbMessageBrowser->append(QString("Critical    : %1").arg(msg));
                     break;
              // Fatal
              case QtFatalMsg:
                     qtbMessageBrowser->append(QString("Fatal : %1").arg(msg));
                     break;
       }
}


int main(int argc, char *argv[]) {

       QApplication app(argc, argv);
       #ifdef Q_OS_WIN
              #ifndef QT_NO_DEBUG_OUTPUT
                     qInstallMsgHandler(debugOutput);
              #endif
       #endif
       app.setApplicationName(app.translate("main", "sGen"));
       app.setOrganizationName("tG Soft");
       app.setOrganizationDomain("tGSoft.ch");
       sGen w;
       w.show();
       return app.exec();
}
```

Appendix F III Source code sGen.cpp

```
/*****************************************************************************
 Implementation of the Saia Code Generator
 -----------------------------------------

 Date         : 15.10.2011
 File       : sGen.cpp
 Author             : Thomas Gasser
 ©          : 2011 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's Saia Code Generator.
/*!
  This is TG's Saia Code Generator
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "sgen.h"                       // Project header
/****************** Includes ************************/


// Constructor / Destructor
//! This is the Constructor of the sGen application.
    /*!
         Here the on start stuff is made
    */
sGen::sGen(QWidget *parent, Qt::WFlags flags): QDialog(parent, flags) {
       ui.setupUi(this);
       // Locals
       bool bTest = false;

       // fill in combobox
       QStringList qslItems;
       qslItems << "getProjects" << "getCPUs" << "getTos" << "getInstances" <<
"getObjectsByType" << "getData" << "getComment" << "getAtributeComment" <<
"getProjectPath" << "getDataForTree" ;
       ui.qcbCommand->addItems(qslItems);

       // dll einbinden
       socketLib = new pTSocketLib();
       generatorLib = new sGenLib();
       libUtility = new pLib();

       // Create settings object
       qsSettings = new QSettings("../cfg/sGen.ini", QSettings::IniFormat, this);

       // Client machine
       clientMachine = new ptClientMachine(this);

       // Configure client machine
       qsSender = "sGen";
       clientMachine->setSender(qsSender);

       // Set command struct to default values
       struCmd.qsProject = "";
       struCmd.qsCpu = "";
```

```
        struCmd.qsCommand = "0";
        struCmd.qsSender = qsSender;
        struCmd.qsConHandle = "";
        struCmd.qsAttribut = "No Attribute";

        // Connecting Signals
        bTest = connect(clientMachine, SIGNAL(connectedToServer()), this,
SLOT(if_connectedToServer()));
        bTest = connect(clientMachine, SIGNAL(disconnectedFromServer()), this,
SLOT(if_disconnectedFromServer()));
        bTest = connect(clientMachine, SIGNAL(answerReady()), this,
SLOT(if_answerIsReady()));

        // Create Start data exchange timer
        qtStartDataExchange = new QTimer(this);
        qtStartDataExchange->setInterval(2000);
        qtStartDataExchange->setSingleShot(true);
        bTest = connect(qtStartDataExchange, SIGNAL(timeout()), this,
SLOT(if_qtStartDataExchange_timeout()));

        // Data State Machine
        qsmDataMachine = new QStateMachine(this);

        // Steps of the Data machine
        qsmsDataMachineIdle = new QState();
        qsmsGetProjectList = new QState();
        qsmsGetCpuList = new QState();
        qsmsGetToList = new QState();
        qsmsGetInstanceList = new QState();
        qsmsGetInstancesByType = new QState();
        qsmsGetData = new QState();
        qsmfsEndDataMachine = new QFinalState();

        // Add steps to the data machine
        qsmDataMachine->addState(qsmsDataMachineIdle);
        qsmDataMachine->addState(qsmsGetProjectList);
        qsmDataMachine->addState(qsmsGetCpuList);
        qsmDataMachine->addState(qsmsGetToList);
        qsmDataMachine->addState(qsmsGetInstanceList);
        qsmDataMachine->addState(qsmsGetInstancesByType);
        qsmDataMachine->addState(qsmsGetData);
        qsmDataMachine->addState(qsmfsEndDataMachine);

        // Set initial step of the data machine
        qsmDataMachine->setInitialState(qsmsDataMachineIdle);

        // Add transitions to the states of the data machine
        qsmsDataMachineIdle->addTransition(this, SIGNAL(doDataExchange()),
qsmsGetProjectList);
        qsmsGetProjectList->addTransition(this, SIGNAL(projectRecceived()),
qsmsGetCpuList);
        qsmsGetCpuList->addTransition(this, SIGNAL(cpusRecceived()), qsmsGetToList);
        qsmsGetToList->addTransition(this, SIGNAL(tosRecceived()), qsmsGetInstanceList);
        qsmsGetInstanceList->addTransition(this, SIGNAL(instancesRecceived()),
qsmsGetInstancesByType);
        qsmsGetInstancesByType->addTransition(this, SIGNAL(instancesByTypeRecceived()),
qsmsGetData);
        qsmsGetData->addTransition(this, SIGNAL(dataRecceived()), qsmfsEndDataMachine);


        // Establish connections of the states of the data machine
        bTest = connect(qsmsDataMachineIdle, SIGNAL(entered()), this,
SLOT(if_qsmsDataMachineIdle_entered()));
```

```
	bTest = connect(qsmsGetProjectList, SIGNAL(entered()), this,
SLOT(if_qsmsGetProjectList_entered()));
	bTest = connect(qsmsGetCpuList, SIGNAL(entered()), this,
SLOT(if_qsmsGetCpuList_entered()));
	bTest = connect(qsmsGetToList, SIGNAL(entered()), this,
SLOT(if_qsmsGetToList_entered()));
	bTest = connect(qsmsGetInstanceList, SIGNAL(entered()), this,
SLOT(if_qsmsGetInstanceList_entered()));
	bTest = connect(qsmsGetInstancesByType, SIGNAL(entered()), this,
SLOT(if_qsmsGetInstancesByType_entered()));
	bTest = connect(qsmsGetData, SIGNAL(entered()), this,
SLOT(if_qsmsGetData_entered()));
	bTest = connect(qsmfsEndDataMachine, SIGNAL(entered()), this,
SLOT(if_qsmfsEndDataMachine_entered()));
	bTest = connect(qsmDataMachine, SIGNAL(stopped()), this,
SLOT(if_qsmfsEndDataMachine_entered()));

	// Set defaults
	qslAnswer.clear();
	qqAswerQueue.clear();
}

//! This is the Destructor of the pTool application.
    /*!
        Here some stuff is made just before the application is finished
    */
sGen::~sGen() {

}

void sGen::on_qpbConnect_clicked() {
	// Locals

	// Connect to server
	clientMachine->connectServer("pTool");

	// Change caption
	if(ui.qpbConnect->text() != "Disconnect") {
		ui.qpbConnect->setText("Disconnect");
	}
	else {
		ui.qpbConnect->setText("Connect");
	}

	// Return value
	return;
}

void sGen::on_qpbSendCommand_clicked() {
	// Locals
	QVariant qvScratch;

	// Check if something is highlighted in the tree combobox
	iDatamachineStep = ui.qcbCommand->currentIndex();

	if(iDatamachineStep < 0) {
		QMessageBox::warning(this, "sGen", "No Command chossen");
	}

	// Set command
	qvScratch  = iDatamachineStep;
	struCmd.qsCommand = qvScratch.toString();
```

```
	// Set cpu
	struCmd.qsCpu = ui.qcbCpus->currentText();

	// Clear answer queue
	if(!qqAswerQueue.isEmpty()) {
		qqAswerQueue.clear();
	}

	// Debug output
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu <<
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;

	// Send command
	clientMachine->sendCommand(&struCmd);

	// Return value
	return;
}

void sGen::on_qpbStartBatch_clicked() {
	// Locals

	// Start data machine
	if(!qsmDataMachine->isRunning()) {
		qsmDataMachine->start();
	}

	// Start Data exchange timer
	qtStartDataExchange->start();

	// Change caption
	ui.qpbStartBatch->setText("Batch started...");

	// Return value
	return;
}

void sGen::if_answerIsReady() {
	// Locals

	// Read answer
	qslAnswer.clear();
	qslAnswer = clientMachine->qslAnswer;
	clientMachine->qslAnswer.clear();
	qqAswerQueue.enqueue(qslAnswer);

	// Check answer
	if(qsmDataMachine->isRunning()) {
		// Decide which Signal to emit
		switch(iDatamachineStep) {
			case 1:
				checkProject();
				qslAnswer.clear();
			break;

			case 2:
				checkCpus();
				qslAnswer.clear();
			break;

			case 3:
				checkTos();
				qslAnswer.clear();
```

```
                        break;

                        case 4:
                               checkInstances();
                               qslAnswer.clear();
                        break;

                        case 5:
                               checkInstancesByType();
                               qslAnswer.clear();
                        break;

                        case 6:
                               checkData();
                               qslAnswer.clear();
                        break;

                        case 7:
                               checkComment();
                               qslAnswer.clear();
                        break;

                        case 8:
                               checkAttributeComment();
                               qslAnswer.clear();
                        break;

                        case 9:
                               checkPath();
                               qslAnswer.clear();
                        break;

                        default:
                               dataMachineError();
                               qslAnswer.clear();
                        break;
                }

        }
        /*else if(qsmGeneratorMachine->isRunning()) {

        }*/
        else {
                // Decide which Signal to emit
                switch(iDatamachineStep) {
                        case 0:
                               checkProject();
                               qslAnswer.clear();
                        break;

                        case 1:
                               checkCpus();
                               qslAnswer.clear();
                        break;

                        case 2:
                               checkTos();
                               qslAnswer.clear();
                        break;

                        case 3:
                               checkInstances();
                               qslAnswer.clear();
```

```
                        break;

                        case 4:
                               checkInstancesByType();
                               qslAnswer.clear();
                        break;

                        case 5:
                               checkData();
                               qslAnswer.clear();
                        break;

                        case 6:
                               checkComment();
                               qslAnswer.clear();
                        break;

                        case 7:
                               checkAttributeComment();
                               qslAnswer.clear();
                        break;

                        case 8:
                               checkPath();
                               qslAnswer.clear();
                        break;

                        case 9:
                               checkDataForTree();
                               qslAnswer.clear();
                        break;

                        default:
                               dataMachineError();
                               qslAnswer.clear();
                        break;
                }
        }


        // Return value
        return;
}

void sGen::if_connectedToServer() {
        // Enable the Send-Button
        ui.qpbSendCommand->setEnabled(true);

        // Return value
        return;
}

void sGen::if_disconnectedFromServer() {
        // Enable the Send-Button
        ui.qpbSendCommand->setEnabled(false);

        // Stopp DataMachine if she is running
        if(qsmDataMachine->isRunning()) {
                qsmDataMachine->stop();
        }

        // Set command struct to default values
        struCmd.qsProject = "";
```

```
	struCmd.qsCpu = "";
	struCmd.qsCommand = "0x00";
	struCmd.qsSender = qsSender;
	struCmd.qsConHandle = "";
	struCmd.qsAttribut = "No Attribute";

	// Return value
	return;
}

void sGen::if_qtStartDataExchange_timeout() {
	// Start Data exchange
	emit doDataExchange();

	// Return value
	return;
}

void sGen::on_qpbDebug_clicked() {
	// Locals
	QStringList qslTempList1, qslTempList2;

	// Hashes
	qslTempList1 = qhDmsNames.keys();
	for(int i = 0; i < qslTempList1.size(); ++i) {
		qslTempList2 = qhDmsNames.value(qslTempList1.at(i));
	}

	qslTempList1 = qhToTypes.keys();
	for(int i = 0; i < qslTempList1.size(); ++i) {
		qslTempList2 = qhDmsNames.value(qslTempList1.at(i));
	}

	qslTempList1 = qhToTypes.keys();
	for(int i = 0; i < qslTempList1.size(); ++i) {
		qslTempList2 = qhDmsNames.value(qslTempList1.at(i));
	}

	qslTempList1 = qslToList;

	// Return value
	return;
}

//! This is the Create-Trees slot.
    /*!
         The data is loaded from the QStringList received from pTool
         \return A bool           true := Reading the list was successful,
                                             false := Reading the list was
unsuccessful.
    */
bool sGen::createTrees(QStringList qslInDataForTree) {
	// Locals
	QString qsScratch;

	// Create system tree
	qtmDataTree = new qcdatatree(this);

	// Call load-function of the model
	qtmDataTree->bLoadData(qslInDataForTree);
	qtmDataTree->setHorizontalHeaderLabels(QStringList() << "Data-Tree");

	// Move address into hash
```

```
	if(!qslInDataForTree.isEmpty()) {
		qsScratch = qslInDataForTree.at(0);
		qsScratch = qsScratch.section(":", 1, 1);
	}
	else {
		qsScratch = "???";
	}

	// Move tree into hash
	qhTrees.insert("Data-Tree", qtmDataTree);

	// Return value
	return true;
}
```

Appendix F IV Source code dataMachine.cpp

```
/*****************************************************************************
 Implementation of the Data Machine of the Saia Generator
 ---------------------------------------------------

 Date         : 04.03.2012
 File       : dataMachine.cpp
 Author             : Thomas Gasser
 ©          : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

// ******************** Includes ************************
#include "stdafx.h"                     // Precompiled headers
#include "sgen.h"                       // Project header
// ***************** End of Includes *********************


// ***************** Step Functions *********************
void sGen::if_qsmsDataMachineIdle_entered() {
       // Debug message
       qDebug() << "Entering step: if_qsmsDataMachineIdle_entered";

       // Set Step counter
       iDatamachineStep = 0;

       // Clear answer queue
       if(!qqAswerQueue.isEmpty()) {
              qqAswerQueue.clear();
       }

       // Return value
       return;
}

void sGen::if_qsmsGetProjectList_entered() {
       // Debug message
       qDebug() << "Entering step: if_qsmsGetProjectList_entered";

       // Set Step counter
       iDatamachineStep = 1;

       // Debug output
       qDebug() << "Getting Project List";

       // Set command
       struCmd.qsCommand = "0";

       // Clear answer queue
       if(!qqAswerQueue.isEmpty()) {
              qqAswerQueue.clear();
       }

       // Send command
       clientMachine->sendCommand(&struCmd);

       // Debug output
```

```
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu << 
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;

	// Return value
	return;
}

void sGen::if_qsmsGetCpuList_entered() {
	// Debug message
	qDebug() << "Entering step: if_qsmsGetCpuList_entered";

	// Set Step counter
	iDatamachineStep = 2;

	// Debug output
	qDebug() << "Getting Cpu List";

	// Set command
	struCmd.qsCommand = "1";

	// Clear answer queue
	if(!qqAswerQueue.isEmpty()) {
		qqAswerQueue.clear();
	}

	// Send command
	clientMachine->sendCommand(&struCmd);

	// Debug output
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu << 
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;

	// Return value
	return;
}

void sGen::if_qsmsGetToList_entered() {
	// Debug message
	qDebug() << "Entering step: if_qsmsGetToList_entered";

	// Set Step counter
	iDatamachineStep = 3;

	// Debug output
	qDebug() << "Getting To List";

	// Set command
	struCmd.qsCommand = "2";

	// Set cpu
	struCmd.qsCpu = ui.qcbCpus->currentText();

	// Clear answer queue
	if(!qqAswerQueue.isEmpty()) {
		qqAswerQueue.clear();
	}

	// Send command
	clientMachine->sendCommand(&struCmd);

	// Debug output
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu << 
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;
```

```
	// Return value
	return;
}

void sGen::if_qsmsGetInstanceList_entered() {
	// Debug message
	qDebug() << "Entering step: if_qsmsGetInstanceList_entered";

	// Set Step counter
	iDatamachineStep = 4;

	// Debug output
	qDebug() << "Getting To List";

	// Set command
	struCmd.qsCommand = "3";

	// Set cpu
	struCmd.qsCpu = ui.qcbCpus->currentText();

	// Clear answer queue
	if(!qqAswerQueue.isEmpty()) {
		qqAswerQueue.clear();
	}

	// Send command
	clientMachine->sendCommand(&struCmd);

	// Debug output
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu <<
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;

	// Return value
	return;
}

void sGen::if_qsmsGetInstancesByType_entered() {
	// Debug message
	qDebug() << "Entering step: if_qsmsGetInstancesByType_entered";

	// Set Step counter
	iDatamachineStep = 5;

	// Debug output
	qDebug() << "Getting To List";

	// Set command
	struCmd.qsCommand = "4";

	// Set cpu
	struCmd.qsCpu = ui.qcbCpus->currentText();

	// Clear answer queue
	if(!qqAswerQueue.isEmpty()) {
		qqAswerQueue.clear();
	}

	// Send command
	clientMachine->sendCommand(&struCmd);

	// Debug output
```

```
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu << 
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;

	// Return value
	return;
}

void sGen::if_qsmsGetData_entered() {
	// Debug message
	qDebug() << "Entering step: if_qsmsGetData_entered";

	// Set Step counter
	iDatamachineStep = 6;

	// Debug output
	qDebug() << "Getting Project data";

	// Set command
	struCmd.qsCommand = "5";

	// Set cpu
	struCmd.qsCpu = ui.qcbCpus->currentText();

	// Clear answer queue
	if(!qqAswerQueue.isEmpty()) {
		qqAswerQueue.clear();
	}

	// Send command
	clientMachine->sendCommand(&struCmd);

	// Debug output
	qDebug() << "Sending command: " << struCmd.qsProject << struCmd.qsCpu << 
struCmd.qsCommand << struCmd.qsSender << struCmd.qsConHandle << struCmd.qsAttribut;

	// Return value
	return;
}

void sGen::if_qsmfsEndDataMachine_entered() {
	// Debug message
	qDebug() << "Entering step: if_qsmfsEndDataMachine_entered";

	// Set Step counter
	iDatamachineStep = 6;

	// Change caption
	ui.qpbStartBatch->setText("Start Batch");

	// Return value
	return;
}

void sGen::if_qsmfsEndDataMachine_stopped() {
	// Debug message
	qDebug() << "Entering step: if_qsmfsEndDataMachine_stopped";
	qDebug() << "Data Machine wil be stopped now";

	// Return value
	return;
}

// *************** End of Step Functions *****************
```

```
// *************** Answer Check Functions *****************
void sGen::checkProject() {
	// Locals
	QStringList qslProjectList;
	bool ok;

	// Debug message
	qDebug() << "Checking project list";

	// Take Project list from queue and clear queue
	if(!qqAswerQueue.isEmpty()) {
		qslProjectList = qqAswerQueue.dequeue();
		if(!qqAswerQueue.isEmpty()) {
			qqAswerQueue.clear();
		}

		// Remove element count and brackets
		qslProjectList.removeOne("{");
		qslProjectList.removeOne("}");
		qslProjectList.removeFirst();

		// Check for Error
		if(!qslProjectList.isEmpty()) {
			if(qslProjectList.at(0) == "ERROR") {
				// Set project to empty string
				struCmd.qsProject = "";

				// Emit signal
				emit(dataMachineError());

				// Return value
				return;
			}

			// Let user choose project
			else if(qslProjectList.size() > 1) {
				struCmd.qsProject = QInputDialog::getItem(0, "Choose
Project" , "Active Projects", qslProjectList, 0, false, &ok);
				if(struCmd.qsProject.isEmpty()) {
					QMessageBox::warning(0, "sGen", "Please choose a
project");
				}
			}
			else {
				if(!qslProjectList.isEmpty()) {
					struCmd.qsProject = qslProjectList.at(0);
				}
				else {
					QMessageBox::warning(0, "sGen", "Empty Project-
List");

					// Return value
					return;
				}
			}
		}
		else {
			// Set project to empty string
			struCmd.qsProject = "";

			// Emit signal
```

```
                        emit(dataMachineError());

                        // Return value
                        return;
                }
                // Fill projects into combobox
                ui.qcbProjects->addItems(qslProjectList);

                // Emit signal
                emit(projectRecceived());
        }
        else {
                // Messagebox
                QMessageBox::critical(0, "sGen", "No Project list recceived");

                // Emit signal
                emit(dataMachineError());
        }

        // Return value
        return;
}

void sGen::checkCpus() {
        // Locals
        QStringList qslCpuList;
        bool ok;

        // Debug message
        qDebug() << "Checking Cpu list";

        // Take Cpu list from queue and clear queue
        if(!qqAswerQueue.isEmpty()) {
                qslCpuList = qqAswerQueue.dequeue();
                if(!qqAswerQueue.isEmpty()) {
                        qqAswerQueue.clear();
                }

                // Remove element count and brackets
                qslCpuList.removeOne("{");
                qslCpuList.removeOne("}");
                qslCpuList.removeFirst();

                // Check for Error
                if(!qslCpuList.isEmpty()) {
                        if(qslCpuList.at(0) == "ERROR") {
                                // Set cpu to empty string
                                struCmd.qsCpu = "";

                                // Emit signal
                                emit(dataMachineError());

                                // Return value
                                return;
                        }
                }
                // Let user choose cpu
                else if(qslCpuList.size() > 2) {
                        struCmd.qsCpu = QInputDialog::getItem(0, "Choose Cpu" , "Cpu's
found", qslCpuList, 0, false, &ok);
                        if(struCmd.qsProject.isEmpty()) {
                                QMessageBox::warning(0, "sGen", "Please choose a Cpu");
                        }
```

```
		}
		else {
			if(!qslCpuList.isEmpty()) {
				struCmd.qsCpu = qslCpuList.at(0);
			}
			else {
				QMessageBox::warning(0, "sGen", "Empty CPU-List");

				// Return value
				return;
			}
		}

		// Fill cpus into combobox
		ui.qcbCpus->addItems(qslCpuList);

		// Emit signal
		emit(cpusRecceived());
	}
	else {
		// Messagebox
		QMessageBox::critical(0, "sGen", "No Cpu list recceived");

		// Emit signal
		emit(dataMachineError());
	}

	// Return value
	return;
}

void sGen::checkTos() {
	// Locals
	QStringList qslToList1;
	QString qsToName, qsScratch;
	QVariant qvScratch;

	// Debug message
	qDebug() << "Checking TO list";

	// Take To list from queue and clear queue
	if(!qqAswerQueue.isEmpty()) {
		qslToList1 = qqAswerQueue.dequeue();
		if(!qqAswerQueue.isEmpty()) {
			qqAswerQueue.clear();
		}

		// Remove element count and brackets
		qslToList1.removeOne("{");
		qslToList1.removeOne("}");

		if(!qslToList1.isEmpty()) {
			qsScratch = qslToList1.takeFirst();
		}
		else {
			QMessageBox::warning(0, "sGen", "Empty TO-List");

			// Return value
			return;
		}

		// Check for Error
		if(!qslToList1.isEmpty()) {
```

```
                        if(qslToList1.at(0) == "ERROR") {
                                // Emit signal
                                emit(dataMachineError());

                                // Return value
                                return;
                        }
                }

                else {
                        // Emit signal
                        emit(dataMachineError());

                        // Return value
                        return;
                }

                // Fill tos into list widget
                ui.qlwTos->clear();
                ui.qlwTos->addItems(qslToList1);

                // Count Items
                qvScratch = qslToList1.size();
                qsScratch.append( "/" % qvScratch.toString());
                ui.qtlToNbrOfItems->setText(qsScratch);

                // Fill TO-Names into list
                for(int i = 0; i < qslToList1.size(); ++i) {
                        qsToName = qslToList1.at(i);
                        qslToList << qsToName;
                }

                // Emit signal
                emit(tosRecceived());
        }
        else {
                // Messagebox
                QMessageBox::critical(0, "sGen", "No TO list recceived");

                // Emit signal
                emit(dataMachineError());
        }

        // Return value
        return;
}

void sGen::checkInstances() {
        // Locals
        QStringList qslInstanceList, qslTempList1, qslTempList2;
        QString qsGroupName, qsScratch;
        QVariant qvScratch;

        // Debug message
        qDebug() << "Checking instance list";

        // Take To list from queue and clear queue
        if(!qqAswerQueue.isEmpty()) {
                qslInstanceList = qqAswerQueue.dequeue();
                if(!qqAswerQueue.isEmpty()) {
                        qqAswerQueue.clear();
                }
```

```
// Remove element count and brackets
qslInstanceList.removeOne("{");
qslInstanceList.removeOne("}");

if(!qslInstanceList.isEmpty()) {
	qsScratch = qslInstanceList.takeFirst();
}
else {
	QMessageBox::warning(0, "sGen", "Empty Instance-List");

	// Return value
	return;
}

// Check for Error
if(!qslInstanceList.isEmpty()) {
	if(qslInstanceList.at(0) == "ERROR") {
		// Emit signal
		emit(dataMachineError());

		// Return value
		return;
	}
}

else {
	// Emit signal
	emit(dataMachineError());

	// Return value
	return;
}

// Copy input list
qslpTInstanceList << qslInstanceList;

// Fill instances into list widget
ui.qlwInstances->clear();
ui.qlwInstances->addItems(qslInstanceList);

// Count Items
qvScratch = qslInstanceList.size();
qsScratch.append( "/" % qvScratch.toString());
ui.qtlInstancesNbrOfItems->setText(qsScratch);

// Get groupnames from list
for(int i = 0; i < qslInstanceList.size(); ++i) {
	qslTempList1 << qslInstanceList.at(i).section(':', 0, 0);
}

// Remove duplicates
qslTempList1.removeDuplicates();

// Get first group name
qsGroupName = qslInstanceList.at(0).section(':', 0, 0);

// Fill data int hash
for(int y = 0; y < qslTempList1.size(); ++y) {
	for(int i = 0; i < qslInstanceList.size(); ++i) {
		// Get group name if neccessary
		if(qslInstanceList.at(i).contains(qslTempList1.at(y))) {
			// Put instance into temp list
			qslTempList2 << qslInstanceList.at(i);
```

```
                        }
                    }
                    // Move TO's into hash
                    qhDmsNames.insert(qslTempList1.at(y), qslTempList2);

                    // Clear temp list
                    qslTempList2.clear();
                }

                // Emit signal
                emit(instancesRecceived());
        }
        else {
                // Messagebox
                QMessageBox::critical(0, "sGen", "No instance list recceived");

                // Emit signal
                emit(dataMachineError());
        }

        // Return value
        return;
}

void sGen::checkInstancesByType() {
        // Locals
        QStringList qslInstanceList, qslTempList1, qslTempList2;
        QString qsToName, qsScratch, qsScratch1, qsScratch2;
        QVariant qvScratch;

        // Debug message
        qDebug() << "Checking instance by type list";

        // Take To list from queue and clear queue
        if(!qqAswerQueue.isEmpty()) {
                qslInstanceList = qqAswerQueue.dequeue();
                if(!qqAswerQueue.isEmpty()) {
                        qqAswerQueue.clear();
                }

                // Remove element count and brackets
                qslInstanceList.removeOne("{");
                qslInstanceList.removeOne("}");
                qsScratch = qslInstanceList.takeFirst();

                // Check for Error
                if(!qslInstanceList.isEmpty()) {
                        if(qslInstanceList.at(0) == "ERROR") {
                                // Emit signal
                                emit(dataMachineError());

                                // Return value
                                return;
                        }
                }

                else {
                        // Emit signal
                        emit(dataMachineError());

                        // Return value
                        return;
                }
```

```
                // Fill instances into list widget
                ui.qlwInstances->addItems(qslInstanceList);

                // Count Items
                qvScratch = qslInstanceList.size();
                qsScratch.append( "/" % qvScratch.toString());
                ui.qtlInstancesNbrOfItems->setText(qsScratch);

                // Sort TO's by Type
                for(int y = 0; y < qslToList.size(); ++y) {
                        for(int i = 0; i < qslInstanceList.size(); ++i) {

                                if(qslInstanceList.at(i).contains(qslToList.at(y))) {
                                        qsScratch1 = qslInstanceList.at(i).section(':', -1,
-1);
                                        qsScratch2 = qslInstanceList.at(i);
                                        qsScratch2 = qsScratch2.remove(qsScratch1,
Qt::CaseInsensitive);
                                        qslTempList1 << qsScratch2;
                                }
                        }
                        // Move TO's into hash
                        qhToTypes.insert(qslToList.at(y), qslTempList1);
                        qslTempList1.clear();
                }

                // Sort TO's by Type and Group
                qslTempList2 = qhDmsNames.keys();

                for(int x = 0; x < qslTempList2.size(); ++x) {
                        for(int y = 0; y < qslToList.size(); ++y) {
                                for(int i = 0; i < qslInstanceList.size(); ++i) {

                                        if(qslInstanceList.at(i).contains(qslToList.at(y))
&& qslInstanceList.at(i).contains(qslTempList2.at(x))) {
                                                qsScratch1 =
qslInstanceList.at(i).section(':', -1, -1);
                                                qsScratch2 = qslInstanceList.at(i);
                                                qsScratch2 = qsScratch2.remove(qsScratch1,
Qt::CaseInsensitive);
                                                qslTempList1 << qsScratch2;
                                        }
                                }
                                // Move TO's into hash
                                qhGroup.insert(qslToList.at(y), qslTempList1);
                                qslTempList1.clear();
                        }
                        // Move hash into hash
                        qhDmsNamesByGroupAndType.insert(qslTempList2.at(x), qhGroup);
                }



                // Emit signal
                emit(instancesByTypeRecceived());
        }
        else {
                // Messagebox
                QMessageBox::critical(0, "sGen", "No instance by type list recceived");

                // Emit signal
                emit(dataMachineError());
```

```
	}

	// Return value
	return;
}

void sGen::checkData() {
	// Locals
	QStringList qslDataList;
	QString qsScratch;
	QVariant qvScratch;

	// Debug message
	qDebug() << "Checking project data";

	// Take To list from queue and clear queue
	if(!qqAswerQueue.isEmpty()) {
		qslDataList = qqAswerQueue.dequeue();
		if(!qqAswerQueue.isEmpty()) {
			qqAswerQueue.clear();
		}

		// Remove element count and brackets
		qslDataList.removeOne("{");
		qslDataList.removeOne("}");
		qsScratch = qslDataList.takeFirst();

		// Check for Error
		if(!qslDataList.isEmpty()) {
			if(qslDataList.at(0) == "ERROR") {
				// Emit signal
				emit(dataMachineError());

				// Return value
				return;
			}
		}
		else {
			// Emit signal
			emit(dataMachineError());

			// Return value
			return;
		}

		// Copy input list
		qslpTDataList << qslDataList;

		// Fill instances into list widget
		ui.qlwData->clear();
		ui.qlwData->addItems(qslDataList);

		// Count Items
		qvScratch = qslDataList.size();
		qsScratch.append( "/" % qvScratch.toString());
		ui.qtlDataNbrOfItems->setText(qsScratch);

		// Emit signal
		emit dataRecceived();
	}
	else {
		// Messagebox
		QMessageBox::critical(0, "sGen", "No data list recceived");
```

```
                // Emit signal
                emit(dataMachineError());
        }

        // Return value
        return;
}

void sGen::checkComment() {
        // Locals
        QStringList qslCommentList;
        QString qsScratch;
        QVariant qvScratch;

        // Debug message
        qDebug() << "Checking ";

        // Take To list from queue and clear queue
        if(!qqAswerQueue.isEmpty()) {
                qslCommentList = qqAswerQueue.dequeue();
                if(!qqAswerQueue.isEmpty()) {
                        qqAswerQueue.clear();
                }

                // Remove element count and brackets
                qslCommentList.removeOne("{");
                qslCommentList.removeOne("}");
                qsScratch = qslCommentList.takeFirst();

                // Check for Error
                if(!qslCommentList.isEmpty()) {
                        if(qslCommentList.at(0) == "ERROR") {
                                // Emit signal
                                emit(dataMachineError());

                                // Return value
                                return;
                        }
                }
                else {
                        // Emit signal
                        emit(dataMachineError());

                        // Return value
                        return;
                }


                // Copy input list
                qslpTCommentList << qslCommentList;

                // Fill instances into list widget
                ui.qlwComments->clear();
                ui.qlwComments->addItems(qslCommentList);

                // Count Items
                qvScratch = qslCommentList.size();
                qsScratch.append( "/" % qvScratch.toString());
                ui.qtlCommentsNbrOfItems->setText(qsScratch);

                // Emit signal
                emit dataRecceived();
```

```
	}
	else {
		// Messagebox
		QMessageBox::critical(0, "sGen", "No comment list recceived");

		// Emit signal
		emit(dataMachineError());
	}

	// Return value
	return;
}

//! This Function checks the answer recceived with the getAttributeComment-Command
(cmd nbr. : 7).
    /*!
      \param - none
      \return void
    */
void sGen::checkAttributeComment() {
	// Locals
	QStringList qslCommentList;
	QString qsScratch;
	QVariant qvScratch;

	// Debug message
	qDebug() << "Checking Attribute comments...";

	// Take To list from queue and clear queue
	if(!qqAswerQueue.isEmpty()) {
		qslCommentList = qqAswerQueue.dequeue();
		if(!qqAswerQueue.isEmpty()) {
			qqAswerQueue.clear();
		}

		// Remove element count and brackets
		qslCommentList.removeOne("{");
		qslCommentList.removeOne("}");
		qsScratch = qslCommentList.takeFirst();

		// Check for Error
		if(!qslCommentList.isEmpty()) {
			if(qslCommentList.at(0) == "ERROR") {
				// Emit signal
				emit(dataMachineError());

				// Return value
				return;
			}
		}
		else {
			// Emit signal
			emit(dataMachineError());

			// Return value
			return;
		}

		// Copy input list
		qslpTAttributeCommentList << qslCommentList;

		// Fill instances into list widget
		ui.qlwAttributeComments->clear();
```

```
		ui.qlwAttributeComments->addItems(qslCommentList);

		// Count Items
		qvScratch = qslCommentList.size();
		qsScratch.append( "/" % qvScratch.toString());
		ui.qtlAttributeCommentsNbrOfItems->setText(qsScratch);

		// Emit signal
		// emit dataRecceived();
	}
	else {
		// Messagebox
		QMessageBox::critical(0, "sGen", "No comment list recceived");

		// Emit signal
		emit(dataMachineError());
	}

	// Return value
	return;
}

//! This Function checks the answer recceived with the getProjectPath-Command (cmd
nbr. : 8).
    /*!
      \param - none
      \return void
    */
void sGen::checkPath() {
	// Locals
	QStringList qslPathList;
	QString qsScratch, qsScratch2;
	QVariant qvScratch;

	// Debug message
	qDebug() << "Checking Project path...";

	// Take To list from queue and clear queue
	if(!qqAswerQueue.isEmpty()) {
		qslPathList = qqAswerQueue.dequeue();
		if(!qqAswerQueue.isEmpty()) {
			qqAswerQueue.clear();
		}

		// Remove element count and brackets
		qslPathList.removeOne("{");
		qslPathList.removeOne("}");
		qsScratch = qslPathList.takeFirst();

		// Check for Error
		if(!qslPathList.isEmpty()) {
			if(qslPathList.at(0) == "ERROR") {
				// Emit signal
				emit(dataMachineError());

				// Return value
				return;
			}
		}
		else {
			// Emit signal
			emit(dataMachineError());
```

```
                        // Return value
                        return;
                }

                // Fill instances into list widget
                ui.qlwPath->clear();
                ui.qlwPath->addItems(qslPathList);


                // Fill instances into line edit
                ui.qlePlcPath->clear();
                qsScratch2 = qslPathList.at(0);
                qsScratch2.replace("cfg/", "pcd/");
                ui.qlePlcPath->setText(qsScratch2);

                // Count Items
                qvScratch = qslPathList.size();
                qsScratch.append( "/" % qvScratch.toString());
                ui.qtlPathNbrOfItems->setText(qsScratch);

                // Emit signal
                // emit dataRecceived();
        }
        else {
                // Messagebox
                QMessageBox::critical(0, "sGen", "No comment list recceived");

                // Emit signal
                emit(dataMachineError());
        }

        // Return value
        return;

}

//! This Function checks the answer recceived with the getDataForTree-Command (cmd
nbr. : 9).
    /*!
      \param - none
      \return void
    */
void sGen::checkDataForTree() {
        // Locals
        QStringList qslDataList;
        QString qsScratch, qsScratch2;
        QVariant qvScratch;

        // Debug message
        qDebug() << "Checking data for tree...";

        // Take data-list from queue and clear queue
        if(!qqAswerQueue.isEmpty()) {
                qslDataList = qqAswerQueue.dequeue();
                if(!qqAswerQueue.isEmpty()) {
                        qqAswerQueue.clear();
                }

                // Remove element count and brackets
                qslDataList.removeOne("{");
                qslDataList.removeOne("}");

                if(!qslDataList.isEmpty()) {
```

```
                qsScratch = qslDataList.takeFirst();
        }
        else {
                QMessageBox::warning(0, "sGen", "Empty DataForTree-List");

                // Return value
                return;
        }

        // Check for Error
        if(!qslDataList.isEmpty()) {
                if(qslDataList.at(0) == "ERROR") {
                        // Emit signal
                        emit(dataMachineError());

                        // Return value
                        return;
                }
        }
        else {
                // Emit signal
                emit(dataMachineError());

                // Return value
                return;
        }

        // Count Items
        qvScratch = qslDataList.size();
        if(qvScratch == qsScratch.toInt() ) {
                // Create tree
                if(!createTrees(qslDataList)) {
                        // Messagebox
                        QMessageBox::critical(0, "sGen/DataMachine", "Creating
Data-Tree failed");

                        // Emit signal
                        emit(dataMachineError());

                        // Return value
                        return;
                }
                else {
                        // Get the text out of the current item
                        ui.qtvsGenTree->setModel(qhTrees.value("Data-Tree"));
                        ui.qtvsGenTree->show();

                        // Emit signal
                        emit dataRecceived();
                }
        }
        else {
                // Messagebox
                QMessageBox::critical(0, "sGen", "No proper data-list recceived");

                // Emit signal
                emit(dataMachineError());

                // Return value
                return;
        }
}
else {
```

```
                // Messagebox
                QMessageBox::critical(0, "sGen", "No proper data-list recceived");

                // Emit signal
                emit(dataMachineError());
        }

        // Return value
        return;

}

void sGen::dataMachineErrorHandler() {
        // Debug message
        qDebug() << "Data-Machine Error";

        // Return value
        return;
}

// ************ End of Answer Check Functions *************
```

Appendix F V Source qcdatatree.h

```
#ifndef QCDATATREE_H
#define QCDATATREE_H

#include <QStandardItemModel>
#include <QtGui>                                    // QT header

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

class qcdatatree : public QStandardItemModel {
       Q_OBJECT

public:
       qcdatatree(QObject *parent);
       ~qcdatatree();

       bool bLoadData(QStringList qsInDataForTree);

private:
       void initialize();

       QList<QStandardItem*> qlDmsNameItems;
       QList<QStandardItem*> qlDataItems;
       QList<QStandardItem*> qlParentItems;

       QStandardItem *qsiDmsNameItem;
       QStandardItem *qsiDataItem;
       QStandardItem *qsiParentItem;
};

#endif // QCDATATREE_H
```

Appendix F VI Source code qcdatatree.cpp

```
/*****************************************************************************
 Implementation of the DataTree of the Saia Code Generator
 ---------------------------------------------------

 Date         : 12.09.2012
 File       : qcdatatree.cpp
 Author             : Thomas Gasser
 ©          : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's Saia Code Generator.
/*!
  This is the DataTree-Class of TG's Saia Code Generator
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "qcdatatree.h"                 // Project header
/****************** Includes ************************/


// Constructor / Destructor
//! This is the Constructor of the sGen application.
    /*!
         Here the on initialization stuff is made
    */
qcdatatree::qcdatatree(QObject *parent): QStandardItemModel(parent) {
       // Inizialize tree
       initialize();
}

//! This is the Destructor of the qcdatatree-class
    /*!
         Here some stuff is made just before the class is destroyed
    */
qcdatatree::~qcdatatree() {

}

//! This is the Function initialize the sGen-Data-Tree
    /*!
         This function is used to initialize the sGen-Data-Tree
         \return An void.
    */
void qcdatatree::initialize() {
       // Set Header Labels
       setHorizontalHeaderLabels(QStringList() << tr("sGen-Data-Tree"));

       // Set alignments
       for (int column = 1; column < columnCount(); ++column) {
        horizontalHeaderItem(column)-
>setTextAlignment(Qt::AlignVCenter|Qt::AlignRight);
       }
}

//! This is the Function loads the sGen-Data-Tree with the data
```

```
    /*!
        This function is used to load the sGen-Data-Tree with the data received from
pTool
        \return A boolean.
    */

bool qcdatatree::bLoadData(QStringList qsInDataForTree) {

       // Locals
       bool bRetValue = false;
       QStringList qslDmsNameItems;
       QString qsDmsName, qsChild, qsScratch;
       int iCount = 0;

       if(qsInDataForTree.isEmpty()) {
              QMessageBox::warning(0, "sGen/DataTree", "There is no data in the data-
list");
       }
       else {
              // Create Items
              qsiParentItem = new QStandardItem();

              // Setup Progress Dialog
              QProgressDialog *qpdProgress = new QProgressDialog("", "Cancel", 0,
qsInDataForTree.size());
              qpdProgress->setWindowModality(Qt::WindowModal);
              qpdProgress->setLabelText("Building Data-Tree...");
              qpdProgress->setValue(0);
              qpdProgress->show();

              // Clear tree
              clear();

              // Initialize tree
              initialize();

              // Create Tree
              qlDmsNameItems.clear();

              while(!qsInDataForTree.isEmpty()) {
                     // Get List-Element
                     qsScratch = qsInDataForTree.takeFirst();
                     iCount++;

                     // Set progress bar
                     qpdProgress->setValue(iCount);

                     // Check if it is a key
                     if(qsScratch.startsWith("Key: ")) {
                            // Reset DMS-Name-Parent
                            qlParentItems.clear();
                            qlParentItems.append(invisibleRootItem());

                            // Remove keyword "Key: "
                            qsDmsName = qsScratch.remove("Key: ");

                            // Split String
                            qslDmsNameItems = qsDmsName.split(":");

                            for(int i = 0; i < qslDmsNameItems.size(); ++i) {
                                   // Check if Item exists
                                   qsScratch = qslDmsNameItems.at(i);
```

```
                                    qsiDmsNameItem = new
QStandardItem(qslDmsNameItems.at(i));

                                    // Insert DMS-Name Items
                                    if(!qlParentItems.isEmpty()) {
                                           if(qlParentItems.at(0) != qsiDmsNameItem-
>parent()) {
                                                  // Check parent text
                                                  QString qsTest01 = qlParentItems.at(0)-
>text();
                                                  int iRowCount = qlParentItems.at(0)-
>rowCount();
                                                  bool bMatch = false;

                                                  if(iRowCount > 0) {

                                                         for(int y = 0; y < iRowCount;
++y){
                                                                QString qsTest02 =
qlParentItems.at(0)->child(y, 0)->text();
                                                                QString       qsTest03 =
qslDmsNameItems.at(i);

                                                                if(qsTest02 == qsTest03)
{
                                                                       bMatch = true;
                                                                       qsiDmsNameItem =
qlParentItems.at(0)->child(y, 0);
       qlParentItems.clear();
       qlParentItems.append(qsiDmsNameItem);
                                                                       break;
                                                                }

                                                         }
                                                         if(!bMatch) {
                                                                // Insert item
                                                                qlParentItems.at(0)-
>appendRow(qsiDmsNameItem);
                                                                qlParentItems.clear();
       qlParentItems.append(qsiDmsNameItem);
                                                         }
                                                         else {
                                                                qlParentItems.clear();
       qlParentItems.append(qsiDmsNameItem);
                                                         }
                                                  }
                                                  else {
                                                         // Insert item
                                                         QString qsTest04 =
qlParentItems.at(0)->text();
                                                         qlParentItems.at(0)-
>appendRow(qsiDmsNameItem);
                                                         qlParentItems.clear();
       qlParentItems.append(qsiDmsNameItem);
                                                  }
                                           }
                                           else {
                                                  qlParentItems.clear();
```

```
                                        qlParentItems.append(qsiDmsNameItem);
                                }
                        }
                        else {
                                qlParentItems.clear();
                                qlParentItems.append(qsiDmsNameItem);
                        }
                }
        }
        else {
                // Create new child item
                qsiDmsNameItem = new QStandardItem(qsScratch);

                // Append Child to the last parent
                qlParentItems.at(0)->appendRow(qsiDmsNameItem);
                qlDataItems.clear();
                qlDataItems.append(qsiDmsNameItem);

                // Get Data-Element from list
                if(!qsInDataForTree.isEmpty()) {
                        qsScratch = qsInDataForTree.takeFirst();
                }

                // increment counter
                iCount++;

                // Set progress bar
                qpdProgress->setValue(iCount);

                // Create new child item
                qsiDataItem = new QStandardItem(qsScratch);
                qlDataItems.at(0)->appendRow(qsiDataItem);
                qlDataItems.clear();
        }
    }
    // Delete objects
    delete qpdProgress;

    // Sortieren
    sort(0, Qt::AscendingOrder);

    // Return value
    return bRetValue;
  }
}
```

Appendix F VII Source code rxpGenerator.cpp

```
/*****************************************************************************
 Implementation of the rxp-Symbol-File Generator
 -----------------------------------------------

 Date         : 11.05.2012
 File        : rxpGenerator.cpp
 Author              : Thomas Gasser
 ©           : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version     : 1.0.0.0
 Changes      :

*****************************************************************************/

// ******************** Includes ************************
#include "stdafx.h"                     // Precompiled headers
#include "sgen.h"                       // Project header
// ***************** End of Includes *********************


// ***************** Step Functions *********************
//! This Slot is triggered by the Generate RXP-Button in the GUI.
    /*!
      \param - None.
        \return void
    */
void sGen::on_qpbGenerateRxp_clicked() {
       // Locals
       bool bTest = false;

       // dll einbinden
       generatorLib = new sGenLib();

       // Generator State machine
       qsmGeneratorMachine = new QStateMachine(this);

       // Steps of the RXP Generator machine
       qsmsOpenRxpFile = new QState();
       qsmsWriteRxpHeader = new QState();
       qsmsWriteSymbols = new QState();
       qsmsCloseRxpFile = new QState();
       qsmfsEndGeneratorMachine = new QFinalState();

       // Add steps to the data machine
       qsmGeneratorMachine->addState(qsmsOpenRxpFile);
       qsmGeneratorMachine->addState(qsmsWriteRxpHeader);
       qsmGeneratorMachine->addState(qsmsWriteSymbols);
       qsmGeneratorMachine->addState(qsmsCloseRxpFile);
       qsmGeneratorMachine->addState(qsmfsEndRxpGeneratorMachine);

       // Set initial step of the data machine
       qsmGeneratorMachine->setInitialState(qsmsOpenRxpFile);

       // Add transitions to the states of the data machine
       qsmsOpenRxpFile->addTransition(this, SIGNAL(rxpFileOpened()),
qsmsWriteRxpHeader);
       qsmsWriteRxpHeader->addTransition(this, SIGNAL(rxpHeaderWritten()),
qsmsWriteSymbols);
       qsmsWriteSymbols->addTransition(this, SIGNAL(rxpSymbolsWritten()),
qsmsCloseRxpFile);
```

```
	qsmsCloseRxpFile->addTransition(this, SIGNAL(rxpFileClosed()),
qsmfsEndRxpGeneratorMachine);

	// Establish connections of the states of the data machine
	bTest = connect(qsmsOpenRxpFile, SIGNAL(entered()), this,
SLOT(if_qsmsOpenRxpFile_entered()));
	bTest = connect(qsmsWriteRxpHeader, SIGNAL(entered()), this,
SLOT(if_qsmsWriteRxpHeader_entered()));
	bTest = connect(qsmsWriteSymbols, SIGNAL(entered()), this,
SLOT(if_qsmsWriteSymbols_entered()));
	bTest = connect(qsmsCloseRxpFile, SIGNAL(entered()), this,
SLOT(if_qsmsCloseRxpFile_entered()));
	bTest = connect(qsmfsEndRxpGeneratorMachine, SIGNAL(entered()), this,
SLOT(if_qsmfsEndRxpGeneratorMachine_entered()));

	// Start machine
	qsmGeneratorMachine->start();
}

//! This is the state-entered-Function for the open rxp-file-state.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmsOpenRxpFile_entered() {
	// Locals
	QString qsProjectPath;

	// Debug message
	qDebug() << "Entering step: if_qsmsOpenRxpFile_entered";

	// Set path
	qsProjectPath = ui.qlePlcPath->text();

	if(qsProjectPath.isEmpty() || qsProjectPath.isNull()) {
		qsProjectPath = QFileDialog::getExistingDirectory(this, tr("Choose the
project path"), "c:/PromosNT/proj/", QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);
		ui.qlePlcPath->setText(qsProjectPath + "/pcd/");
		qsProjectPath = ui.qlePlcPath->text();
	}

	// Set path and filename
	if(qsProjectPath.isEmpty()) {
		QMessageBox::warning(this, "sGen", "Please Coose a project");
		return;
	}
	else {
		// Set input path
		if(ui.qleRxpFileName->text().isEmpty()){
			QMessageBox::warning(this, "sGen", "Please insert a filename");
			return;
		}
		else {
			// assemble complete path
			qsNameOfOutputFile = qsProjectPath % ui.qcbCpus->currentText() %
"/" % ui.qleRxpFileName->text();

			// Set filename
			qfOutput.setFileName(qsNameOfOutputFile);

			// Open / create new file
			qtsOutput.setDevice(&qfOutput);
```

```
                    if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
                          QMessageBox::warning(0, "sGen", "Error while creating the
rxp-file");
                    }
                    else {
                          // Emite Signal
                          emit rxpFileOpened();
                    }
             }
      }

      // Return value
      return;
}

//! This is the state-entered-Function for the write-header-state.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmsWriteRxpHeader_entered() {
      // Locals
      QString qsScratch1;

      // Debug message
      qDebug() << "Entering step: if_qsmsWriteRxpHeader_entered";

      // Get rxp header
      qsScratch1 = qsSettings->value("SymbolHeader/Header", "No entry").toString();

      // Write header
      qtsOutput << qsScratch1 << "\n";

      // emit signal
      emit rxpHeaderWritten();

      // Return value
      return;
}

//! This is the state-entered-Function for the write-symbols-state.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmsWriteSymbols_entered() {
      // Locals
      QString qsScratch1, qsScratch2, qsScratch3, qsScratch4, qsScratch5, qsData,
qsAddress;
      QStringList qslTempList1, qslTempList2, qslTempList3, qslTempList4,
qslTempList5;

      // Debug message
      qDebug() << "Entering step: if_qsmsWriteSymbols_entered";

      // Assemble Symbols
      for(int i = 0; i < qslpTInstanceList.size(); ++i) {
            // Get instance name
            qsScratch5 = qslpTInstanceList.at(i);

            // find elements of this instance
            qslTempList1 = qslpTDataList.filter(qsScratch5, Qt::CaseInsensitive);
            qslTempList2 = qslpTCommentList.filter(qsScratch5, Qt::CaseInsensitive);
```

```
		qslTempList3 = qslpTAttributeCommentList.filter(qsScratch5,
Qt::CaseInsensitive);

		//
		qslTempList4.clear();
		for(int y = 0; y < qslTempList1.size(); ++y) {
			qsScratch3 = qslTempList1.at(y).section(": (", 0, 0);
			qslTempList4 << qsScratch3;
		}

		// Create symbols
		qslTempList5 = generatorLib->makeSymbol(qslTempList4);

		// assemble the string
		qslTempList4.clear();
		for(int y = 0; y < qslTempList1.size(); ++y) {
			// Assemble Symbolname, Datatyp and Address
			qsScratch2 = qslTempList1.at(y).section(":", -2, -1);
			qsScratch2 = qsScratch2.remove(")");
			qsScratch2 = qsScratch2.replace(": (", ":=");
			qsScratch2 = qsScratch2.remove("( ");
			qsScratch2 = qsScratch2.remove(" ");

			qsData = qsScratch2.section(":=", 1, 1);
			qsData = qsData.section(".", 0, 0);

			qsAddress = qsScratch2.section(":=", 0, 0);

			qsScratch2 = qsAddress % ":=" % qsData;

			qsScratch2 = qsScratch2.insert(1, ";");
			qsScratch2 = qsScratch2.prepend(";");

			qsScratch3 = qsScratch2.prepend(qslTempList5.at(y));

			// Assemble Comments
			qsScratch2 = qslTempList2.at(0).section(":", -1, -1);
			qsScratch2.remove("(");
			qsScratch2.remove(")");
			qsScratch4 = qslTempList3.at(y).section(":", -1, -1);
			qsScratch4.remove("(");
			qsScratch4.remove(")");
			qsScratch2.append(" / ");
			qsScratch2.append(qsScratch4);
			qsScratch2.append("; 1");
			qsScratch3.append(";");
			qsScratch3.append(qsScratch2);

			qslTempList4 << qsScratch3;
		}
		// Write lines to file
		for(int i = 0; i < qslTempList4.size(); ++i) {
			qsScratch1 = qslTempList4.at(i);
			qtsOutput << qsScratch1 << "\n";
		}
	}

	// emit signal
	emit rxpSymbolsWritten();

	// Return value
	return;
}
```

```
//! This is the state-entered-Function for the close-file-state.
    /*!
      \param - None.
         \return void
    */
void sGen::if_qsmsCloseRxpFile_entered() {
       // Debug message
       qDebug() << "Entering step: if_qsmsCloseRxpFile_entered";

       // Debug output
       qDebug() << "closing rxp file...";

       // Close file
       qfOutput.close();

       // emit signal
       emit rxpFileClosed();

       // Return value
       return;
}

//! This is the state-entered-Function for the final-state of the rxp generator
machine.
    /*!
      \param - None.
         \return void
    */
void sGen::if_qsmfsEndRxpGeneratorMachine_entered() {
       // Debug message
       qDebug() << "Entering step: if_qsmfsEndGeneratorMachine_entered";

       // emit signal
       emit rxpGenerated();

       // Return value
       return;
}
// *************** End of Step Functions ******************
```

Appendix F VIII Source code vmGenerator.cpp

```
/*****************************************************************************
 Implementation of the VM Code Generator
 ---------------------------------------

 Date         : 09.11.2012
 File        : svmGenerator.cpp
 Author              : Thomas Gasser
 ©           : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version     : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's VM Code Generator.
/*!
  This is TG's VM Code Generator. The VM code is genarated as COB to esure that there
are no siede effects
*/

/****************** Includes ************************/
#include "stdafx.h"                      // Precompiled headers
#include "sgen.h"                        // Project header
/****************** Includes ************************/

/******************* Defines ************************/
#define QT_USE_FAST_CONCATENATION
/******************* Defines ************************/


// ***************** Step Functions *********************
//! This Slot is triggered by the Generate RXP-Button in the GUI.
    /*!
      \param - None.
         \return void
    */
void sGen::on_qpbGenerateVm_clicked() {
       // Locals
       bool bTest = false;

       // dll einbinden
       generatorLib = new sGenLib();

       // Generator State machine
       qsmGeneratorMachine = new QStateMachine(this);

       // Steps of the RXP Generator machine
       qsmsOpenVmFile = new QState();
       qsmsWriteVmHeader = new QState();
       qsmsWriteVmSymbols = new QState();
       qsmsCloseVmFile = new QState();
       qsmfsEndVmGeneratorMachine = new QFinalState();

       // Add steps to the data machine
       qsmGeneratorMachine->addState(qsmsOpenVmFile);
       qsmGeneratorMachine->addState(qsmsWriteVmHeader);
       qsmGeneratorMachine->addState(qsmsWriteVmSymbols);
       qsmGeneratorMachine->addState(qsmsCloseVmFile);
       qsmGeneratorMachine->addState(qsmfsEndVmGeneratorMachine);
```

```
	// Set initial step of the data machine
	qsmGeneratorMachine->setInitialState(qsmsOpenVmFile);

	// Add transitions to the states of the data machine
	qsmsOpenVmFile->addTransition(this, SIGNAL(vmFileOpened()), qsmsWriteVmHeader);
	qsmsWriteVmHeader->addTransition(this, SIGNAL(vmHeaderWritten()),
qsmsWriteVmSymbols);
	qsmsWriteVmSymbols->addTransition(this, SIGNAL(vmSymbolsWritten()),
qsmsCloseVmFile);
	qsmsCloseVmFile->addTransition(this, SIGNAL(vmFileClosed()),
qsmfsEndVmGeneratorMachine);

	// Establish connections of the states of the data machine
	bTest = connect(qsmsOpenVmFile, SIGNAL(entered()), this,
SLOT(if_qsmsOpenVmFile_entered()));
	bTest = connect(qsmsWriteVmHeader, SIGNAL(entered()), this,
SLOT(if_qsmsWriteVmHeader_entered()));
	bTest = connect(qsmsWriteVmSymbols, SIGNAL(entered()), this,
SLOT(if_qsmsWriteVmSymbols_entered()));
	bTest = connect(qsmsCloseVmFile, SIGNAL(entered()), this,
SLOT(if_qsmsCloseVmFile_entered()));
	bTest = connect(qsmfsEndVmGeneratorMachine, SIGNAL(entered()), this,
SLOT(if_qsmfsEndVmGeneratorMachine_entered()));

	// Start machine
	qsmGeneratorMachine->start();
}

//! This is the state-entered-Function for the open vm-file-state.
    /*!
      \param - None.
        \return void
    */

void sGen::if_qsmsOpenVmFile_entered() {
	// Locals
	QString qsProjectPath;

	// Debug message
	qDebug() << "Entering step: if_qsmsOpenVmFile_entered";

	// Set path
	qsProjectPath = ui.qlePlcPath->text();

	if(qsProjectPath.isEmpty() || qsProjectPath.isNull()) {
		qsProjectPath = QFileDialog::getExistingDirectory(this, tr("Choose the
project path"), "c:/PromosNT/proj/", QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);
		ui.qlePlcPath->setText(qsProjectPath + "/pcd/");
		qsProjectPath = ui.qlePlcPath->text();
	}

	// Set path and filename
	if(qsProjectPath.isEmpty()) {
		QMessageBox::warning(this, "sGen", "Please choose a project");
		return;
	}
	else {
		// Set input path
		if(ui.qleVmFileName->text().isEmpty()){
			QMessageBox::warning(this, "sGen", "Please insert a filename for
the vm-file");
			return;
```

```
		}
		else {
			// assemble complete path
			qsNameOfOutputFile = qsProjectPath % ui.qcbCpus->currentText() %
"/" % ui.qleVmFileName->text();

			// Set filename
			qfOutput.setFileName(qsNameOfOutputFile);

			// Open / create new file
			qtsOutput.setDevice(&qfOutput);
			if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
				QMessageBox::warning(0, "sGen", "Error while creating the
vm-file");

				// Stop machine
				qsmGeneratorMachine->stop();
			}
			else {
				// Emite Signal
				emit vmFileOpened();
			}
		}
	}
	// Return value
	return;
}

//! This is the state-entered-Function for the write-header-state.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmsWriteVmHeader_entered() {
	// Locals
	QString qsScratch;

	// Debug message
	qDebug() << "Entering step: if_qsmsWriteVmHeader_entered";

	// Fill Header data into structure
	qsSettings->beginGroup("VmTemplates");
	struHeader.qsHeaderTemplate = qsSettings->value("srcHeader", "No
entry").toString();
	struHeader.qsPcdType = generatorLib->pcdTyp(ui.qlePlcPath->text(), ui.qcbCpus-
>currentText());
	struHeader.qsPgVersion = generatorLib->pgVersion(ui.qlePlcPath->text(),
ui.qcbCpus->currentText());
	struHeader.qsBlockTyp = "COB";
	struHeader.qsStationNumber = generatorLib->stationNumber(ui.qlePlcPath->text(),
ui.qcbCpus->currentText());
	struHeader.qsOutputFile  = qsNameOfOutputFile;

	// Write Header
	if(generatorLib->writeHeader(&struHeader) != 0) {
		// Show warning
		QMessageBox::warning(0, "sGen", "Error while writing header, machine will
be stopped");

		// Stop machine
		qsmGeneratorMachine->stop();
	}
```

```
	// emit signal
	emit vmHeaderWritten();
	qsSettings->endGroup();

	// Return value
	return;
}

//! This is the state-entered-Function for the write-symbols-state.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmsWriteVmSymbols_entered() {
	// Locals
	QString qsScratch, qsScratch1, qsScratch2, qsScratch3, qsScratch4, qsParent,
qsDmsName, qsComment, qsAddress, qsGeneratorTemplate, qsOutputString, qsDbItems1,
qsDbItems2, qsDbItems3, qsDbItems4, qsDbItems5, qsDbItems6, qsDmsNameLastPart;
	QStringList qslDmsNames, qslDatapointData, qslKeys, qslLtoDbAdresses,
qslVmDbAdresses, qslLtoInputAddresses, qslVmInputAddresses;
	QList<QStandardItem*> qlResult, qlResultAddress;
	QStandardItem *qsiResult, *qsiResultAddress;
	QStandardItem *qsiChild1, *qsiChild2, *qsiChild3, *qsiChildAddress;
	int iRowCount = 0, iRowCount2 = 0, iRowCount3 = 0, iDbIndex = 0, a = 0,
iKeyCount;
	QVariant qvScratch1, qvScratch2, qvScratch3;
	QStringList qslParIn, qslParOut, qslOutputData;

	// Debug message
	qDebug() << "Entering step: if_qsmsWriteVmSymbols_entered";


	////////////////////////////////////////////////////////////////////////////
////////////
	// Find LTO's
	qlResult = qhTrees.value("Data-Tree")->findItems("LTO01", Qt::MatchRecursive |
Qt::MatchContains);

	// First Run: Get all LTO's DMS-Names
	if(!qlResult.isEmpty()) {
		// Get DMS-Names
		qslDmsNames = generatorLib->getDmsNames(qlResult);
	}
	else {
		// Debug output
		qDebug() << "First run: Getting LTO's failed";

		// Return value
		return;
	}
	////////////////////////////////////////////////////////////////////////////
////////////


	////////////////////////////////////////////////////////////////////////////
////////////
	// Second run: Get DB data and elements of the LTO's
	if(!qlResult.isEmpty()) {
		// Clear List
		qlHashes2.clear();

		// Get data
		for(int i = 0; i < qlResult.size(); ++i) {
```

```
                        // Create LTO-Hash
                        qlHashes2 << generatorLib->createLtoHash(qlResult.at(i)->parent()-
>parent(), qslDmsNames.at(i));
                }
        }
        else {
                // Debug output
                qDebug() << "Second run: Getting LTO data failed";

                // Return value
                return;
        }
        ////////////////////////////////////////////////////////////////////////////////
///////////


        ////////////////////////////////////////////////////////////////////////////////
///////////
        // Third run: process data
        // Generate helper ressources
        qvScratch1 = qlHashes2.size();

        // Fill in Helper register
        qsSettings->beginGroup("LTO01GEN");
        qslOutputData << generatorLib->writeHelperRegister(qvScratch1.toInt(),
qsSettings->value("3", "No entry").toString());
        qsSettings->endGroup();

        // Get DB-Address
        if(!qlHashes2.isEmpty()) {
                for(int a = 0; a < qlHashes2.size(); ++a) {
                        //
                        qmhLtoHash = qlHashes2.at(a);

                        //
                        qvScratch2 = a;

                        // get data
                        QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>::const_iterator i;
                        for(i = qmhLtoHash->constBegin(); i != qmhLtoHash->constEnd();
++i) {

                                // Find DB-Address
                                qsSettings->beginGroup("LTO01");
                                QMultiHash<QString, QHash<QString,
QString>*>::const_iterator x = i.value()->find(qsSettings->value("2", "No
entry").toString());
                                while (x != i.value()->end() && x.key() == qsSettings-
>value("2", "No entry").toString()) {
                                        // Get Key
                                        qsScratch2 = x.key();

                                        // get data
                                        QHash<QString, QString>::const_iterator y =
x.value()->find("Address");
                                        while (y != x.value()->end() && y.key() ==
"Address") {
                                                // Get DB Address
                                                qsAddress = y.value();

                                                // Output data
                                                qvScratch1 = qlResult.size();
```

```
                                        // Insert Data into List
                                        qslOutputData << "CFG_LTO0" <<
qvScratch2.toString() << "\t" << "EQU" << "\t" << "DB " << qsAddress << "\t" << ";
Datablock for: " << qsScratch1 << "\n" << "DB CFG_LTO0" << qvScratch2.toString();

                                        // Move DB Symbol-Name into StringList
                                        qslLtoDbAdresses << "CFG_LTO0" %
qvScratch2.toString();

                                        // increment iterator
                                        ++y;
                                }
                                // increment iterator
                                ++x;
                        }
                        qsSettings->endGroup();

                        // Get number of DB items
                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator d;
                        for(d = i.value()->constBegin(); d != i.value()-
>constEnd(); ++d) {
                                // Get Key
                                qsScratch2 = d.key();

                                // Set Variables
                                qvScratch1 = 0;

                                // get data
                                QHash<QString, QString>::const_iterator e =
d.value()->find("DBIndex");
                                while (e != d.value()->end() && e.key() ==
"DBIndex") {
                                        if(e.value().toInt() > iDbIndex) {

                                                iDbIndex = e.value().toInt();
                                                qvScratch1 = iDbIndex + 1;

                                                // Output data
                                                qsDbItems1 = " [" %
qvScratch1.toString() % "] ";
                                        }
                                        // increment iterator
                                        ++e;
                                }
                        }


                        // Get DB configuration data
                        qsSettings->beginGroup("LTO01GEN");
                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator f = i.value()->find(qsSettings->value("1", "No
entry").toString());
                        while (f != i.value()->end() && f.key() == qsSettings-
>value("1", "No entry").toString()) {
                                // Get Key
                                qsScratch2 = f.key();

                                // get data
                                QHash<QString, QString>::const_iterator g =
f.value()->find("DatapointValue");
```

```
                                while (g != f.value()->end() && g.key() ==
"DatapointValue") {
                                        // Get DB Data
                                        qsDbItems2 = g.value();

                                        // increment iterator
                                        ++g;
                                }

                                // increment iterator
                                ++f;
                        }
                        qsSettings->endGroup();


                        // Get DB configuration data
                        qsSettings->beginGroup("LTO01GEN");
                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator ff = i.value()->find(qsSettings->value("2", "No
entry").toString());
                        while (ff != i.value()->end() && ff.key() == qsSettings-
>value("2", "No entry").toString()) {
                                // Get Key
                                qsScratch2 = ff.key();

                                // get data
                                QHash<QString, QString>::const_iterator gg =
ff.value()->find("DatapointValue");
                                while (gg != ff.value()->end() && gg.key() ==
"DatapointValue") {
                                        // Get DB Data
                                        qsDbItems3 = gg.value();
                                        qsDbItems3 = ", " % qsDbItems3;

                                        // increment iterator
                                        ++gg;
                                }

                                // increment iterator
                                ++ff;
                        }
                        qsSettings->endGroup();


                        // Insert Pointers to Input addresses
                        if(qsScratch2.contains("IN_Input")) {
                                qsScratch = qsScratch2;
                        }

                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator ffff = i.value()->find(qsScratch);
                        while (ffff != i.value()->end() && ffff.key() == qsScratch)
{
                                // Get Key
                                qsScratch3 = ffff.key();

                                // get data
                                QHash<QString, QString>::const_iterator gggg =
ffff.value()->find("PAR_IN");
                                while (gggg != ffff.value()->end() && gggg.key() ==
"PAR_IN") {
                                        // Get DB Data
                                        qsScratch4 = gggg.value();
```

```
                                        // Check string
                                        if(qsScratch4.contains("Logical Input")) {
                                               // insert 0
                                               qsDbItems4.prepend(", 0");
                                        }
                                        else {
                                               // Check string
                                               if(qsScratch4.section('.', 0, 0) ==
"F") {
                                                      // Insert flag-address
                                                      qsDbItems4.prepend(", " %
qsScratch4.section('.', 1, 1));
                                               }

                                               else {
                                                      // Get last bit of the DMS-Name
                                                      qsDmsNameLastPart =
qsScratch4.section(':', -1, -1);

                                                      // Search for address
                                                      qlResultAddress =
qhTrees.value("Data-Tree")->findItems(qsDmsNameLastPart, Qt::MatchRecursive |
Qt::MatchContains);

                                                      if(!qlResultAddress.isEmpty()) {
                                                             for(int i = 0; i <
qlResultAddress.size(); ++i) {
                                                                    // Assemble DMS-
Name
                                                                    // At first get
Parent of the item at i
                                                                    qsiResultAddress =
qlResultAddress.at(i);

                                                                    // Check there is a
parent
                                                                    qsiResult =
qsiResultAddress;
                                                                    if(qsiResult != 0)
{
       while(qsiResult->parent()) {

       qsDmsName.prepend(qsiResult->text() % ":");

       qsiResult = qsiResult->parent();
                                                                           }
                                                                    }
                                                                    // Move DMS-Name
into string list
       qsDmsName.remove(qsDmsName.lastIndexOf(":"), 1);

                                                                    // Check DMS-Name
                                                                    if(qsDmsName ==
qsScratch4) {
       if(qsiResultAddress->hasChildren()) {
                                                                                  //
       QString qsTest2 = qsiResultAddress->text();
```

```
        iRowCount = qsiResultAddress->rowCount();

                                                                                //
Get Child

        qsiChild1 = qsiResultAddress->child(0, 0);

        QString qsTest3 = qsiChild1->text();

        if(qsiChild1->text() == "Address") {

        // Check if child has children itself

        if(qsiChild1->hasChildren()) {

                // Get Address, which is the child's text

                qsiChildAddress = qsiChild1->child(0, 0);

                qsAddress = qsiChildAddress->text();

                qslLtoInputAddresses << qsAddress;


                // break loops

                i = qlResultAddress.size();

        }
                                                                                        }
                                                                                }
                                                                                else {
                                                                                        //
insert 0

        qsDbItems4.prepend(", 0");
                                                                                        }
                                                                                }

                                                                                // Clear string
                                                                                qsDmsName.clear();
                                                                        }
                                                                }

                                                                                // insert Address
                                                                                qsDbItems4.prepend(", " %
qsAddress);
                                                                        }
                                                                }

                                                                        // increment iterator
                                                                        ++gggg;

                                                        }

                                                        // increment iterator
                                                        ++ffff;
                                                }

                                                // Insert Polarities of the inputs
                                                if(qsScratch2.contains("CFG_LogicInput")) {
                                                        qsScratch = qsScratch2;
```

```
                        }

                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator fffff = i.value()->find(qsScratch);
                        while (fffff != i.value()->end() && fffff.key() ==
qsScratch) {
                                // Get Key
                                qsScratch3 = fffff.key();

                                // get data
                                QHash<QString, QString>::const_iterator ggggg =
fffff.value()->find("DatapointValue");
                                while (ggggg != fffff.value()->end() && ggggg.key()
== "DatapointValue") {
                                        // Get DB Data
                                        qsScratch4 = ggggg.value();
                                        qsDbItems5.prepend(", " % qsScratch4);

                                        // increment iterator
                                        ++ggggg;

                                }

                                // increment iterator
                                ++fffff;
                        }
                }

                // Insert Reserve Configuration Datapoints
                for(int i = 0; i < 8; ++i) {
                        qsDbItems3.append(", 0");
                }

                // Insert Output pointers
                for(int i = 0; i < 2; ++i) {
                        qsDbItems6.append(", 0");
                }

                // Insert NewLine
                qsDbItems6.append("\n\n\n");

                // Append DB-Data to the StringList
                qslOutputData << qsDbItems1;
                qslOutputData << qsDbItems2;
                qslOutputData << qsDbItems3;
                qslOutputData << qsDbItems4;
                qslOutputData << qsDbItems5;
                qslOutputData << qsDbItems6;

                // Clear Strings
                qsDbItems1.clear();
                qsDbItems2.clear();
                qsDbItems3.clear();
                qsDbItems4.clear();
                qsDbItems5.clear();
                qsDbItems6.clear();

                // Set Variables
                iDbIndex = 0;

                // Insert LTO FB-Call
                // Get all keys
                qsSettings->beginGroup("LTO01");
```

```
                        qslKeys = qsSettings->allKeys();
                        qsSettings->endGroup();

                        // Get number of keys
                        iKeyCount = qslKeys.size();

                        // Insert Data
                        for(int i = 1; i <= iKeyCount; ++i) {
                                //
                                qsSettings->beginGroup("LTO01");
                                qvScratch1 = i;
                                qsScratch1 = qsSettings->value(qvScratch1.toString(), "No
entry").toString();
                                qsSettings->endGroup();

                                // Assemble the data
                                switch(qvScratch1.toInt()) {
                                        // Header
                                        case 1:
                                                qsGeneratorTemplate.append(qsScratch1 % "\n");
                                        break;

                                        // DB Address
                                        case 2:
                                        // DB Address (for Pointer)
                                        case 3:
                                                qsScratch1 = qsScratch1 %"\t; [" %
qvScratch1.toString() % "]" % qsScratch1  % "\n";
                                                qsGeneratorTemplate.append("DB " %
qslLtoDbAdresses.at(a) % "; [" % qvScratch1.toString() % "]" % "DB Address"  % "\n");
                                        break;

                                        // Configuration Pointer
                                        case 4:
                                        // Output Address
                                        case 5:
                                        // OutputInvers Address
                                        case 6:
                                        // Output Address (for Pointer)
                                        case 7:
                                        // OutputInvers Address (for Pointer)
                                        case 8:
                                                qsScratch1 = qsScratch1 % "\t; [" %
qvScratch1.toString() % "]" % qsScratch1  % "\n";
                                                qsScratch3 = generatorLib-
>makeSymbol(qslDmsNames.at(a), qsScratch1);
                                                qsGeneratorTemplate.append(qsScratch3);
                                        break;

                                        // Helper Register
                                        case 9:
                                                qsSettings->beginGroup("LTO01GEN");
                                                qsGeneratorTemplate.append(qsSettings-
>value("3", "No entry").toString() % "+" % qvScratch2.toString() % "\t; [" %
qvScratch1.toString() % "]" % qsScratch1  % "\n\n\n");
                                                qsSettings->endGroup();
                                        break;

                                        // Error
                                        default:
                                                qsGeneratorTemplate.append("Something is
wrong\n");
                                }
```

```
                }
            }
            // Footer
            qsSettings->beginGroup("LTO01FOOTER");
            qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString()
% "\n\n\n");
            qsSettings->endGroup();
        }
        else {
            // Debug output
            qDebug() << "Third run: processing LTO data failed";

            // Return value
            return;
        }
        ////////////////////////////////////////////////////////////////////////////
////////////


        ////////////////////////////////////////////////////////////////////////////
////////////
        // Find VM's
        qlResult = qhTrees.value("Data-Tree")->findItems("VM001", Qt::MatchRecursive |
Qt::MatchContains);
        ////////////////////////////////////////////////////////////////////////////
////////////


        ////////////////////////////////////////////////////////////////////////////
////////////
        // First run: process vm data
        if(!qlResult.isEmpty()) {
            // Get DMS-Names
            qslDmsNames.clear();
            qslDmsNames = generatorLib->getDmsNames(qlResult);
        }
        else {
            // Debug output
            qDebug() << "Fourth run: processing VM data failed";

            // Return value
            return;
        }
        ////////////////////////////////////////////////////////////////////////////
////////////


        ////////////////////////////////////////////////////////////////////////////
////////////
        // Second run: Get DB data and elements of the VM's
        // Header
        qsSettings->beginGroup("VM001HEADER");
        qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString() %
"\n\n");
        qsSettings->endGroup();


        if(!qlResult.isEmpty()) {
            // Clear List
            qlHashes2.clear();

            // Get data
            for(int i = 0; i < qlResult.size(); ++i) {
```

```
                        // Create LTO-Hash
                        qlHashes2 << generatorLib->createVmHash(qlResult.at(i)->parent()-
>parent(), qslDmsNames.at(i));
                }
        }
        else {
                // Debug output
                qDebug() << "Second run: Getting LTO data failed";

                // Return value
                return;
        }
        ////////////////////////////////////////////////////////////////////////////////
///////////


        ////////////////////////////////////////////////////////////////////////////////
///////////
        // Third run: process data
        // Generate helper ressources
        qsSettings->beginGroup("VM001");
        // Get keys
        qslKeys = qsSettings->allKeys();
        // Get number of keys
        iKeyCount = qslKeys.size();
        //
        for(int i = 1; i <= iKeyCount; ++i) {
                qvScratch1 = i;

                // Count number of used helper registers
                if(qsSettings->value(qvScratch1.toString(), "No entry").toString() ==
"Register") {
                        qvScratch3 = qvScratch3.toInt() + 1;
                }
        }

        qsSettings->endGroup();

        qvScratch1 = qlHashes2.size();

        // Fill in Helper register
        qsSettings->beginGroup("VM001GEN");
        qsGeneratorTemplate.append(generatorLib->writeHelperRegister(qvScratch3.toInt(),
qsSettings->value("4", "No entry").toString()));
        qsSettings->endGroup();

        // Get DB-Address
        if(!qlHashes2.isEmpty()) {
                for(int a = 0; a < qlHashes2.size(); ++a) {
                        //
                        qmhLtoHash = qlHashes2.at(a);
                        qvScratch2 = qlHashes2.size();

                        // get data
                        QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>::const_iterator i;
                        for(i = qmhLtoHash->constBegin(); i != qmhLtoHash->constEnd();
++i) {

                                // Find DB-Address
                                qsSettings->beginGroup("VM001");
```

```
                        QString qsTest = qsSettings->value("2", "No
entry").toString();

                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator x = i.value()->find(qsSettings->value("2", "No
entry").toString());
                        while (x != i.value()->end() && x.key() == qsSettings-
>value("2", "No entry").toString()) {
                                // Get Key
                                qsScratch2 = x.key();

                                // get data
                                QHash<QString, QString>::const_iterator y =
x.value()->find("Address");
                                while (y != x.value()->end() && y.key() ==
"Address") {
                                        // Get DB Address
                                        qsAddress = y.value();

                                        // Output data
                                        qvScratch1 = qlResult.size();

                                        // Insert Data into List
                                        qsGeneratorTemplate.append("CFG_VM0" %
qvScratch2.toString() % "\t" % "EQU" % "\t" % "DB " % qsAddress % "\t" % "; Datablock
for: " % qsScratch1 % "\n" % "DB CFG_VM0" % qvScratch2.toString());

                                        // Move DB Symbol-Name into StringList
                                        qslVmDbAdresses << "CFG_VM0" %
qvScratch2.toString();


                                        // increment iterator
                                        ++y;
                                }
                                // increment iterator
                                ++x;
                        }
                        qsSettings->endGroup();

                        // Get number of DB items
                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator d;
                        for(d = i.value()->constBegin(); d != i.value()-
>constEnd(); ++d) {
                                // Get Key
                                qsScratch2 = d.key();

                                // Set Variables
                                qvScratch1 = 0;

                                // get data
                                QHash<QString, QString>::const_iterator e =
d.value()->find("DBIndex");
                                while (e != d.value()->end() && e.key() ==
"DBIndex") {
                                        if(e.value().toInt() > iDbIndex) {

                                                iDbIndex = e.value().toInt();
                                                qvScratch1 = iDbIndex + 1;

                                                // Output data
                                                qsDbItems1 = " [" %
qvScratch1.toString() % "] ";
```

```
                                        }
                                        // increment iterator
                                        ++e;
                                }
                        }

                        // Insert Pointers to LTO addresses
                        qsSettings->beginGroup("VM001GEN");

                        qsScratch = qsSettings->value("3", "No entry").toString();
                        if(qsScratch2.contains(qsScratch)) {
                                qsScratch = qsScratch2;
                        }

                        QMultiHash<QString, QHash<QString,
QString>*>::const_iterator ffff = i.value()->find(qsScratch);
                        while (ffff != i.value()->end() && ffff.key() == qsScratch)
{
                                // Get Key
                                qsScratch3 = ffff.key();

                                // get data
                                QHash<QString, QString>::const_iterator gggg =
ffff.value()->find("PAR_IN");
                                while (gggg != ffff.value()->end() && gggg.key() ==
"PAR_IN") {
                                        // Get DB Data
                                        qsScratch4 = gggg.value();

                                        // Check string
                                        if(qsScratch4.contains("Pointer to the LTO"))
{
                                                // insert 0
                                                qsDbItems3.prepend(", 0");
                                        }
                                        else {
                                                // Check string
                                                if(qsScratch4.section('.', 0, 0) ==
"F") {
                                                        // Insert flag-address
                                                        qsDbItems3.prepend(", " %
qsScratch4.section('.', 1, 1));
                                                }

                                                else {
                                                        // Get last bit of the DMS-Name
                                                        qsDmsNameLastPart =
qsScratch4.section(':', -1, -1);

                                                        // Search for address
                                                        qlResultAddress =
qhTrees.value("Data-Tree")->findItems(qsDmsNameLastPart, Qt::MatchRecursive |
Qt::MatchContains);

                                                        if(!qlResultAddress.isEmpty()) {
                                                                for(int i = 0; i <
qlResultAddress.size(); ++i) {
                                                                        // Assemble DMS-
Name
                                                                        // At first get
Parent of the item at i
                                                                        qsiResultAddress =
qlResultAddress.at(i);
```

```
parent
qsiResultAddress;
{
        while(qsiResult->parent()) {
        qsDmsName.prepend(qsiResult->text() % ":");
        qsiResult = qsiResult->parent();

into string list
        qsDmsName.remove(qsDmsName.lastIndexOf(":"), 1);

qsScratch4) {
        if(qsiResultAddress->hasChildren()) {
        QString qsTest2 = qsiResultAddress->text();
        iRowCount = qsiResultAddress->rowCount();
Get Child
        qsiChild1 = qsiResultAddress->child(0, 0);
        QString qsTest3 = qsiChild1->text();
        if(qsiChild1->text() == "Address") {
        // Check if child has children itself
        if(qsiChild1->hasChildren()) {
                // Get Address, which is the child's text
                qsiChildAddress = qsiChild1->child(0, 0);
                qsAddress = qsiChildAddress->text();
                qslVmInputAddresses << qsAddress;

                // break loops
                i = qlResultAddress.size();
        }

insert 0
                // Check there is a
                qsiResult =
                if(qsiResult != 0)

                        }
                }
                // Move DMS-Name

                // Check DMS-Name
                if(qsDmsName ==

                                //

                                //

                        }
                }
                else {
                        //
```

```
		qsDbItems3.prepend(", 0");
                                                                                }
                                                                        }

                                                                        // Clear string
                                                                        qsDmsName.clear();
                                                                }
                                                        }

                                                        // insert Address
                                                        qsDbItems3.prepend(", " %
qsAddress);
                                                }
                                        }

                                        // increment iterator
                                        ++gggg;

                                }

                                // increment iterator
                                ++ffff;
                        }
                        qsSettings->endGroup();
                }

                // Add Number of used LTO's
                qvScratch1 = qslVmInputAddresses.size();
                qsDbItems2.prepend(qvScratch1.toString());

                // Prepend Cycletime
                qsDbItems2.prepend("0, ");

                // Insert Reserve Configuration Datapoints
                for(int c = 0; c < 8; ++c) {
                        qsDbItems2.append(", 0");
                }

                // Insert NewLine
                qsDbItems3.append("\n\n\n");

                // Append DB-Data to the StringList
                qsGeneratorTemplate.append(qsDbItems1);
                qsGeneratorTemplate.append(qsDbItems2);
                qsGeneratorTemplate.append(qsDbItems3);

                // Clear Strings
                qsDbItems1.clear();
                qsDbItems2.clear();
                qsDbItems3.clear();

                // Set Variables
                iDbIndex = 0;

                // Insert VM FB-Call
                // Get all keys
                qsSettings->beginGroup("VM001");
                qslKeys = qsSettings->allKeys();
                qsSettings->endGroup();

                // Get number of keys
                iKeyCount = qslKeys.size();
```

```
                    qvScratch2 = qvScratch3;

                    // Insert Data
                    for(int i = 1; i <= iKeyCount; ++i) {
                           //
                           qsSettings->beginGroup("VM001");
                           qvScratch1 = i;
                           qsScratch1 = qsSettings->value(qvScratch1.toString(), "No
entry").toString();
                           qsSettings->endGroup();

                           // Assemble the data
                           switch(qvScratch1.toInt()) {
                                  // Header
                                  case 1:
                                         qsGeneratorTemplate.append(qsScratch1 % "\n");
                                  break;

                                  // DB Address
                                  case 2:
                                  // DB Address (for Pointer)
                                  case 3:
                                         qsScratch1 = qsScratch1 %"\t; [" %
qvScratch1.toString() % "]" % qsScratch1  % "\n";
                                         qsGeneratorTemplate.append("DB " %
qslVmDbAdresses.at(a) % "; [" % qvScratch1.toString() % "]" % "DB Address"  % "\n");
                                  break;

                                  // Configuration Pointer
                                  case 4:
                                         qsScratch1 = qsScratch1 % "\t; [" %
qvScratch1.toString() % "]" % qsScratch1  % "\n";
                                         qsScratch3 = generatorLib-
>makeSymbol(qslDmsNames.at(a), qsScratch1);
                                         qsGeneratorTemplate.append(qsScratch3);
                                  break;

                                  // Helper Register
                                  case 5:
                                  // Helper Register
                                  case 6:
                                  // Helper Register
                                  case 7:
                                  // Helper Register
                                  case 8:
                                  // Helper Register
                                  case 9:
                                         qsSettings->beginGroup("VM001GEN");
                                         qvScratch3 = qvScratch1.toInt() -
qvScratch2.toInt();
                                         qsGeneratorTemplate.append(qsSettings-
>value("4", "No entry").toString() % "+" % qvScratch3.toString() % "\t; [" %
qvScratch1.toString() % "]" % qsScratch1  % "\n");
                                         qsSettings->endGroup();
                                  break;

                                  // Error
                                  default:
                                         qsGeneratorTemplate.append("Something is
wrong\n");
                           }
                    }
```

```
                }
                // Add New-lines
                qsGeneratorTemplate.append("\n\n");

                // VM Footer
                qsSettings->beginGroup("VM001FOOTER");
                qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString()
% "\n\n\n");
                qsSettings->endGroup();

                // COB Footer
                qsSettings->beginGroup("COBFOOTER");
                qsGeneratorTemplate.append(qsSettings->value("1", "No entry").toString()
% "\n\n\n");
                qsSettings->endGroup();
        }
        else {
                // Debug output
                qDebug() << "Third run: processing VM data failed";

                // Return value
                return;
        }

        ////////////////////////////////////////////////////////////////////////////
///////////


        ////////////////////////////////////////////////////////////////////////////
///////////
        // Output Data to the file
        // Set stream to the end of the file
        qtsOutput.seek(qfOutput.size());

        // Append data to the StringList
        qslOutputData << qsGeneratorTemplate;

        // Output Data
        for(int i = 0; i < qslOutputData.size(); ++i) {
                qtsOutput << qslOutputData.at(i);
        }
        ////////////////////////////////////////////////////////////////////////////
///////////

        // emit signal
        emit vmSymbolsWritten();

        // Return value
        return;
}

//! This is the state-entered-Function for the close-file-state.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmsCloseVmFile_entered() {
        // Debug message
        qDebug() << "Entering step: if_qsmsCloseVmFile_entered";

        // Debug output
        qDebug() << "closing vm-file...";
```

```
        // Close file
        qfOutput.close();

        // emit signal
        emit vmFileClosed();

        // Return value
        return;
}

//! This is the state-entered-Function for the final-state of the vm generator
machine.
    /*!
      \param - None.
        \return void
    */
void sGen::if_qsmfsEndVmGeneratorMachine_entered() {
        // Debug message
        qDebug() << "Entering step: if_qsmfsEndVmGeneratorMachine_entered";

        // emit signal
        emit vmGenerated();

        // Return value
        return;
}
// *************** End of Step Functions *****************
```

Appendix G Sources sGenLib

Appendix G I Source code sGenLib.h

```
#ifndef SGENLIB_H
#define SGENLIB_H

#include <QtGui>                          // QT-Headers
#include "D:\Thoemus_Stuff\Projekte\C++\pTool\global\globals.h"

/****************** Defines ************************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ************************


//!  This is the sGenLib Class, it is used to generate IL-Code for Saia-Burgess PLC-
Systems. It works in close connection with the ProMoS NT SCADA-System developed by MST
Systemtechnik AG.
/*!
  This Library is designed to generate IL-Code for Saia-Burgess PLC-Systems. As input
for the generators the promos.dms and the bmo.dms Files are used.
  The above mentioned Files will be accessed offline, this means that ProMoS NT needs
not to run. The Library will be used to enhance the existing ProMoS NT Code-Generator.
*/
class sGenLib
{
public:
	sGenLib();
	~sGenLib();

	// Settings
	QSettings *qsSettings;
	QSettings *qsLibSettings;

	// Symbol specific Functions
	QString makeSymbol(QString qsDmsName, QString qsAttribute);
	QString makeSymbol(QString qsDmsName);
	QStringList makeSymbol(QStringList qslDmsNames);

	// Generator Functions
	int writeHeader(const struct struHeaderData *struHeader);
	QString writeHelperRegister(int iNbrOfElements, QString qsName);

	QStringList getDmsNames(QList<QStandardItem*> qlInData);
	QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
createLtoHash(QStandardItem *qsiResult, QString qsnDmsName);
	QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
createVmHash(QStandardItem *qsiResult, QString qsInDmsName);

	/*
	int createNewFile(QString qsFileName);
	int createNewFile(const struct struGeneratorData *struGenData);
	int generateNichtAutoCode(const struct struGeneratorData *struGenData, const
struct struGenNichtAuto *struNichtAuto);
	int writeFooter(const struct struFooterData *struFooter);
	int writeNichtAuto(const struct struGenNichtAuto *struNichtAuto);
	int writeBetrieb(const struct struBetriebData *struBetrieb);
	int writeStoerungen(const struct struStoerungenData *struStoerungen);
	int insertInMain(const struct struInsertInMainData *struInsertInMain);
	*/
```

```
        // Functions to get informations out of the Saia-Files
        QString pcdTyp(QString qsInPath, QString qsInCpuName);
        QString stationNumber(QString qsInPath, QString qsInCpuName);
        QString pgVersion(QString qsInPath, QString qsInCpuName);

private:

};

#endif // SGENLIB_H
```

Appendix G II Source code sGenLib.cpp

```
/*****************************************************************************
 Implementation of the sGenLib DLL
 ---------------------------------

 Date         : 13.12.2010
 File       : sgenlib.cpp
 Author             : Thomas Gasser
 ©          : 2010 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's Saia-Burgess Code Generator Library.
/*!
  This Library is designed to use with the SCADA-System ProMoS NT from MST
Systemtechnik AG.
  It provides functions to generate IL-Code from the configuration Files of ProMoS NT.
The
  Library is designed to work offline.
*/

/****************** Includes ************************/
#include "stdafx.h"                     // Precompiled headers
#include "sgenlib.h"              // Project header
/****************** Includes ************************/

/******************* Defines ************************/
#define QT_USE_FAST_CONCATENATION
/******************* Defines ************************/


// Constructor / Destructor
//! This is the Constructor of the sGenLib Library.
    /*!
      At the moment the Constructor of the sGenLib Library is not used at all.
    */
sGenLib::sGenLib() {
       // Create library settings object
       qsLibSettings = new QSettings("../cfg/sTool.ini", QSettings::IniFormat, 0);

}

//! This is the Destructor of the sGenLib Library.
    /*!
      At the moment the Destructor of the sGenLib Library is not used at all.
    */
sGenLib::~sGenLib() {
       // Delete Settings
       delete qsLibSettings;

}

// Symbol specific functions
// p-Functions

//! This Function creates a Saia compatible Symbol out of the two given Parameters (eq
H02:MT:500, Istwert => H02.MT_500.Istwert).
    /*!
```

```
      \param qsDmsName A QString which holds the DMS-Name to be changed into a Symbol.
        \param qsAttribute A QString which holds the Attribute to be added at the end
of the Symbolnames
      \return A QString which holds the Symbol.
        \sa makeSymbol(QString qsDmsName), makeSymbol(QStringList qslDmsNames).
        */
QString sGenLib::makeSymbol(QString qsDmsName, QString qsAttribute) {
       // Check Attribute
       if(qsAttribute == "No entry") {
              QMessageBox::warning(0, "sGenLib", "Falsches Attribute");
              return "";
       }
       else {
              // Take DMS-Name and shape it to the needs of promos
              qsDmsName = qsDmsName.section(":", 1, qsDmsName.count(":"));
              qsDmsName = qsDmsName.replace(qsDmsName.lastIndexOf(":") , 1, "_");
              qsDmsName = qsDmsName.replace(":", ".");
              qsAttribute = qsAttribute.prepend(".");
              qsDmsName = qsDmsName.append(qsAttribute);
       }

       // Return value
       return qsDmsName;
}


//! This Function creates a Saia compatible Symbol out of the given Parameter (eq
H02:MT:500, Istwert => H02.MT_500.Istwert).
    /*!
      \param qsDmsName A QString which holds the DMS-Name to be changed into a Symbol.
      \return A QString which holds the Symbol.
        \sa makeSymbol(QString qsDmsName, QString qsAttribute), makeSymbol(QStringList
qslDmsNames).
        */
QString sGenLib::makeSymbol(QString qsDmsName) {
       // Locals
       QString qsRetString, qsScratch;

       // Take DMS-Name and shape it to the needs of promos
              qsScratch = qsDmsName.section(":", 0, (qsDmsName.count(":") -1));
              qsScratch = qsScratch.replace(qsScratch.lastIndexOf(":", -1), 1, "_");
              qsScratch = qsScratch.append("." + qsDmsName.section(":",
qsDmsName.count(":")));
              qsRetString = qsScratch.replace(":", ".");

       // Return value
       return qsRetString;
}


//! This Function creates a Saia compatible Symbol out of the given Parameter (eq
H02:MT:500, Istwert => H02.MT_500.Istwert).
    /*!
      \param qslDmsNames A QStringList which holds the DMS-Names to be changed into a
Symbol.
      \return A QStringList which holds the Symbol generated.
        \sa makeSymbol(QString qsDmsName, QString qsAttribute), makeSymbol(QString
qsDmsName).
        */
QStringList sGenLib::makeSymbol(QStringList qslDmsNames) {
       // Locals
       QStringList qslRetList;
       QString qsDmsName, qsScratch;
```

```
	// Take DMS-Name and shape it to the needs of promos qsDmsName.lastIndexOf(":")
	for(int i = 0; i < qslDmsNames.size(); ++i) {
		qsDmsName = qslDmsNames.at(i);
		qsScratch = qsDmsName.section(":", 0, (qsDmsName.count(":") -1));
		qsScratch = qsScratch.replace(qsScratch.lastIndexOf(":", -1), 1, "_");
		qsScratch = qsScratch.append("." + qsDmsName.section(":",
qsDmsName.count(":")));
		qsDmsName = qsScratch.replace(":", ".");
		qslRetList << qsDmsName;
	}

	// Return value
	return qslRetList;
}


// Writer-Functions
//! This Function creates a new File.
    /*!
      \param *struGenData A structure which holds the data needed by the Generator.
        \return A QStringList which holds the Symbol generated.
        \sa createNewFile(QString qsFileName).
        */
/*int sGenLib::createNewFile(const struct struGeneratorData *struGenData) {
	// Locals
	QFile qfOutput, qfInput;
	QTextStream qtsInput, qtsOutput;
	QString qsline, qsScratch, qsHeader;
	QStringList qslBlockList;
	QDateTime dtDateTime;
	int iRetValue;
	bool ok;

	// Set Filename
	qfOutput.setFileName(struGenData->qsOutputFile);

	// Check file to exist and rename it if it exist
	if(qfOutput.exists()) {
		// set filename to *.bak
		qsScratch = struGenData->qsOutputFile;
		qsScratch.replace(".src", ".bak");
		qfOutput.setFileName(qsScratch);

		// remove existing *.bak file
		if(qfOutput.exists()) {
			qfOutput.remove();
		}

		// set filename back to original
		qfOutput.setFileName(struGenData->qsOutputFile);

		// copy original file to *.bak file
		qfOutput.copy(qsScratch);
	}

	// Create new file
	qtsOutput.setDevice(&qfOutput);
	if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
		QMessageBox::warning(0, "sGenLib", "Die Datei konnte nicht erstellt
werden");
		iRetValue = -1;
	}
```

```
	// Close file
	qfOutput.close();

	// Return value
	return iRetValue;
}*/

//! This Function creates a new File.
    /*!
      \param qsFileName A QString which holds the Path to the place where the new File
should be generatet to.
        \return An Integer  0 := New File createt,
                                                -1 := New File could not be created
        \sa createNewFile(const struct struGeneratorData *struGenData).
        */
/*int sGenLib::createNewFile(QString qsFileName) {
	// Locals
	QFile qfOutput;
	QTextStream qtsOutput;
	QString qsScratch;
	QDateTime dtDateTime;
	int iRetValue;

	// Set Filename
	qfOutput.setFileName(qsFileName);

	// Check file to exist and rename it if it exist
	if(qfOutput.exists()) {
		// set filename to *.bak
		qsScratch = qsFileName;
		qsScratch.replace(".src", ".bak");
		qfOutput.setFileName(qsScratch);

		// remove existing *.bak file
		if(qfOutput.exists()) {
			qfOutput.remove();
		}

		// set filename back to original
		qfOutput.setFileName(qsFileName);

		// copy original file to *.bak file
		qfOutput.copy(qsScratch);
	}

	// Create new file
	qtsOutput.setDevice(&qfOutput);
	if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
		QMessageBox::warning(0, "sGenLib", "Die Datei konnte nicht erstellt
werden");
		iRetValue = -1;
	}
	else {
		iRetValue = 0;
	}

	// Close file
	qfOutput.close();

	// Return value
	return iRetValue;
}*/
```

```
//! This Function generates the "Nicht Auto" File.
    /*!
      \param *struGenData A structure which holds the data needed by the Generator.
        \param *struNichtAuto A structure which holds the data needed by the "Nicht
Auto" Generator.
        \return An Integer 0 := "Nicht Auto" File createt,
                                                -1 := File not found
        \sa createNewFile(const struct struGeneratorData *struGenData).
        */
/*int sGenLib::generateNichtAutoCode(const struct struGeneratorData *struGenData,
const struct struGenNichtAuto *struNichtAuto) {
       // Locals
       QFile qfOutput;
       QTextStream qtsOutput;
       QString qsline, qsScratch, qsFileName;
       QStringList qslScratch;
       int iRetValue;

       // Create settings object
       QSettings *qsSettings = new QSettings("../cfg/GenNichtAuto.ini",
QSettings::IniFormat, 0);

       // Open output file
       qfOutput.setFileName(struGenData->qsOutputFile);
       qtsOutput.setDevice(&qfOutput);
       if (!qfOutput.open(QIODevice::ReadWrite | QIODevice::Text)) {
              QMessageBox::warning(0, "sGenLib", "Kein File im Ordner gefunden");
              iRetValue = -1;
       }
       else {
              // Set file pointer to the end of the file
              qtsOutput.seek(qfOutput.size());

              // Sort List
              qslScratch = struNichtAuto->qslDmsNames;
              qslScratch.sort();

              // Write Block Header
              // Get values from ini-File
              qsScratch = qsSettings->value("PcdBlocks/ttBlockHeader", "No
entry").toString();
              qtsOutput << qsScratch << "\n\n";

              // Write Block
              qsFileName = struGenData->qsOutputFile.section('/', -1);
              qsScratch = qsSettings->value("PcdBlocks/ttBlockStart", "No
entry").toString();
              qtsOutput << "\t" << qsScratch << " PB_" + qsFileName.section('.', 0, 0)
<<"\n\n";

              // Write Conjunction Header
              qsScratch = qsSettings->value("Templates/ttHeader", "No
entry").toString();
              qtsOutput << qsScratch << "\n";

              qsScratch = qsSettings->value("Templates/ttHeaderLabel", "No
entry").toString();
              qtsOutput << qsScratch << "\n\n";

              // Create conjunctions
              for(int i = 0; i < qslScratch.size(); ++i) {
                     if(i < 1) {
```

```
                        qtsOutput << "\t" << "STH" << "\t" << qslScratch.at(i) <<
"\n";
                }
                else {
                        qtsOutput << "\t" << "ORH" << "\t" << qslScratch.at(i) <<
"\n";
                }
        }
        // Create conjunction result
        qtsOutput << "\t" << "OUT" << "\t" << struNichtAuto->qsOutputTo << "\n";

        // Write Conjunction footer
        qsScratch = qsSettings->value("Templates/ttFooterLabel", "No
entry").toString();
        qtsOutput << "\n\n" << qsScratch << "\n";

        qsScratch = qsSettings->value("Templates/ttFooter", "No
entry").toString();
        qtsOutput << qsScratch << "\n\n";

        // Write Block footer
        qsScratch = qsSettings->value("PcdBlocks/ttEndLabel", "No
entry").toString();
        qtsOutput << qsScratch << "\n";

        // Write endblock
        qsScratch = qsSettings->value("PcdBlocks/ttBlockEnd", "No
entry").toString();
        qtsOutput << "\t" << qsScratch << "\n\n";

        qsScratch = qsSettings->value("PcdBlocks/ttBlockFooter", "No
entry").toString();
        qtsOutput << qsScratch << "\n";

        // Write EndGroup
        qtsOutput << "$ENDGROUP" << "\n";

        // Set return value
        iRetValue = 0;
    }

    // Close file
    qfOutput.close();

    // Return value
    return iRetValue;
}*/

//! This Function generates the File Header.
  /*!
    \param *struHeader A structure which holds the data needed to generate the File
Header.
      \return An Integer 0 := File Header written correctly,
                                        -1 := Output File not found,
                                        -2 := No Template File found
      */
int sGenLib::writeHeader(const struct struHeaderData *struHeader) {
    // Locals
    QFile qfInput, qfOutput;
    QTextStream qtsInput, qtsOutput;
    QString qsHeader, qsScratch;
    QDateTime dtDateTime;
    int iRetValue;
```

```
	// Open template file
	qfInput.setFileName(struHeader->qsHeaderTemplate);
	qtsInput.setDevice(&qfInput);
	if (!qfInput.open(QIODevice::ReadOnly | QIODevice::Text)) {
		QMessageBox::warning(0, "sGenLib", "Kein Template-File gefunden");
		iRetValue = -2;
	}
	else {
		// Read header template
		qsHeader = qtsInput.readAll();

		// Close template file
		qfInput.close();

		// Insert Date and Time and replace strings

		qsScratch = dtDateTime.currentDateTime().toString("dddd dd. MMMM yyyy
hh:mm:ss");
		qsHeader.replace("<DateTime>", qsScratch);
		qsHeader.replace("<PcdType>", struHeader->qsPcdType);
		qsScratch = struHeader->qsOutputFile.section('/', -1);
		qsHeader.replace("<FileName>", qsScratch);
		qsHeader.replace("<PgVersion>", struHeader->qsPgVersion);
		qsHeader.replace("<Year>",
dtDateTime.currentDateTime().toString("yyyy"));
		qsHeader.replace("<Version>", "1.0.0.0");
		qsHeader.replace("<Changes>", "-");
		qsHeader.replace("<Station>", "$STATION " + struHeader->qsStationNumber);
		qsHeader.replace("<blockName>", qsScratch.section('.', 0, 0));
		qsHeader.replace("<blockTyp>", struHeader->qsBlockTyp);
		qsHeader.replace("<groupName>", qsScratch.section('.', 0, 0));
	}

	// Open output file
	qfOutput.setFileName(struHeader->qsOutputFile);
	qtsOutput.setDevice(&qfOutput);
	// check whether the file is allready opened
	if (!qfOutput.isOpen()) {
		if (!qfOutput.open(QIODevice::WriteOnly | QIODevice::Text)) {
			QMessageBox::warning(0, "sGenLib", "Die Datei konnte nicht
geöffnet werden");
			iRetValue = -1;
		}
		else {
			// Write data to the output file
			qtsOutput << qsHeader << "\n\n\n";

			// set return code to success
			iRetValue = 0;
		}
	}
	else {
		// Write data to the output file
		qtsOutput << qsHeader << "\n\n\n";

		// set return code to success
		iRetValue = 0;
	}

	// Close output file
	// qfOutput.close();
```

```
	// Return value
	return iRetValue;
}


//! This Function assembles the generator-string .
    /*!
      \param iNbrOfElements An integer holding the number of needed helper registers.
        \param qsName A QString holding the PLC-SymbolName
        \return A QString which holds the assembled string.
        \sa writeHelperRegister(int iNbrOfElements).
        */
QString  sGenLib::writeHelperRegister(int iNbrOfElements, QString qsName) {
	// Locals
	QVariant qvScratch;
	QStringList qslScratch;
	QString qsRetValue;

	// Get number of helper-registers
	qvScratch = iNbrOfElements;

	// Assemble String
	qslScratch << qsName << "\t" << "EQU R" << "[" << qvScratch.toString() << "]"<<
"\t; Helper Registers" << "\n\n";
	qsRetValue = qslScratch.join("");

	// Return value
	return qsRetValue;
}


//! This Function assembles the dms names.
    /*!
      \param qlInData A QList<QStandardItem*> holding the data to be processed.
        \return A QStringList which holding the assembled DMS-Names.
        \sa getDmsNames(QList<QStandardItem*> qlInData).
        */
QStringList sGenLib::getDmsNames(QList<QStandardItem*> qlInData) {
	// Locals
	QString qsDmsName;
	QStringList qslDmsNames;
	QStandardItem *qsiResult;

	// Get all LTO's DMS-Names
	if(!qlInData.isEmpty()) {
		for(int i = 0; i < qlInData.size(); ++i) {
			// Assemble DMS-Name
			// At first get Parent of the item at i
			qsiResult = qlInData.at(i)->parent()->parent();

			// Check there is a parent
			if(qsiResult != 0) {
				while(qsiResult->parent()) {
					qsDmsName.prepend(qsiResult->text() % ":");
					qsiResult = qsiResult->parent();
				}
			}
			// Move DMS-Name into string list
			qsDmsName.remove(qsDmsName.lastIndexOf(":"), 1);
			qslDmsNames << qsDmsName;

			// Clear string
			qsDmsName.clear();
```

```
		}

		// Return value
		return qslDmsNames;
	}
	else {
		// MessageBox
		QMessageBox::warning(0, "sGenLib", "Error getting DMS-Names");

		// Clear list
		qslDmsNames.clear();
		qslDmsNames << "ERROR";

		// Return value
		return qslDmsNames;
	}
}


//! This Function creates the LTO Hash.
    /*!
      \param *qsiResult A Pointer to a QStandardItem holding the data to be processed.
        \param qsInDmsName A QString holding the DMS-Name
        \return A QMultiHash which holding the LTO-Hash.
        \sa createLtoHash(QStandardItem *qsiResult, QString qsInDmsName).
        */
QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
sGenLib::createLtoHash(QStandardItem *qsiResult, QString qsInDmsName) {
	// Locals
	QString qsParent, qsScratch1, qsScratch2, qsScratch3;
	QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
qlReturnHash;
	QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*> *qmhLtoHash;
	QMultiHash<QString, QHash<QString, QString>*> *qmhDatapointHash;
	QHash<QString, QString>    *qhDataHash;
	QStandardItem *qsiChild1, *qsiChild2, *qsiChild3, *qsiChildAddress;
	int iRowCount = 0, iRowCount2= 0, iRowCount3= 0;

	// Create a Data-Hash
	qmhLtoHash = new QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>();
	qlReturnHash << qmhLtoHash;

	// At first get Parent of the item at i
	qsParent = qsiResult->text();

	// Check there are children
	if(qsiResult->hasChildren()) {
		// Check how many children the item has
		iRowCount = qsiResult->rowCount();
		// Check every child to be DB, F or R
		for(int x = 0; x < iRowCount; ++x) {
			// Create Multi-Hashe
			qmhDatapointHash = new QMultiHash<QString, QHash<QString,
QString>*>();

			// Get child on row y, column 0 (allways 0 in tree's)
			qsiChild1 = qsiResult->child(x, 0);
			qsScratch1 = qsiChild1->text();
			// Check if child has children itself
			if(qsiChild1->hasChildren()) {
				// Check how many children the item has
				iRowCount2 = qsiChild1->rowCount();
```

```
                        // Get grand-children
                        for(int y = 0; y < iRowCount2; ++y) {
                                // Get child on row z, column 0 (allways 0 in
tree's)
                                qsiChild2 = qsiChild1->child(y, 0);
                                qsScratch2 = qsiChild2->text();

                                // Check if child has children itself
                                if(qsiChild2->hasChildren()) {
                                        // Check how many children the item has
                                        iRowCount3 = qsiChild2->rowCount();
                                        // Get grand-children
                                        for(int z = 0; z < iRowCount3; ++z) {
                                                // Get child on row z, column 0
(allways 0 in tree's)
                                                qsiChild3 = qsiChild2->child(z, 0);
                                                qsScratch3 = qsiChild3->text();

                                                // Create a Data-Hash
                                                qhDataHash = new QHash<QString,
QString>();

                                                // Insert data into hash
                                                qhDataHash->insert(qsScratch2,
qsScratch3);
                                        }
                                }

                                // Insert Data-Hash into Datapoint-Hash
                                qmhDatapointHash->insert(qsScratch1, qhDataHash);
                        }
                }

                // Insert Data-Hash into Datapoint-Hash
                qmhLtoHash->insert(qsInDmsName, qmhDatapointHash);
        }
        // Return value
        return qlReturnHash;
    }
    else {
        // MessageBox
        QMessageBox::warning(0, "sGenLib", "Error creating Hash");

        // Clear Hash
        qlReturnHash.clear();

        // Return value
        return qlReturnHash;
    }
}


//! This Function creates the VM Hash.
    /*!
      \param *qsiResult A Pointer to a QStandardItem holding the data to be processed.
        \param qsInDmsName A QString holding the DMS-Name
        \return A QMultiHash which holding the LTO-Hash.
        \sa createLtoHash(QStandardItem *qsiResult, QString qsInDmsName).
        */
QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
sGenLib::createVmHash(QStandardItem *qsiResult, QString qsInDmsName) {
    // Locals
    QString qsParent, qsScratch1, qsScratch2, qsScratch3;
```

```
	QList<QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*>*>
qlReturnHash;
	QMultiHash<QString, QMultiHash<QString, QHash<QString, QString>*>*> *qmhLtoHash;
	QMultiHash<QString, QHash<QString, QString>*> *qmhDatapointHash;
	QHash<QString, QString>    *qhDataHash;
	QStandardItem *qsiChild1, *qsiChild2, *qsiChild3, *qsiChildAddress;
	int iRowCount = 0, iRowCount2= 0, iRowCount3= 0;

	// Create a Data-Hash
	qmhLtoHash = new QMultiHash<QString, QMultiHash<QString, QHash<QString,
QString>*>*>();
	qlReturnHash << qmhLtoHash;

	// At first get Parent of the item at i
	qsParent = qsiResult->text();

	// Check there are children
	if(qsiResult->hasChildren()) {
		// Check how many children the item has
		iRowCount = qsiResult->rowCount();
		// Check every child to be DB, F or R
		for(int x = 0; x < iRowCount; ++x) {
			// Create Multi-Hashe
			qmhDatapointHash = new QMultiHash<QString, QHash<QString,
QString>*>();

			// Get child on row y, column 0 (allways 0 in tree's)
			qsiChild1 = qsiResult->child(x, 0);
			qsScratch1 = qsiChild1->text();
			// Check if child has children itself
			if(qsiChild1->hasChildren()) {
				// Check how many children the item has
				iRowCount2 = qsiChild1->rowCount();
				// Get grand-children
				for(int y = 0; y < iRowCount2; ++y) {
					// Get child on row z, column 0 (allways 0 in
tree's)
					qsiChild2 = qsiChild1->child(y, 0);
					qsScratch2 = qsiChild2->text();

					// Check if child has children itself
					if(qsiChild2->hasChildren()) {
						// Check how many children the item has
						iRowCount3 = qsiChild2->rowCount();
						// Get grand-children
						for(int z = 0; z < iRowCount3; ++z) {
							// Get child on row z, column 0
(allways 0 in tree's)
							qsiChild3 = qsiChild2->child(z, 0);
							qsScratch3 = qsiChild3->text();

							// Create a Data-Hash
							qhDataHash = new QHash<QString,
QString>();

							// Insert data into hash
							qhDataHash->insert(qsScratch2,
qsScratch3);
						}
					}

					// Insert Data-Hash into Datapoint-Hash
					qmhDatapointHash->insert(qsScratch1, qhDataHash);
```

```
                        }
                    }

                    // Insert Data-Hash into Datapoint-Hash
                    qmhLtoHash->insert(qsInDmsName, qmhDatapointHash);
                }
                // Return value
                return qlReturnHash;
        }
        else {
                // MessageBox
                QMessageBox::warning(0, "sGenLib", "Error creating Hash");

                // Clear Hash
                qlReturnHash.clear();

                // Return value
                return qlReturnHash;
        }
}


//! This Function returns the PCD-Typ of the given cpu.
    /*!
      \param none.
        \return A QString which holds PCD-Typ.
        */
QString sGenLib::pcdTyp(QString qsInPath, QString qsInCpuName) {
        // Locals
        QString qsScratch, qsRetValue;

        // Create settings object
        qsScratch = qsInPath % qsInCpuName % "/" % qsLibSettings->value("FileNames/2",
"No entry").toString();
        qsSettings = new QSettings(qsScratch, QSettings::IniFormat, 0);

        // get data
        qsScratch = qsLibSettings->value("Groups/2", "No entry").toString() % "/" %
qsLibSettings->value("Keys/2", "No entry").toString();
        qsRetValue = qsSettings->value(qsScratch, "No entry").toString();

        // Delete QSettings
        delete qsSettings;

        // Return value
        return qsRetValue;
}

//! This Function returns the cpu's station number.
    /*!
      \param none.
        \return A QString which holds PCD-Typ.
        */
QString sGenLib::stationNumber(QString qsInPath, QString qsInCpuName) {
        // Locals
        QString qsScratch, qsRetValue;

        // Create settings object
        qsScratch = qsInPath % qsInCpuName % "/" % qsLibSettings->value("FileNames/3",
"No entry").toString();
        qsScratch = qsScratch.replace("CpuName", qsInCpuName);
        qsSettings = new QSettings(qsScratch, QSettings::IniFormat, 0);
```

```
	// get data
	qsScratch = qsLibSettings->value("Groups/3", "No entry").toString() % "/" %
qsLibSettings->value("Keys/3", "No entry").toString();
	qsRetValue = qsSettings->value(qsScratch, "No entry").toString();

	// Delete QSettings
	delete qsSettings;

	// Return value
	return qsRetValue;
}

//! This Function returns the version of the used pg.
    /*!
      \param none.
        \return A QString which holds pg version.
        */
QString sGenLib::pgVersion(QString qsInPath, QString qsInCpuName) {
	// Locals
	QString qsScratch, qsRetValue;

	// Create settings object
	qsScratch = qsInPath % qsInCpuName % "/" % qsLibSettings->value("FileNames/1",
"No entry").toString();
	qsScratch = qsScratch.replace("CpuName", qsInCpuName);
	qsSettings = new QSettings(qsScratch, QSettings::IniFormat, 0);

	// get data
	qsScratch = qsLibSettings->value("Groups/1", "No entry").toString() % "/" %
qsLibSettings->value("Keys/1", "No entry").toString();
	qsRetValue = qsSettings->value(qsScratch, "No entry").toString();

	// Delete QSettings
	delete qsSettings;

	// Return value
	return qsRetValue;
}
```

Appendix H Sources pTSocketLib

Appendix H I Source code pTSocketLib.h

```
#ifndef PTSOCKETLIB_H
#define PTSOCKETLIB_H

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/


//!  This is the pTSocketLib class, it is used to simplify the intertask communication
between the pTools.
/*!
  This Library is designed to extract data from the pTool-server.
*/

class pTSocketLib {
public:
	pTSocketLib();
	~pTSocketLib();

/**************** Commands Structure ***************/
//! This structure holds the Command and the sender.
/*!
   This structure is used to hold the Command and the sender of the command.
*/
	struct struCommand {
		QString qsProject;		/*!< Project.*/
		QString qsCpu;			/*!< Cpu.*/
		QString qsCommand;		/*!< Command.*/
		QString qsSender;		/*!< Sender.*/
		QString qsConHandle;	/*!< Connection Handle.*/
		QString qsAttribut;		/*!< Attribute.*/
	};

/************** Ende Commands Structure *************/


private:



};

#endif // PTSOCKETLIB_H
```

Appendix H II Source code ptsocketlib.cpp

```
/*****************************************************************************
 Implementation of the pTSocketLib DLL
 -------------------------------------

 Date         : 20.01.2012
 File       : pTSocketLib.cpp
 Author             : Thomas Gasser
 ©          : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's Socket Library.
/*!
  This Library implements the functionality for the intertask communication between
the pTools.
*/

/****************** Includes ************************/
#include "stdafx.h"                    // Precompiled headers
#include "ptsocketlib.h"          // Project header
/****************** Includes ************************/

// Constructor / Destructor
//! This is the Constructor of the pTSocketLib Library.
    /*!
      At the moment the Constructor of the pTSocketLib Library is not used at all.
    */
pTSocketLib::pTSocketLib() {

}

//! This is the Destructor of the pTSocketLib Library.
    /*!
      At the moment the Destructor of the pTSocketLib Library is not used at all.
    */
pTSocketLib::~pTSocketLib() {

}
```

Appendix H III Source code ptclientmachine.h

```
#ifndef PTCLIENTMACHINE_H
#define PTCLIENTMACHINE_H

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

/****************** Includes **********************/
#include <QObject>                    // QObject
#include <QtGui>                      // QtGui
#include "common.h"                   // Common things
#include <qstatemachine.h>         // QStateMachine
#include <qstate.h>                   // QState
#include <qsignaltransition.h>     // QSignalTransition
#include <qqueue.h>                   // QQueue
#include <QLocalSocket>               // QLocalSocket
/****************** Includes **********************/


class ptClientMachine : public QObject {
	Q_OBJECT

/******************* Publics **********************/
public:
	ptClientMachine(QObject *parent);
	~ptClientMachine();

	// Answer
	QStringList qslAnswer;

	// Interface functions
	void sendCommand(const struct struCommand *struInCmd);
	int setSender(QString qsInSender);
	void connectServer(QString qsInServerName);

	// Public signals
signals:
	void connectedToServer();
	void disconnectedFromServer();
	void answerReady();
	void serverConnected();

/****************** End Publics *********************/

/******************* Privates **********************/
private:
	// State machine
	QStateMachine *qsmClientMachine;
	QState *qsmsNotConnected;
	QState *qsmsConnected;
	QState *qsmsIdle;
	QState *qsmsSendCommand;
	QState *qsmsWaitForAnswer;
	QState *qsmsCheckAnswer;
	QState *qsmsError;
	QSignalTransition *qsmstConnected;
	QSignalTransition *qsmstDisconnected;

	// Timeout Timer
	QTimer *qtTimeout;
```

```
        // Repetition couter
        int iRepeatCounter;
        int iMaxRepeatCout;

        // Message Queue
        QQueue<QStringList> qqCommandQueue;

        // Attributes
        QString qsSender;

        // Local Socket
        QLocalSocket *serverConnection;

        // Structures
        struct struCommand struCmd;

private slots:
        void on_serverConnection_connected();
        void on_serverConnection_disconnected();
        void on_serverConnection_error(QLocalSocket::LocalSocketError socketError);
        void on_serverConnection_stateChanged(QLocalSocket::LocalSocketState
socketState);

        void on_timer_timeout();

        void if_machine_started();

        void on_qsmsNotConnected_entered();
        void on_qsmsNotConnected_exited();
        void on_qsmsConnected_entered();
        void on_qsmsConnected_exited();

        void on_qsmsIdle_entered();
        void on_qsmsIdle_exited();
        void on_qsmsSendCommand_entered();
        void on_qsmsSendCommand_exited();
        void on_qsmsWaitForAnswer_entered();
        void on_qsmsWaitForAnswer_exited();
        void on_qsmsCheckAnswer_entered();
        void on_qsmsCheckAnswer_exited();
        void on_qsmsError_entered();
        void on_qsmsError_exited();

signals:
        void commandToSend();
    void commandSent();
        void answerCorrect();
        void answerFaulty();
        void waitForRest();
        void error();
        void errorDealedWith();

/****************** End Privates *********************/
};

#endif // PTCLIENTMACHINE_H
```

Appendix H IV Source code ptclientmachine.cpp

```
/*****************************************************************************
 Implementation of the pTSocketLib client state machine
 ------------------------------------------------

 Date         : 20.01.2012
 File       : pTClientMachine.cpp
 Author             : Thomas Gasser
 ©          : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version    : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's Client State Machine.
/*!
  This is the Client State Machine.
*/

/****************** Includes ************************/
#include "ptclientmachine.h"            // Project header
/****************** Includes ************************/

// Constructor / Destructor
//! This is the Constructor of the Client State Machine.
    /*!
      At the moment the Constructor of the Client State Machine is not used at all.
    */
ptClientMachine::ptClientMachine(QObject *parent): QObject(parent) {
       // Locals
       bool bTest = false;

       // Timeout timer
       qtTimeout = new QTimer(this);
       qtTimeout->setInterval(10000);
       qtTimeout->setSingleShot(true);

       // Create Socket
       serverConnection = new QLocalSocket(this);

       // Connect Signals
       bTest = connect(serverConnection, SIGNAL(connected()), this,
SLOT(on_serverConnection_connected()));
       bTest = connect(serverConnection, SIGNAL(disconnected()), this,
SLOT(on_serverConnection_disconnected()));
       bTest = connect(serverConnection, SIGNAL(error(QLocalSocket::LocalSocketError)),
this,  SLOT(on_serverConnection_error(QLocalSocket::LocalSocketError)));
       bTest = connect(serverConnection,
SIGNAL(stateChanged(QLocalSocket::LocalSocketState)), this,
SLOT(on_serverConnection_stateChanged(QLocalSocket::LocalSocketState)));

       // State Machine
       qsmClientMachine = new QStateMachine(this);

       bTest = connect(qsmClientMachine, SIGNAL(started()), this,
SLOT(if_machine_started()));

       // Main States
       qsmsNotConnected = new QState();
       qsmsConnected = new QState();
```

```
        // Sub-States
        qsmsIdle = new QState(qsmsConnected);
        qsmsSendCommand = new QState(qsmsConnected);
        qsmsWaitForAnswer = new QState(qsmsConnected);
        qsmsCheckAnswer = new QState(qsmsConnected);
        qsmsError = new QState(qsmsConnected);
        qsmsConnected->setInitialState(qsmsIdle);

        // Add States
        qsmClientMachine->addState(qsmsNotConnected);
        qsmClientMachine->addState(qsmsConnected);

        // Set initial step
        qsmClientMachine->setInitialState(qsmsNotConnected);

        // Transitions
        qsmstConnected = new QSignalTransition();
        qsmstDisconnected = new QSignalTransition();

        // Configure Transitions
        qsmstConnected->setTargetState(qsmsNotConnected);
        qsmstConnected->setSenderObject(serverConnection);
        qsmstConnected->setSignal(SIGNAL(disconnected()));

        qsmstDisconnected->setTargetState(qsmsConnected);
        qsmstDisconnected->setSenderObject(serverConnection);
        qsmstDisconnected->setSignal(SIGNAL(connected()));

        // Add transitions to main states
        qsmsNotConnected->addTransition(qsmstDisconnected);
        qsmsConnected->addTransition(qsmstConnected);

        // Add transitions to sub states
        qsmsIdle->addTransition(this, SIGNAL(commandToSend()), qsmsSendCommand);

        qsmsSendCommand->addTransition(this, SIGNAL(commandSent()), qsmsWaitForAnswer);
        qsmsSendCommand->addTransition(this, SIGNAL(error()), qsmsError);

        qsmsWaitForAnswer->addTransition(qtTimeout, SIGNAL(timeout()), qsmsSendCommand);
        qsmsWaitForAnswer->addTransition(serverConnection, SIGNAL(readyRead()),
qsmsCheckAnswer);

        qsmsCheckAnswer->addTransition(this, SIGNAL(answerCorrect()), qsmsIdle);
        qsmsCheckAnswer->addTransition(this, SIGNAL(answerFaulty()), qsmsSendCommand);
        qsmsCheckAnswer->addTransition(this, SIGNAL(waitForRest()), qsmsWaitForAnswer);

        qsmsError->addTransition(this, SIGNAL(errorDealedWith()), qsmsIdle);

        // Establish connections of sub states
        bTest = connect(qsmsIdle, SIGNAL(entered()), this, SLOT(on_qsmsIdle_entered()));
        bTest = connect(qsmsIdle, SIGNAL(exited()), this, SLOT(on_qsmsIdle_exited()));

        bTest = connect(qsmsSendCommand, SIGNAL(entered()), this,
SLOT(on_qsmsSendCommand_entered()));
        bTest = connect(qsmsSendCommand, SIGNAL(exited()), this,
SLOT(on_qsmsSendCommand_exited()));

        bTest = connect(qsmsWaitForAnswer, SIGNAL(entered()), this,
SLOT(on_qsmsWaitForAnswer_entered()));
        bTest = connect(qsmsWaitForAnswer, SIGNAL(exited()), this,
SLOT(on_qsmsWaitForAnswer_exited()));
```

```
	bTest = connect(qsmsCheckAnswer, SIGNAL(entered()), this,
SLOT(on_qsmsCheckAnswer_entered()));
	bTest = connect(qsmsCheckAnswer, SIGNAL(exited()), this,
SLOT(on_qsmsCheckAnswer_exited()));

	bTest = connect(qsmsError, SIGNAL(entered()), this,
SLOT(on_qsmsError_entered()));
	bTest = connect(qsmsError, SIGNAL(exited()), this, SLOT(on_qsmsError_exited()));

	// Start machine
	qsmClientMachine->start();

	// Initialize Constants
	iMaxRepeatCout = 3;

	// Set defaults
	qslAnswer.clear();
	qqCommandQueue.clear();
}

//! This is the Destructor of the Client State Machine.
    /*!
      At the moment the Destructor of the Client State Machine is not used at all.
    */
ptClientMachine::~ptClientMachine() {

}

// ****************** Socket Functions ******************

void ptClientMachine::on_serverConnection_connected() {
	// Locals

	// Debug output
	qDebug() << "Connected to pTool";

	// Emit Signal
	emit connectedToServer();

	// Return value
	return;
}


void ptClientMachine::on_serverConnection_disconnected() {
	// Locals

	// Debug output
	qDebug() << "Disconnected from pTool";

	// Emit Signal
	emit disconnectedFromServer();

	// Return value
	return;
}


void ptClientMachine::on_serverConnection_error(QLocalSocket::LocalSocketError
socketError) {
	// Locals

	// Debug output
```

```
	qDebug() << serverConnection->errorString();

	// Return value
	return;
}


void ptClientMachine::on_serverConnection_stateChanged(QLocalSocket::LocalSocketState
socketState) {
	// Debug output
	switch(socketState) {
		// UnconnectedState
		case QLocalSocket::UnconnectedState:
			qDebug() << "Unconnected";
			break;

		// ConnectingState
		case QLocalSocket::ConnectingState:
			qDebug() << "Connecting...";
			break;

		// ConnectedState
		case QLocalSocket::ConnectedState:
			qDebug() << "Connected";
			break;

		// ClosingState
		case QLocalSocket::ClosingState:
			qDebug() << "Closing...";
			break;
	}
	// Return value
	return;
}

void ptClientMachine::if_machine_started() {
	// Locals

	// Debug output
	qDebug() << "ptclientmachine started";

	// Return value
	return;
}

// *************** End of Socket Functions ***************


// *************** Interface Functions ******************

//! This Function sends a command to the server.
    /*!
      \param iInCmd An Integer which holds the command.
      \return A QStringList with the answer to the command
    */
void ptClientMachine::sendCommand(const struct struCommand *struInCmd) {
	// Locals
	QStringList qslNewCommand;

	// struCmd.qsProject << struCmd.qsCpu << struCmd.qsCommand << struCmd.qsSender
<< struCmd.qsConHandle << << struCmd.qsAttribut
	qslNewCommand.clear();
	qslNewCommand << struInCmd->qsProject;
```

```
	qslNewCommand << struInCmd->qsCpu;
	qslNewCommand << struInCmd->qsCommand;
	qslNewCommand << struInCmd->qsSender;
	qslNewCommand << struInCmd->qsConHandle;
	qslNewCommand << struInCmd->qsAttribut;

	// Place command structur into queue
	qqCommandQueue.enqueue(qslNewCommand);

	// Emit signal
	emit commandToSend();

	// Return value
	return;
}

//! This Function sets the sender in the clientmachine.
    /*!
      \param qsInSender A QString holding the name of the Cpu where the active TO is
used.
      \return An Integer:   0 := Sender was set correctly,
                                                -1 := Error while setting the sender
    */
int ptClientMachine::setSender(QString qsInSender) {
	// Locals
	int iRetValue = 0;

	// Set Sender
	if(qsInSender.isEmpty() || qsInSender.isNull()) {
		// Set Sender
		qsSender = "";

		// Set return value
		iRetValue = -1;
	}

	else {
		// Set Sender
		qsSender = qsInSender;

		// Set return value
		iRetValue = 0;
	}

	// Return value
	return iRetValue;
}

//! This Function try to connect to the server.
    /*!
      \param qsServerName A QString which holds the server name.
      \return none
    */
void ptClientMachine::connectServer(QString qsInServerName) {
	// Locals

	// Check
	if(!qsInServerName.isEmpty() && !qsInServerName.isNull()) {
		// Connect / Disconnect
		if(serverConnection->state() != QLocalSocket::ConnectedState) {
			// Debug output
			qDebug() << "Try to connect to Server...";
```

```
                        // Try to connect to Server
                        serverConnection->connectToServer(qsInServerName,
QIODevice::ReadWrite);
                }
                else {
                        // Debug output
                        qDebug() << "Try to disconnect from pTool...";

                        // Disconnect from pTool
                        serverConnection->disconnectFromServer();
                }
        }
        else {
                // Debug output
                        qDebug() << "Invalide Servername";
        }
}

// ************** End of Interface Functions **************


// *************** Timer Timeout Function ***************

//! This Slot is dealing with a timer timeout.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_timer_timeout() {
        // Locals

        // Inkrement timeout counter
        iRepeatCounter++;

        // Debug message
                qDebug() << "Timeout" << iRepeatCounter;

        // Return value
        return;
}

// ************* End of Timer Timeout Function *************


// *************** Statemachine Functions ***************

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsNotConnected_entered(){
        // Debug message
        qDebug() << "Entering step: qsmsNotConnected";

        // Clear queue
        qqCommandQueue.clear();

        // Clear StringList
        qslAnswer.clear();

        // Return value
        return;
```

```
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsNotConnected_exited() {
       qDebug() << "Leaving step: qsmsNotConnected";

       // Return value
       return;
}

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsConnected_entered(){
       qDebug() << "Entering step: qsmsConnected";

       // Return value
       return;
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsConnected_exited() {
       qDebug() << "Leaving step: qsmsConnected";

       // Return value
       return;
}

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsIdle_entered() {
       // Debug message
       qDebug() << "Entered step: qsmsIdle";

       // Clear StringList
       qslAnswer.clear();

       // Check queue not to be empty
       if(!qqCommandQueue.isEmpty()) {
              // Remove head of queue and delete it
              qqCommandQueue.removeFirst();

              // Debug message
              qDebug() << "Removed head of queue";

              // Set repeat counter to 0
              iRepeatCounter = 0;
       }

       // Check if another command should be sent
```

```
	if(!qqCommandQueue.isEmpty()) {
		// Debug message
		qDebug() << "There are more commands to send...";

		// Emit signal
		emit commandToSend();
	}

	// Return value
	return;
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsIdle_exited() {
	qDebug() << "Leaving step: qsmsIdle";

	// Return value
	return;
}

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsSendCommand_entered() {
	// Locals
	QDataStream qdsSendStream(serverConnection);
	qdsSendStream.setVersion(QDataStream::Qt_4_8);
	QStringList qslCommand;
	QString qsScratch;

	// Inkrement repeat counter
	iRepeatCounter++;

	// Check Timeout Counter
	if(iRepeatCounter >= iMaxRepeatCout) {
		// Debug message
		qDebug() << "Max Timeout Count reached";
		// Emit signal
		emit error();
	}
	else {
		// Check queue not to be empty
		if(!qqCommandQueue.isEmpty()) {

			// Get command from queue
			qslCommand.clear();
			qslCommand = qqCommandQueue.head();

			// Debug output
			qDebug() << "ptClientMachine: Sending command";

			// Sendig Command
			while(!qslCommand.isEmpty()) {
				qsScratch = qslCommand.takeFirst();
				qdsSendStream << qsScratch;
			}
```

```
                        // Emit signal
                        emit commandSent();
                }
                else {
                        qDebug() << "Empty queue!";
                }
        }
        // Return value
        return;
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsSendCommand_exited() {
        qDebug() << "Leaving step: qsmsSendCommand";

        // Return value
        return;
}

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsWaitForAnswer_entered() {
        // Start Timeout timer
        qtTimeout->start();

        // Debug message
                qDebug() << "Waiting for answer...";

        // Return value
        return;
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsWaitForAnswer_exited() {
        qDebug() << "Leaving step: qsmsWaitForAnswer";

        // Return value
        return;
}

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsCheckAnswer_entered() {
        // Locals
        QDataStream qdsInputData(serverConnection);
        qdsInputData.setVersion(QDataStream::Qt_4_8);
        QString qsScratch, qsScratch2;
        QStringList qslTempList;
```

```
	// Show debug message and return
	qDebug() << "ptClientMachine: there is data to read...";

	// Read data
	while(serverConnection->bytesAvailable()) {
		if(!qdsInputData.atEnd()) {
			// Read Data
			qdsInputData >> qsScratch;

			// Fill answer into StringList
			qslAnswer << qsScratch;

			// Debug output
			qDebug() << "Value read: " << qsScratch;
		}
	}

	// Emit signals
	if(qslAnswer.isEmpty()) {
		// Inkrement counter
		iRepeatCounter++;

		// Debug message
		qDebug() << "Faulty answer" << iRepeatCounter;

		// No answer
		emit answerFaulty();
	}

	else if(!qslAnswer.contains("{") && !qslAnswer.contains("}")) {
		// Inkrement counter
		iRepeatCounter++;

		// Debug message
		qDebug() << "Faulty answer" << iRepeatCounter;

		// No answer
		emit answerFaulty();
	}

	else if(qslAnswer.contains("{") && !qslAnswer.contains("}")) { //
qslAnswer.size() < qslAnswer.at(1).toInt()
		// Debug message
		qDebug() << "Waiting for the rest of the answer...";

		// There should be more values to read
		emit waitForRest();
	}
	else {
		// Start Timeout timer
		qtTimeout->stop();

		// get real data
		for(int i = qslAnswer.indexOf("{"); i <= qslAnswer.indexOf("}"); ++i) {
			qsScratch2 = qslAnswer.at(i);
			qslTempList << qsScratch2;
		}

		qslAnswer.clear();
		qslAnswer << qslTempList;

		// Some answer recceived
		emit answerCorrect();
```

```
                emit answerReady();
        }

        // Return value
        return;
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsCheckAnswer_exited() {
        // Debug message
        qDebug() << "Leaving step: qsmsCheckAnswer";

        // Return value
        return;
}

//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsError_entered() {
        // Show debug message and return
        qDebug() << "ptClientMachine: An Error occured";

        // Clear String List
        qslAnswer.clear();

        // Fill error message into String List
        qslAnswer.append("An Error occured");

        // Clear message queue
        qqCommandQueue.clear();

        // Set repetition counter to 0
        iRepeatCounter = 0;

        // Emit Signal
        emit errorDealedWith();

        // Return value
        return;
}

//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptClientMachine::on_qsmsError_exited() {
        // Debug message
        qDebug() << "Leaving step: on_qsmsError_exited";

        // Return value
        return;
}

// ************* End of Statemachine Functions ************
```

Appendix H V Source code ptservermachine.h

```
#ifndef PTSERVERMACHINE_H
#define PTSERVERMACHINE_H

/****************** Defines ***********************/
#define QT_USE_FAST_CONCATENATION
/****************** Defines ***********************/

/****************** Includes **********************/
#include <QObject>                    // QObject
#include <QtGui>                      // QtGui
#include "common.h"                   // Common things
#include <qstatemachine.h>         // QStateMachine
#include <qstate.h>                   // QState
#include <qsignaltransition.h>     // QSignalTransition
#include <qqueue.h>                   // QQueue
#include <QLocalServer>               // QLocalServer
#include <QLocalSocket>               // QLocalSocket
/****************** Includes **********************/

class ptServerMachine : public QObject {
	Q_OBJECT

/******************* Publics **********************/
public:
	ptServerMachine(QObject *parent);
	~ptServerMachine();

	// Interface functions
	int setData(QStringList qslInAnswer);

	// Public signals
signals:
	void executeCommand(const struct struCommand *struInCmd);

/****************** End Publics *********************/

/******************* Privates **********************/
private:
	// State machine
	QStateMachine *qsmServerMachine;
	QState *qsmsNotConnected;
	QState *qsmsConnected;
	QState *qsmsIdle;
	QState *qsmsReceiveCommand;
	QState *qsmsCheckCommand;
	QState *qsmsExecuteCommand;
	QState *qsmsSendAnswer;
	QState *qsmsError;

	// Timeout Timer
	QTimer *qtTimeout;

	// Repetition couter
	int iRepeatCounter;
	int iMaxRepeatCount;

	// Commandlength
	int iCommandLength;

	// Command StringList
```

```
        QStringList qslCommand;

        // Message Queues
        QQueue<QStringList> qqDataQueue;
        QQueue<QStringList> qqCommandQueue;

        // Structures
        struct struCommand struCmd;

        // Local Server
        QLocalServer *qlsLocalServer;
        QLocalSocket *clientConnection;
        QHash<QString, QLocalSocket *> qhClientConnection;


private slots:
        // Local Server Functions
        void if_qlsServer_new_connection();
        void if_clientConnection_readyRead();
        void if_clientConnection_stateChanged( QLocalSocket::LocalSocketState
socketState);

        void on_timer_timeout();

        void if_machine_started();

        void on_qsmsNotConnected_entered();
        void on_qsmsNotConnected_exited();
        void on_qsmsConnected_entered();
        void on_qsmsConnected_exited();

        void on_qsmsIdle_entered();
        void on_qsmsIdle_exited();
        void on_qsmsReceiveCommand_entered();
        void on_qsmsReceiveCommand_exited();
        void on_qsmsCheckCommand_entered();
        void on_qsmsCheckCommand_exited();
        void on_qsmsExecuteCommand_entered();
        void on_qsmsExecuteCommand_exited();
        void on_qsmsSendAnswer_entered();
        void on_qsmsSendAnswer_exited();
        void on_qsmsError_entered();
        void on_qsmsError_exited();

signals:
        void connectionReady();
        void connectionClosed();

        void commandToReceive();
    void commandReceived();
        void waitForRest();
        void commandCorrect();
        void commandFaulty();
        void commandExecuted();
        void answerFaulty();
        void answerSent();
        void moreCommands();
        void error();
        void errorDealedWith();

/****************** End Privates ********************/
};
#endif // PTSERVERMACHINE_H
```

Appendix H VI Source code ptservermachine.cpp

```
/*****************************************************************************
 Implementation of the pTSocketLib server state machine
 ------------------------------------------------

 Date         : 24.03.2012
 File        : pTServerMachine.cpp
 Author              : Thomas Gasser
 ©           : 2012 by Thomas Gasser
 eMail        : t.gasser@bluemail.ch
 Version     : 1.0.0.0
 Changes      :

*****************************************************************************/

//! TG's Server State Machine.
/*!
  This is the Server State Machine.
*/

/****************** Includes ************************/
#include "ptservermachine.h"             // Project header
/****************** Includes ************************/

// Constructor / Destructor
//! This is the Constructor of the Client State Machine.
    /*!
      At the moment the Constructor of the Client State Machine is not used at all.
    */

ptServerMachine::ptServerMachine(QObject *parent): QObject(parent) {
       // Local's
       bool bTest = false;

       // Create and start local server
       qlsLocalServer = new QLocalServer(this);
       qlsLocalServer->setMaxPendingConnections(1);

       if (!qlsLocalServer->listen("pTool")) {
              QMessageBox::critical(0, tr("Local-Server"),tr("Unable to start the
server: %1").arg(qlsLocalServer->errorString()));
              return;
    }

       // Connect Server Signal
       bTest = connect(qlsLocalServer, SIGNAL(newConnection()), this,
SLOT(if_qlsServer_new_connection()));

       // Timeout timer
       qtTimeout = new QTimer(this);
       qtTimeout->setInterval(10000);
       qtTimeout->setSingleShot(true);

       // State Machine
       qsmServerMachine = new QStateMachine(this);
       bTest = connect(qsmServerMachine, SIGNAL(started()), this,
SLOT(if_machine_started()));

       // Main States
       qsmsNotConnected = new QState();
       qsmsConnected = new QState();
```

```
        // Sub-States
        qsmsIdle = new QState(qsmsConnected);
        qsmsReceiveCommand = new QState(qsmsConnected);
        qsmsReceiveCommand = new QState(qsmsConnected);
        qsmsCheckCommand = new QState(qsmsConnected);
        qsmsExecuteCommand = new QState(qsmsConnected);
        qsmsSendAnswer = new QState(qsmsConnected);
        qsmsError = new QState(qsmsConnected);
        qsmsConnected->setInitialState(qsmsIdle);

        // Add States
        qsmServerMachine->addState(qsmsNotConnected);
        qsmServerMachine->addState(qsmsConnected);

        // Set initial step
        qsmServerMachine->setInitialState(qsmsNotConnected);

        // Add transitions to main states
        qsmsNotConnected->addTransition(this, SIGNAL(connectionReady()), qsmsConnected);
        qsmsConnected->addTransition(this, SIGNAL(connectionClosed()),
qsmsNotConnected);

        // Add transitions to sub states
        qsmsIdle->addTransition(this, SIGNAL(commandToReceive()), qsmsReceiveCommand);

        qsmsReceiveCommand->addTransition(this, SIGNAL(commandReceived()),
qsmsCheckCommand);
        qsmsReceiveCommand->addTransition(this, SIGNAL(error()), qsmsError);

        qsmsCheckCommand->addTransition(this, SIGNAL(commandCorrect()),
qsmsExecuteCommand);
        qsmsCheckCommand->addTransition(this, SIGNAL(commandFaulty()), qsmsError);
        qsmsCheckCommand->addTransition(this, SIGNAL(waitForRest()),
qsmsReceiveCommand);

        qsmsExecuteCommand->addTransition(this, SIGNAL(commandExecuted()),
qsmsSendAnswer);
        qsmsExecuteCommand->addTransition(qtTimeout, SIGNAL(timeout()), qsmsError);

        qsmsSendAnswer->addTransition(this, SIGNAL(answerSent()), qsmsIdle);
        qsmsSendAnswer->addTransition(this, SIGNAL(moreCommands()), qsmsCheckCommand);
        qsmsSendAnswer->addTransition(this, SIGNAL(answerFaulty()), qsmsError);

        qsmsError->addTransition(this, SIGNAL(errorDealedWith()), qsmsSendAnswer);

        // Establish connections of sub states
        bTest = connect(qsmsIdle, SIGNAL(entered()), this, SLOT(on_qsmsIdle_entered()));
        bTest = connect(qsmsIdle, SIGNAL(exited()), this, SLOT(on_qsmsIdle_exited()));

        bTest = connect(qsmsReceiveCommand, SIGNAL(entered()), this,
SLOT(on_qsmsReceiveCommand_entered()));
        bTest = connect(qsmsReceiveCommand, SIGNAL(exited()), this,
SLOT(on_qsmsReceiveCommand_exited()));

        bTest = connect(qsmsCheckCommand, SIGNAL(entered()), this,
SLOT(on_qsmsCheckCommand_entered()));
        bTest = connect(qsmsCheckCommand, SIGNAL(exited()), this,
SLOT(on_qsmsCheckCommand_exited()));

        bTest = connect(qsmsExecuteCommand, SIGNAL(entered()), this,
SLOT(on_qsmsExecuteCommand_entered()));
```

```
	bTest = connect(qsmsExecuteCommand, SIGNAL(exited()), this,
SLOT(on_qsmsExecuteCommand_exited()));

	bTest = connect(qsmsSendAnswer, SIGNAL(entered()), this,
SLOT(on_qsmsSendAnswer_entered()));
	bTest = connect(qsmsSendAnswer, SIGNAL(exited()), this,
SLOT(on_qsmsSendAnswer_exited()));

	bTest = connect(qsmsError, SIGNAL(entered()), this,
SLOT(on_qsmsError_entered()));
	bTest = connect(qsmsError, SIGNAL(exited()), this, SLOT(on_qsmsError_exited()));

	// Start machine
	qsmServerMachine->start();

	// Initialize Constants
	iMaxRepeatCount = 3;
	iRepeatCounter = 0;
	iCommandLength = 6;
}


//! This is the Destructor of the pTServerMachine application.
    /*!
        Here some stuff is done just before the application is finished
    */
ptServerMachine::~ptServerMachine() {

}


// ****************** Server Functions ******************
//! This Function is called if a new pending connection to the pTool-Server is made.
    /*!
        \return void
    */
void ptServerMachine::if_qlsServer_new_connection() {
	// Locals
	bool bTest;

	// Show Message Box and return
	qDebug() << "Local-Server: New pending connection";

	// Client connection
	clientConnection = new QLocalSocket(this);
	qhClientConnection.insert("Test", clientConnection);

	// Get pending conecction
	clientConnection = qlsLocalServer->nextPendingConnection();
	bTest = connect(clientConnection, SIGNAL(readyRead()), this,
SLOT(if_clientConnection_readyRead()));
	bTest = connect(clientConnection,
SIGNAL(stateChanged(QLocalSocket::LocalSocketState)), this,
SLOT(if_clientConnection_stateChanged(QLocalSocket::LocalSocketState)));

	// Debug output
	qDebug() << "Socket Descriptor new Connection: " << clientConnection-
>socketDescriptor();

	// Emit signal
	emit connectionReady();

	// Return value
```

```
	return;
}


//! This Function is called if there is new data to read on the Local Socket.
    /*!
         \return void
    */
void ptServerMachine::if_clientConnection_readyRead() {
	// Show debug message
	qDebug() << "Local-Server: there is data to read...";

	// Emit Signal
	emit commandToReceive();

	// Return value
	return;

}


//! This Slot is called if the state of the local socket is changed.
    /*!
         \return void
    */
void  ptServerMachine::if_clientConnection_stateChanged(
QLocalSocket::LocalSocketState socketState) {
	// Show debug message
	qDebug() << "Local-Server: state changed...";

	// Check socket state
	switch(socketState) {
		case QLocalSocket::ClosingState:
			// Debug message
			qDebug() << "Local-Server: closing...";
		break;

		case QLocalSocket::UnconnectedState:
			// Debug message
			qDebug() << "Local-Server: unconnected";

			// Emit signal
			emit connectionClosed();
		break;

		case QLocalSocket::ConnectingState:
			// Debug message
			qDebug() << "Local-Server: connecting...";
		break;

		case QLocalSocket::ConnectedState:
			// Debug message
			qDebug() << "Local-Server: connected";
		break;

		default:
			// Debug message
			qDebug() << "Local-Server: unknow state";
		break;
	}

	// Return value
	return;
```

```
}


// *************** End of Server Functions ***************

// ***************** Timer Timeout Function ****************

//! This Slot is dealing with a timer timeout.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_timer_timeout() {
       // Locals

       // Increment timeout counter
       iRepeatCounter++;

       // Debug message
              qDebug() << "Timeout " << iRepeatCounter;

       // Return value
       return;
}


// ************* End of Timer Timeout Function *************


// ***************** Statemachine Functions ****************

//! This Slot is dealing with stuff when the server mchine is started.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::if_machine_started() {
       // Debug output
       qDebug() << "ptservermachine started";

       // Return value
       return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsNotConnected_entered() {
       qDebug() << "Entering step: qsmsNotConnected";

       if(clientConnection->state() == QLocalSocket::UnconnectedState) {
       // Delete client connection
              delete clientConnection;
       }

       // Return value
       return;
}
```

```
//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsNotConnected_exited() {
       qDebug() << "Leaving step: qsmsNotConnected";

       // Return value
       return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsConnected_entered() {
       qDebug() << "Entering step: qsmsConnected";

       // Return value
       return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsConnected_exited() {
       qDebug() << "Leaving step: qsmsConnected";

       // Return value
       return;
}


// *************** Interface Functions ******************

//! This Function sets the processed data (answer).
    /*!
      \param qslAnswer A QStringList which holds the data (answer).
      \return An Integer    0 := Setting answer successful,
                                              -1 := Setting answer unsuccessful
    */
int ptServerMachine::setData(QStringList qslInAnswer) {
       // Locals
       int iRetValue;

       // Check input
       if(!qslInAnswer.isEmpty()) {
              // Shift answer into queue
              qqDataQueue.enqueue(qslInAnswer);

              // emit Signal
              emit commandExecuted();

              // Set return value
              iRetValue = 0;
       }
       else {
```

```
                // emit Signal
                emit answerFaulty();

                // Set return value
                iRetValue = -1;
        }

        // Clear answer list
        qslInAnswer.clear();

        // Stopp timeout timer
        qtTimeout->stop();

        // Return value
        return iRetValue;
}


// ************** End of Interface Functions **************


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsIdle_entered() {
        // Debug message
        qDebug() << "Entering step: qsmsIdle";

        // Clear command list
        qslCommand.clear();

        // Clear queue
        if(!qqDataQueue.isEmpty()) {
                qqDataQueue.clear();
        }

        // Clear repeat counter
        iRepeatCounter = 0;

        // Return value
        return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsIdle_exited() {
        qDebug() << "Leaving step: qsmsIdle";

        // Return value
        return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
```

```
    */
void ptServerMachine::on_qsmsReceiveCommand_entered() {
	// Debug message
	qDebug() << "Entering step: qsmsReceiveCommand";

	// Locals
	QDataStream qdsInputData(clientConnection);
	qdsInputData.setVersion(QDataStream::Qt_4_8);
	QString qsScratch;
	QStringList qslTempList;

	// Check repeat counter
	if(iRepeatCounter <= iMaxRepeatCount) {
		// Read data
		while(clientConnection->bytesAvailable()) {
			if(!qdsInputData.atEnd()) {
				// Read command
				qdsInputData >> qsScratch;

				// Fill command into StringList
				qslCommand << qsScratch;
			}
		}

		// Check command
		if(qslCommand.size() == iCommandLength) {
			// Debug message
			qDebug() << "Single command received";

			// Move command to the queue
			qqCommandQueue.enqueue(qslCommand);
		}
		else if(qslCommand.size() < iCommandLength){
			// Debug message
			qDebug() << "Parts of a command received";
			// Move command to the queue
			qqCommandQueue.enqueue(qslCommand);
		}
		else {
			// Debug message
			qDebug() << "More than one command received";

			// Split commands
			while(!qslCommand.isEmpty()) {
				// Get one command
				for(int i = 0; i < iCommandLength; ++i) {
					qsScratch = qslCommand.takeFirst();
					qslTempList << qsScratch;
				}
				// Move command to the queue
				qqCommandQueue.enqueue(qslTempList);

				// Clear tem list
				qslTempList.clear();
			}
		}

		// Emit Signal
		emit commandReceived();
	}
	else {
		// Debug message
		qDebug() << "Too many repetitions";
```

```
		// emit Signal
		emit error();
	}

	// Return value
	return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsReceiveCommand_exited() {
	qDebug() << "Leaving step: qsmsReceiveCommand";

	// Return value
	return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsCheckCommand_entered() {
	// Debug message
	qDebug() << "Entering step: qsmsCheckCommand";

	// Locals
	QStringList qslTempList1, qslTempList2;

	// Check size of command queue
	if(qqCommandQueue.size() > 1) {
		// Get heads of command queue
		qslTempList1 = qqCommandQueue.dequeue();
		qslTempList2 = qqCommandQueue.dequeue();

		// Compare lists
		if(qslTempList1.operator!=(qslTempList2)) {
			// Debug message
			qDebug() << "Commands are different";

			// Move command 2 back to the queue
			qqCommandQueue.enqueue(qslTempList2);
		}
		else {
			// Debug message
			qDebug() << "Commands are the same";

			// delete command 2
			qslTempList2.clear();
		}
	}
	else {
		// Get heads of command queue
		qslTempList1 = qqCommandQueue.dequeue();
	}

	// Get Command out of the StringList
```

```
	if(qslTempList1.size() == iCommandLength) {
		struCmd.qsProject = qslTempList1.at(0);
		struCmd.qsCpu = qslTempList1.at(1);
		struCmd.qsCommand = qslTempList1.at(2);
		struCmd.qsSender = qslTempList1.at(3);
		struCmd.qsConHandle = qslTempList1.at(4);
		struCmd.qsAttribut = qslTempList1.at(5);

		// Clear repeat counter
		iRepeatCounter = 0;

		// Debug output
		qDebug() << "Command is for Project: " << struCmd.qsProject;
		qDebug() << "Command is for Cpu: " << struCmd.qsCpu;
		qDebug() << "Command: " << struCmd.qsCommand;
		qDebug() << "Sender: " << struCmd.qsSender;
		qDebug() << "Connection Handle: " << struCmd.qsConHandle;
		qDebug() << "Attribut: " << struCmd.qsAttribut;

		// Emit Signal
		emit commandCorrect();
	}
	else if(qslTempList1.size() < iCommandLength) {
		// Debug message
		qDebug() << "Command not complete...";

		// Set command list
		qslCommand = qslTempList1;

		// Increment repeat counter
		iRepeatCounter++;

		// emit Signal
		emit waitForRest();
	}
	else {
		// Debug message
		qDebug() << "Command faulty";

		// emit Signal
		emit commandFaulty();
	}

	// Return value
	return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsCheckCommand_exited() {
	qDebug() << "Leaving step: qsmsCheckCommand";

	// Return value
	return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
```

```
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsExecuteCommand_entered() {
       qDebug() << "Entering step: qsmsExecuteCommand";

       // Give Command to the app
       emit executeCommand(&struCmd);

       // Clear repeat counter
       iRepeatCounter = 0;

       // Start timeout timer
       qtTimeout->start();

       // Return value
       return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsExecuteCommand_exited() {
       qDebug() << "Leaving step: qsmsExecuteCommand";

       // Return value
       return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsSendAnswer_entered()  {
       // Debug message
       qDebug() << "Entering step: qsmsSendAnswer";

       // Locals
       QDataStream qdsSendStream(clientConnection);
       qdsSendStream.setVersion(QDataStream::Qt_4_8);
       QStringList qslAnswer;
       QString qsScratch;

       // Start timeout timer
       qtTimeout->stop();

       // Check queue not to be empty
       if(!qqDataQueue.isEmpty()) {

              // Get command from queue
              qslAnswer.clear();
              qslAnswer = qqDataQueue.dequeue();

              // Debug output
              qDebug() << "ptServerMachine: Sending answer";

              // Insert start and stopp brackets
              qslAnswer.prepend("{");
```

```
                qslAnswer.append("}");

                // Sendig Command
                while(!qslAnswer.isEmpty()) {
                        qsScratch = qslAnswer.takeFirst();
                        qdsSendStream << qsScratch;
                }

                // Check if there are more commands in the queue
                if(!qqCommandQueue.isEmpty()) {
                        // Emit signal
                        emit moreCommands();
                }
                else {
                        // Emit signal
                        emit answerSent();
                }
        }
        else {
                // Debug message
                qDebug() << "Empty queue!";

                // Emit signal
                emit answerFaulty();
        }

        // Return value
        return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsSendAnswer_exited() {
        // Debug message
        qDebug() << "Leaving step: qsmsSendAnswer";

        // Return value
        return;
}


//! This Slot is dealing with stuff when entering the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsError_entered() {
        // Debug message
        qDebug() << "Entering step: qsmsError";

        // Locals
        QStringList qslAnswer;
        QVariant qvAnswerSize;

        // Clear answer list
        qslAnswer.clear();

        // Fill answer list
        qslAnswer << "ERROR";
```

```
        // Get answer size
        qvAnswerSize = qslAnswer.size();

        // Insert answer list size
        qslAnswer.prepend(qvAnswerSize.toString());

        // Put answer into queue
        qqDataQueue.clear();
        qqDataQueue.enqueue(qslAnswer);

        // emit Signal
        emit errorDealedWith();

        // Return value
        return;
}


//! This Slot is dealing with stuff when exiting the step.
    /*!
      \param none.
      \return none.
    */
void ptServerMachine::on_qsmsError_exited() {
        // Debug message
        qDebug() << "Leaving step: qsmsError";

        // Return value
        return;
}

// ************* End of Statemachine Functions ************
```

Appendix H VII Source code common.h

```
ifndef COMMON_H
#define COMMON_H

/*************** Commands Structure ***************/
//! This structure holds the Command and the sender.
/*!
   This structure is used to hold the Command and the sender of the command.
*/
       struct struCommand {
              QString qsProject;               /*!< Project.*/
              QString qsCpu;                          /*!< Cpu.*/
              QString qsCommand;               /*!< Command.*/
              QString qsSender;                /*!< Sender.*/
              QString qsConHandle;       /*!< Connection Handle.*/
              QString qsAttribut;              /*!< Attribute.*/
       };

/************** Ende Commands Structure *************/

#endif // COMMON_H
```

Appendix I Generated IL-Code

```
; ********************************************************************************
;
; Code generated by TG
; --------------------
;
; Datum          : Freitag 05. April 2013 11:01:43
; SPS            : PCD3.M5540
; Datei          : tGVm.src
; PG5-Version    : V2.0.220.200
; Autor          : Saia Generator by TG
; © : 2013 by TG; Ida-Sträuli-Strasse 65; CH-8404 Winterthur
;
; Version        : 1.0.0.0
; Aenderungen    : -
;
; ********************************************************************************


; ******************* Station ************************          ; Programm gehört auf diese S-Bus Station
$STATION 10
; ***************** Ende Station **********************


; ******************** XOB 16 ************************          ; Beim RestartCold ausgeführte Befehle
$INIT
$ENDINIT
; ***************** Ende XOB 16 **********************


; ******************* Includes ***********************

; ***************** Ende Includes ********************


; ***************** Definitionen *********************
; ********** PEQU PB                                          ; Block definieren
COB_tGVm      PEQU   COB                                      ; PEQU Block (EQU + Public)
```

```
; ********** import Variablen                    ; aus anderen PB's importieren
        EXTN   LTO01                             ; Import Symbol
        EXTN   VM001                             ; Import Symbol


$GROUP tGVm                                      ; Gruppendeklaration
; ********** Variablen                           ; Statische, lokale Variablen erstellen


; ************** Ende Definitionen ******************


; ****************** Start COB **********************

        COB    COB_tGVm                          ; Start COB
               0                                 ; No Timeout


; ************ Memory for temporary Data ************
TEMP_MEM:

DEFTMP M 2                                       ; defines 2K bytes of memory for the temp data stack

END_TEMP_MEM:
; ********* Ende Memory for temporary Data **********


; ****************** Temporary Data *****************
TEMP_DATA:

END_TEMP_DATA:
; ************** Ende Temporary Data ****************


; ******************** LTO's ************************
LTOS:
; ********** Resources
```

```
rHrLTO01        EQU     R[4]                            ; Helper Registers

CFG_LTO00       EQU     DB 4007                         ; Datablock for:
DB CFG_LTO00 [32] 1, 9, 0, 0, 0, 0, 0, 0, 0, 0, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

CFG_LTO01       EQU     DB 4008                         ; Datablock for: Register
DB CFG_LTO01 [32] 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

CFG_LTO02       EQU     DB 4000                         ; Datablock for: Register
DB CFG_LTO02 [32] 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2200, 300, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0

CFG_LTO03       EQU     DB 4001                         ; Datablock for: Register
DB CFG_LTO03 [32] 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2201, 301, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0

; ********** Call FB

        CFB     LTO01                                   ; Call LTO01
                DB CFG_LTO00                            ; [2]DB Address
                DB CFG_LTO00                            ; [3]DB Address
                H09.LG.CTRL_001.CFG_Pointer             ; [4]CFG_Pointer
                H09.LG.CTRL_001.OUT_Output              ; [5]OUT_Output
                H09.LG.CTRL_001.OUT_OutputInvers        ; [6]OUT_OutputInvers
                H09.LG.CTRL_001.OUT_Output              ; [7]OUT_Output
                H09.LG.CTRL_001.OUT_OutputInvers        ; [8]OUT_OutputInvers
                rHrLTO01+0                              ; [9]Register

; ********** Call FB

        CFB     LTO01                                   ; Call LTO01
                DB CFG_LTO01                            ; [2]DB Address
                DB CFG_LTO01                            ; [3]DB Address
```

```
                H09.LG.CTRL_002.CFG_Pointer         ; [4]CFG_Pointer
                H09.LG.CTRL_002.OUT_Output          ; [5]OUT_Output
                H09.LG.CTRL_002.OUT_OutputInvers ; [6]OUT_OutputInvers
                H09.LG.CTRL_002.OUT_Output          ; [7]OUT_Output
                H09.LG.CTRL_002.OUT_OutputInvers ; [8]OUT_OutputInvers
                rHrLTO01+1                          ; [9]Register


; ********** Call FB

        CFB     LTO01                               ; Call LTO01
                DB CFG_LTO02                        ; [2]DB Address
                DB CFG_LTO02                        ; [3]DB Address
                H20.LG.CTRL.LOGIC_001.CFG_Pointer       ; [4]CFG_Pointer
                H20.LG.CTRL.LOGIC_001.OUT_Output        ; [5]OUT_Output
                H20.LG.CTRL.LOGIC_001.OUT_OutputInvers  ; [6]OUT_OutputInvers
                H20.LG.CTRL.LOGIC_001.OUT_Output        ; [7]OUT_Output
                H20.LG.CTRL.LOGIC_001.OUT_OutputInvers  ; [8]OUT_OutputInvers
                rHrLTO01+2                              ; [9]Register


; ********** Call FB

        CFB     LTO01                               ; Call LTO01
                DB CFG_LTO03                        ; [2]DB Address
                DB CFG_LTO03                        ; [3]DB Address
                H21.LG.CTRL.LOGIC_001.CFG_Pointer       ; [4]CFG_Pointer
                H21.LG.CTRL.LOGIC_001.OUT_Output        ; [5]OUT_Output
                H21.LG.CTRL.LOGIC_001.OUT_OutputInvers  ; [6]OUT_OutputInvers
                H21.LG.CTRL.LOGIC_001.OUT_Output        ; [7]OUT_Output
                H21.LG.CTRL.LOGIC_001.OUT_OutputInvers  ; [8]OUT_OutputInvers
                rHrLTO01+3                              ; [9]Register


END_LTO:
; ****************** Ende LTO's **********************


; ********************** VM's ************************
VM:
```

```
; ********** Resources

rHrVM001        EQU     R[5]                                    ; Helper Registers

CFG_VM01        EQU     DB 4011                                 ; Datablock for: Register
DB CFG_VM01 [20] 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2006, 2208, 2209, 2210, 0, 0, 0, 0, 0, 0



; ********** Call FB

        CFB     VM001                                   ; Virtual Machine to be used with the LTO01
                DB CFG_VM01                     ; [2]DB Address
                 DB CFG_VM01                            ; [3]DB Address
                H20.LG.CTRL.VM_001.CFG_Pointer  ; [4]CFG_Pointer
                rHrVM001+0                              ; [5]Register
                rHrVM001+1                              ; [6]Register
                rHrVM001+2                              ; [7]Register
                rHrVM001+3                              ; [8]Register
                rHrVM001+4                              ; [9]Register

END_VM:
; ****************** Ende VM's **********************


END_COB:

ECOB


; ****************** Ende COB **********************
$ENDGROUP
```

Appendix J Configuration files

Appendix J I pTool.ini

```
[LastProject]
Path=C:/PromosNT/proj/tgTest/cfg/
Name=tgTest

[ToolPath]
Path=C:/promosnt/proj/

[Tools]
1=Saia-&Code-Schleuder
2=&Kommunikation
3=&Nicht-Auto

[FileNames]
DMS=promos.dms
TO=bmo.dms
```

Appendix J II sGen.ini

```
[SymbolHeader]
Header="Symbol;Media;Address;Comment := Value ;Scope"


[VmTemplates]
srcHeader=../cfg/srcVmHeader.ini
srcFooter=../cfg/srcVmFooter.ini


[LTO01GEN]
1=CFG_Function
2=CFG_NbrOfUsedInputs
3=rHrLTO01


[LTO01]
1="; ********** Call FB\n\n\tCFB\tLTO01\t; Call LTO01"
2=CFG_Function
3=CFG_Function
4=CFG_Pointer
5=OUT_Output
6=OUT_OutputInvers
7=OUT_Output
8=OUT_OutputInvers
9=Register


[LTO01FOOTER]
1="END_LTO:\n; ****************** Ende LTO's **********************"


[VM001HEADER]
1="; ********************* VM's ************************\nVM:\n; ********** Resources"


[VM001GEN]
1=CFG_CycleTime
2=CFG_LTO_Cnt
```

```
3=CFG_LTO_Pointer
4=rHrVM001



[VM001]
1="; ********** Call FB\n\n\tCFB\tVM001\t; Virtual Machine to be used with the LTO01"
2=CFG_CycleTime
3=CFG_CycleTime
4=CFG_Pointer
5=Register
6=Register
7=Register
8=Register
9=Register


[VM001FOOTER]
1="END_VM:\n; ****************** Ende VM's **********************"


[VMADRLOADER]
1="; ********** Load LTO Ardesses\n\n\tCFB\tfbLoadAdr\t; Call Function Load Addresses"
2=CFG_VM00
3=IN_LTO00
4=IN_LTO01
5=IN_LTO02
6=IN_LTO03
7=IN_LTO04
8=IN_LTO05
9=IN_LTO06
10=IN_LTO07
11=IN_LTO08
12=IN_LTO09


[COBFOOTER]
1="END_COB:\n\n\tECOB\n\n\n; ****************** Ende COB **********************\n$ENDGROUP"
```

Appendix J III sTool.ini

```
[FileNames]
1=CpuName.saia5pc
2=PCD.SCFG
3=CpuName.5hw

[Groups]
1=Device
2=ConfigNT
3=PresistentSettings

[Keys]
1=PG5Version
2=PCDType
3=sbus_Station
```

Appendix J IV srcVmHeader.ini

```
; ********************************************************************************
;
; Code generated by TG
; --------------------
;
; Datum          : <DateTime>
; SPS            : <PcdType>
; Datei          : <FileName>
; PG5-Version    : <PgVersion>
; Autor          : Saia Generator by TG
; © : <Year> by TG; Ida-Sträuli-Strasse 65; CH-8404 Winterthur
;
; Version        : <Version>
; Aenderungen    : <Changes>
;
; ********************************************************************************


; ******************* Station ***********************          ; Programm gehört auf diese S-Bus Station
<Station>
; ***************** Ende Station **********************


; ******************** XOB 16 ************************          ; Beim RestartCold ausgeführte Befehle
$INIT
$ENDINIT
; ***************** Ende XOB 16 **********************


; ******************* Includes ***********************

; ***************** Ende Includes ********************


; ***************** Definitionen *********************
; ********** PEQU PB                                            ; Block definieren
COB_<blockName>      PEQU <blockTyp>                             ; PEQU Block (EQU + Public)
```

```
; ********** import Variablen                    ; aus anderen PB's importieren
        EXTN   LTO01                             ; Import Symbol
        EXTN   VM001                             ; Import Symbol


$GROUP <groupName>                               ; Gruppendeklaration
; ********** Variablen                           ; Statische, lokale Variablen erstellen


; ************** Ende Definitionen ******************


; ****************** Start COB **********************

        COB COB_<blockName>                      ; Start COB
            0                                    ; No Timeout


; ************ Memory for temporary Data ************
TEMP_MEM:

DEFTMP M 2                                       ; defines 2K bytes of memory for the temp data stack

END_TEMP_MEM:
; ********* Ende Memory for temporary Data **********


; ****************** Temporary Data *****************
TEMP_DATA:

END_TEMP_DATA:
; ************** Ende Temporary Data ****************


; ******************** LTO's ************************
LTOS:
; ********** Resources
```

Appendix J V srcVmFooter.ini

```
END_VM:
; ****************** Ende VM ************************


; Ende PB
END:
        ECOB                                    ; Ende PB
$ENDGROUP
```

Appendix K Versions PLC-System

Appendix K I PLC-System

Product Information
PCD Type: PCD3.M5540
System ID: M5540 (55h)
Firmware Version: 1.16.69
Hardware Version: D 2 (02h)
Modification: 2 (02h)
Fab. Date (year/week): 2007/15
Serial Number: 00BC3824
MAC-Address: 00 50 C2 6F DB A7

CPU Extension
ASN: PCD3.EXT0
Part Number: 463667920
Serial Number: 00BDDB41
HW Version: B (0x42)
Modifications: None (0xFF)
Fab. Date (year/week): 2007/14

Flash M1
ASN: PCD7.R550M04
Part Number: 463949000
Serial Number: 00C04100
HW Version: B (0x42)
Modifications: None (0xFF)
Fab. Date (year/week): 2007/15

Flash M2
ASN: PCD7.R500
Part Number: 463948970
Serial Number: 01266217
HW Version: B (0x42)
Modifications: None (0xFF)
Fab. Date (year/week): 2007/50

Battery Module
ASN: PCD3.BAT0
Part Number: 463948980
Serial Number: 00A68E91
HW Version: A (0x41)
Modifications: None (0xFF)
Fab. Date (year/week): 2007/08

Appendix K II Programming Software PG5

PG5 Patch 2 Installed

AdjustWnd52.dll 2.0.220.0 OK Saia PG5 V2 Adjust Window, by Gábor Domonyi

AGLink40_Sym.DLL 4.0.12.8 ACCON-AGLink S7-SymbolikPro 4.x for SIMATIC PLCs [.\Web Editor 5_15_02]

AutoRouteLib.dll V2.0.220.200 OK Saia Fupla Editor AutoRoute Library

Core.dll 2.0.20.0 [.\SWebConnect]

DDC_Add-On.exe 2.0.220.0 OK

DDC_AddOnLib.dll 2.0.220.0 OK

DUNZIP32.dll 4.00.04 OK DynaZIP-32 Multi-Threading UnZIP DLL

Dxusb10.dll 1.0.9.0 DriverX USB DLL

Dxusb10.dll 1.0.9.0 DriverX USB DLL [.\SWebConnect]

DZIP32.dll 4.00.04 OK DynaZIP-32 Multi-Threading ZIP DLL
EnvDTE.dll 8.0.50727.42(RT OK EnvDTE.dll
ExternalCore.dll 2.0.20.0 [.\SWebConnect]
FBE.exe 2.0.239.0 SAIA FBox Builder [.\FBox Builder]
FBELib.dll 2.0.239.0 SAIA FBE Library [.\FBox Builder]
FBLM.exe 2.0.239.0 SAIA FBox Key Generator [.\FBox Builder]
hcrtf.exe 4.03.0002 Microsoft® Help Workshop [.\FBox Builder]
hcw.exe 4.03.0002 Microsoft® Help Workshop [.\FBox Builder]
HELPDECO.EXE [.\FBox Builder]
HMI_Edit.exe 2.00.0210 SAIA HMI Editor [.\HMI Editor]
htmlhelp.exe 4.74.8703 Microsoft® HTML Help Author [.\FBox Builder]
hwdll.dll 4.03.0002 Microsoft® Help Workshop [.\FBox Builder]
IconEditor.exe 2.00.0002 [.\HMI Editor]
Interop.SCommXX7.dll 1.0.0.0 [.\SWebConnect]
Interop.STracer.dll 1.0.0.0 OK
KEYLIB32.dll 4.109a KEYLIB32 [.\Web Editor 5_15_02]
LangMan.exe 2.0.239.0 SAIA FBE Language Manager [.\FBox Builder]
LibCgi.dll 2.0.20.0 [.\SWebConnect]
LibDK3964.dll 2.0.20.0 [.\SWebConnect]
LibDriverXUsb.dll 0.0.3.0 LibDriverXUsb [.\SWebConnect]
LonIPComp52.exe 2.0.220.200 OK LonIP_Compiler
LonIPConf52.exe 2.0.220.200 OK Saia LON Configuration Editor
MacroDlg.exe 1.5.12.99 SpiderControl Macro Dialog [.\Web Editor 5_15_02\MacroDlgComp]
MacroDlg.exe 1.5.12.99 SpiderControl Macro Dialog [.\Web Editor 5_15_02\MacroDlgRuntime]
MakeLinCheck.exe V2.0.220.0 OK Saia PG5 V2 Library Info File Checksum Generator
MakeSaiaZip.exe 2.0.220.0 OK Saia PG5 SaiaZip File Generator
MakeSrx.exe V2.0.220.0 OK Saia Source File Encrypter
MakeSrx.exe V2.0.220.0 OK Saia Source File Encrypter [.\FBox Builder]
MiniHttpd.dll 1.2.0.24584 MiniHttpd HTTP Server Library [.\SWebConnect]
NGMScan2.exe2.0.14 NG-MScan 2 [..\Program Files (x86)\Engiby\NGMScan2]
PLC_FontGenerator.exe 1.0.0.1 SpiderControl PLC Editor IniNet Font Generator [.\Web Editor 5_15_02]
Pop3Lib.dll 2.0.20.0 [.\SWebConnect]
RegKey52.exe V2.0.220.0 OK RegKey
RegKey52.exe V2.0.220.0 OK RegKey [.\Web Editor 5_15_02]
Saia.Net.exe 2.0.20.0 [.\SWebConnect]
SAIA_BlockIO.dll 2.0.239.0 SAIA FBE Block I/O Handler [.\FBox Builder]
SAIA_BlockObj.dll 2.0.239.0 SAIA FBE Block Object [.\FBox Builder]

SAIA_HelpGen.dll 2.0.239.0 SAIA FBE Help Compiler [.\FBox Builder]
SAIA_MsgSvr.exe 2.0.239.0 SAIA FBE Message Server [.\FBox Builder]
SaiaBurgess.SCS.MailInterface.dll 2.0.20.0 [.\SWebConnect]
SaiaConf52.dll V2.0.220.200 OK SaiaConf
SaiaFtpCli.dll 2.0.220.0 OK SaiaFtpCli
SaiaInst.exe 2.0.239.0 SAIA Library Package Installer [.\FBox Builder]
Saialib52.dll 2.0.220.0 OK SaiaLib52
SaiaNtFs.dll 2.0.215.0 SaiaNtFs [.\SD_Flash_Explorer]
Sasm52.dll V2.0.220.0 OK Saia PG5 V2 Build Utility
SB_PluginUtil.dll 2.0.20.0 [.\SWebConnect]
SBACnet_Compiler_PG5_2_0.exe V2.0.220.0 BACnet compiler [.\BACnet]
SBACnet_Config_PG5_2_0.exe 2.0.220.200 SBACnet_Config_PG5_2_0 [.\BACnet]
SBACnet_Datatype.dll 2.0.220.200 SBACnet_Datatype [.\BACnet]
SBlkDnld52.dll V2.0.220.0 OK New Block file downloader
SBMaster_PluginUtil.dll 2.0.20.0 [.\SWebConnect]
SBMaster_ProfiPlugin.dll 2.0.20.0 [.\SWebConnect\SBMaster Plugins]
SBMaster_SerialPlugin.dll 2.0.20.0 [.\SWebConnect\SBMaster Plugins]
SBMaster_UdpPlugin.dll 2.0.20.0 [.\SWebConnect\SBMaster Plugins]
SBMaster_USBPlugin.dll 2.0.20.0 [.\SWebConnect\SBMaster Plugins]
SBMasterDlg.dll 2.0.20.0 [.\SWebConnect]
SBMasterDll.dll 2.0.20.0 [.\SWebConnect]
SBMasterDrv.dll 2.0.20.0 [.\SWebConnect]
SBMasterUtil.dll 2.0.20.0 [.\SWebConnect]
SBSlave_PluginUtil.dll 2.0.20.0 [.\SWebConnect]
SBSlave_UdpPlugin.dll 2.0.20.0 [.\SWebConnect\SBSlave Plugins]
SBSlaveDll.dll 2.0.20.0 [.\SWebConnect]
SBSlaveDrv.dll 2.0.20.0 [.\SWebConnect]
SBSlaveUtil.dll 2.0.20.0 [.\SWebConnect]
SBuESFwDnld52.exe V2.0.220.0 OK Bues Firmware Download Utility
Sbug52.exe V2.0.220.0 OK Saia PG5 Online Debugger
Sbuild52.dll 2.0.220.1 OK Saia PG5 Build Utilities
SBUtil.dll 2.0.20.0 [.\SWebConnect]
SColl52.dll 2.0.220.200 OK Saia PG5 V2 Symbol Collector
SCommCom52.dll V2.0.220.0 OK SAIA Communication Driver
SCommDlg52.dll V2.0.220.0 OK SAIA Communication Driver
SCommDll52.dll V2.0.220.0 OK SAIA Communication Driver
SCommDLLClient.dll 2.0.20.0 [.\SWebConnect]
SCommDrv52.exe V2.0.220.0 OK SAIA Communication Driver
SCommNative.dll 2.0.20.0 [.\SWebConnect]

SCommUsb.dll	V2.0.220.0	OK	SAIA USB Driver
SCommUsbId.dll	V2.0.220.0	OK	SAIA USB Identifier
SCommUsr52.dll	V2.0.220.0	OK	SAIA Communication Driver
SCommUtil.dll	2.0.20.0		[.\SWebConnect]
SCommWrap.dll	2.0.220.0	OK	SCommWrap
SCommXX7.exe	1.0.2.13		Saia XX7 Communication driver [.\SWebConnect]
Sconf52.exe	V2.0.220.200	OK	Saia PG5 Online Configurator
Sdasm52.exe	as Sdisasm52.dll		Saia PG5 Disassembler
Sdat52.exe	V2.0.220.0	OK	Saia PG5 V2 Data Transfer Utility
SDBIf52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor DB Interface
SDevConf52.exe	2.0.220.200	OK	SDevConf52
SdFs.exe	2.0.215.0		Saia SD file system explorer [.\SD_Flash_Explorer]
Sdisasm52.dll	V2.0.220.0	OK	Saia PG5 V2 Disassembler
secho.exe		OK	
sedata52.dll	V2.0.220.200	OK	Saia PG5 V2 Symbol Editor Edit Data
Sedit52.exe	V2.0.220.0	OK	Saia PG5 V2 IL Editor
Sfequ52.exe	2.0.220.200	OK	Saia PG5 V2 Symbol Converter
SFtp52.exe	2.0.220.0	OK	SFtp52
Sfup52.exe	V2.0.220.200	OK	Saia Fupla Editor
Sfupgen52.exe	V2.0.220.200	OK	Fupla Code Converter-Generator
SFWDnld52.exe	V2.0.220.0	OK	Firmware download utility
SFwDnldAssist52.exe	V2.0.220.0	OK	Firmware download assistant
Sgraf52.exe	V2.0.220.0	OK	Saia Graftec Editor
SGSF52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor Global Symbol updater
Simag52.dll	V2.0.220.0	OK	Saia PG5 V2 PCD Memory Image Generator
SImp52.dll	V2.0.220.0	OK	Saia Template Page Importer
simp52u.dll	V2.0.220.0	OK	Saia Template Page Importer
SInstall52.exe	2.0.220.0	OK	Saia PG5 Installer
SKeyGen52.exe	V2.0.220.0	OK	Saia PG5 V2 Key Generator
SLibConv52.exe	2.0.220.0	OK	Library Converter, by Peter Nagy
Sload52.dll	V2.0.220.200	OK	Saia PG5 V2 Up/Downloader Library
Sload52.exe	V2.0.220.200	OK	Saia PG5 V2 Command Line Up/Downloader
Smake52.exe	as Sasm52	OK	Saia PG5 Command Line Build Utility
SMsgWnd52.dll	2.0.220.0	OK	Saia PG5 V2 Message Window, by Gábor Domonyi
Snet52.exe	V2.0.220.0	OK	Saia Network Configurator
SnetSpm.dll	1.0.2.0	OK	Spm Messaging interface

SNgen52.exe	2.0.220.0	OK	SNgen52
SPCDImp52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor PCD Importer
Spglib52.dll	V2.0.220.1	OK	Spglib52.dll
Spglib52.dll	V2.0.220.1	OK	Spglib52.dll [.\Web Editor 5_15_02]
Spm52.exe	V2.0.220.200	OK	Saia PG5 V2 Project Manager
Spm52Startup.exe	V2.0.220.200	OK	SPM Startup Application
SpromSvr52.dll	V2.0.220.0	OK	Hex File Generator
SPropEditors52.dll	2.0.220.0	OK	SPropEditors52
sqlcnf52.exe	V2.0.220.0	OK	SAIA BuES1++ SQL Configurator
SRemoteServer52.dll	2.0.220.200	OK	SRemoteServer52
SRename52.dll	V2.0.220.0	OK	Saia PG5 V2 Global Symbol Renamer
Sres52.dll	V2.0.220.0	OK	Saia PG5 V2 Resource List Generator
SRio52.exe	2.0.220.200	OK	S-RIO Network Configurator
SRIOBuilder.exe	2.0.220.200	OK	SDevConf52
SRM52.exe	2.0.220.200	OK	Saia PG5 V2 Symbol File Editor
SRMComm52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor Communication Object
SRMDB52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor Database
SRMSvr52.exe	2.0.220.200	OK	Saia PG5 V2 Symbol Editor Server
SRMTE52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor
SRXPImp52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor RXP Importer
SSY5Imp52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor SY5 Importer
STaskbar52.dll	V2.0.220.0	OK	Windows7 Shell Extension Library
stdole.dll	7.00.9466	OK	
StracerProxy.dll	1.0.0.0	[.\SD_Flash_Explorer]	
STXTImp52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor TXT Importer
Sview52.exe	V2.0.220.0	OK	Saia PG5 V2 Text File Viewer
SVISImp52.dll	2.0.220.200	OK	Symbol Editor Vision Importer
Sww52.exe	V2.0.220.0	OK	Saia PG5 Watch Window
SXLSImp52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor XLS Importer
SXtpLib.dll	V2.0.220.0	OK	SXtpLib Dynamic Link Library
SymDisp52.exe	as Sasm52		Saia PG5 PCD/OBJ File Viewer
SymDlg52.dll	V2.0.220.0	OK	Saia PG5 V2 Global Symbol Browser
SymDlg52.dll	V2.0.220.0	OK	Saia PG5 V2 Global Symbol Browser [.\Web Editor 5_15_02]
SymFrm52.dll	2.0.220.200	OK	Saia PG5 V2 Symbol Editor Frame
SymXX7Dlg.dll	1.0.0.12		SAIA XX7 Symbol Explorer [.\Web Editor 5_15_02]
Syncfusion.Core.dll	6.302.0.6	OK	Syncfusion.Core
Syncfusion.Diagram.Windows.dll	6.302.0.6	OK	Syncfusion.Diagram.Windows

Syncfusion.Edit.Windows.dll	6.302.0.6	OK	Syncfusion.Edit.Windows
Syncfusion.Grid.Base.dll	6.302.0.6	OK	Syncfusion.Grid.Base
Syncfusion.Grid.Grouping.Base.dll	6.302.0.6	OK	Syncfusion.Grid.Grouping.Base
Syncfusion.Grid.Grouping.Windows.dll	6.302.0.6	OK	Syncfusion.Grid.Grouping.Windows
Syncfusion.Grid.Windows.dll	6.302.0.6	OK	Syncfusion.Grid.Windows
Syncfusion.Grouping.Base.dll	6.302.0.6	OK	Syncfusion.Grouping.Base
Syncfusion.Grouping.Windows.dll	6.302.0.6	OK	Syncfusion.Grouping.Windows
Syncfusion.HTMLUI.Base.dll	6.302.0.6	OK	Syncfusion.HTMLUI.Base
Syncfusion.HTMLUI.Windows.dll	6.302.0.6	OK	Syncfusion.HTMLUI.Windows
syncfusion.scripting.base.dll	6.302.0.6	OK	Syncfusion.Scripting.Base
Syncfusion.Shared.Base.dll	6.302.0.6	OK	Syncfusion.Shared.Base
Syncfusion.Shared.Windows.dll	6.302.0.6	OK	Syncfusion.Shared.Windows
Syncfusion.Tools.Base.dll	6.302.0.6	OK	Syncfusion.Tools.Base
Syncfusion.Tools.Windows.dll	6.302.0.6	OK	Syncfusion.Tools.Windows
Tapi2Lib.dll	1.0.2.5		Tapi2Lib [.\SWebConnect]
ToolkitPro1031vc80.dll	10.3.1.0	OK	Xtreme Toolkit ProT Library
ToolkitPro1031vc80U.dll	10.3.1.0	OK	Xtreme Toolkit ProT Library
Tracewin.exe	1.6.3.0	OK	Copyright ©Saia-Burgess 1999-2010
TrayAdminFF.dll	2.0.20.0		[.\SWebConnect]
UDPTrace.exe	1.0.2014.26584		[.\SWebConnect]
UDPTraceCapture.exe	2.0.001	OK	UDP Trace Capture
ugrid.dll	1.0.0.1	OK	ugrid DLL
WdfCoInstaller01009.dll	1.9.7600.16385		(WDF Coinstaller [.\Driver USB1\amd64]
Web-Builder-C.exe	2.00.0200	OK	
Web-Editor.exe	5.15.02.5150221		Saia® S-Web Editor MFC Application [.\Web Editor 5_15_02]
WebConnect.dll	2.0.20.0		[.\SWebConnect]
WebHMIServer.dll	2.0.19.6		[.\SWebConnect]
WinCEHelper.dll	2.0.20.0		[.\SWebConnect\SBMaster Plugins]
winusbcoinstaller2.dll	6.1.7600.16385		(Windows Driver Foundation - User-mode Platform Device Update Co-Installer [.\Driver USB1\amd64]
XX7SComm.dll	2.0.20.0		[.\SWebConnect]
ZipDll.dll	1.73.3.0		ZipDLL [.\FBox Builder]

Appendix L Versions ProMoS NT

Installed versions:

_SetupVersion	1.6.1.9	01.03.13 12:45:25
AlmMng.exe	1.6.8.23	[29.11.12 06:33:20]
AlmView.exe	1.6.8.36	[25.02.13 16:25:22]
AsciiExport.exe	1.6.7.0	[28.06.12 23:14:10]
AudioAlmTelTerminal.exe	1.5.0.22	[11.08.10 17:13:08]
bmp2jpg.exe	1.6.7.0	[28.06.12 23:14:54]
changepwd.exe	1.6.7.1	[28.06.12 23:16:30]
ClkCfg.exe	1.6.7.3	[28.06.12 23:14:12]
clkmng.exe	1.6.8.3	[27.11.12 22:42:10]
dms.exe	1.6.8.102	[15.02.13 12:16:46]
ESPADRIVER.exe	1.6.7.1	[28.06.12 23:14:22]
ge.exe	1.6.8.127	[29.08.12 11:04:24]
ge_1.exe	1.6.7.124	[28.06.12 23:14:36]
HDAMng.exe	1.6.7.12	[28.06.12 23:14:40]
Logger.exe	1.6.7.3	[28.06.12 23:14:54]
MalmCfg.exe	1.6.7.17	[28.0.12 23:14:58]
MalmMng.exe	1.6.8.36	[19.12.12 14:56:18]
oList.exe	1.6.7.2	[28.06.12 23:16:50]
OPCDriver.exe	1.6.8.6	[27.02.13 11:16:00]
pBackup.exe	1.6.8.1	[31.10.12 14:30:50]
pCalc.exe	1.6.7.9	[12.12.12 13:43:18]
pChart.exe	1.6.8.28	[27.11.12 21:50:32]
PDBS.exe	1.6.8.24	[27.11.12 22:58:30]
pdbsdump.exe	1.5.7.0	[28.06.12 23:17:06]
pEdit.exe	1.6.7.0	[28.06.12 23:17:00]
PET.exe	1.6.8.79	[28.02.13 11:43:52]
PET_1.exe	1.6.7.72	[29.06.12 14:47:50]
PFTP.exe	1.6.8.3	[19.11.12 09:49:32]
pList.exe	1.6.7.2	[28.06.12 23:17:02]
plogin.exe	1.6.8.9	[15.10.12 16:05:28]
PMosFilePicker.exe	1.6.8.1	[23.08.12 17:16:06]
Ppaint.exe	1.6.7.0	[28.06.12 23:15:36]
pRestore.exe	1.6.1.1	[27.02.13 10:05:54]
ProjectCfg.exe	1.6.8.17	[27.02.13 10:34:26]
promos.exe	1.6.7.8	[28.06.12 23:17:40]
PrtDumpWin.exe	1.6.7.0	[28.06.12 23:15:44]
prtformat.exe	1.6.8.5	[20.11.12 12:29:28]
prtmng.exe	1.6.7.3	[28.06.12 23:15:48]

PrtView.exe	1.6.7.2	[28.06.12 23:15:52]
pSMS.exe	1.6.7.8	[28.06.12 23:17:04]
pStop.exe	1.6.8.3	[15.10.12 15:55:56]
puser.exe	1.6.8.8	[15.10.12 20:48:04]
PWEB.exe	1.6.8.52	[29.11.12 10:24:40]
SDriver.exe	1.6.9.39	[04.03.13 09:24:54]
SetDMSVal.exe	1.6.7.2	[28.06.12 23:16:10]
ShowVersion.exe	1.6.7.0	[28.06.12 23:16:12]
SNMPDriver.exe	1.6.7.7	[28.06.12 23:16:10]
TAPIDRIVER.exe	1.6.7.2	[28.06.12 23:16:16]
Cityruf.dll	1.0.0.0	[21.09.04 08:48:20]
csh.dll	1.0.0.30	[24.08.99 15:17:48]
hswpage.dll	1.1.2.0	[29.12.99 23:27:26]
hswsms.dll	1.1.3.0	[07.05.00 22:10:06]
HSWTAP.dll	0.0.0.1	[25.04.03 09:42:40]
pdbs.dll	1.6.8.10	[03.09.12 15:15:44]
pmosfunc.dll	1.6.8.49	[28.02.13 09:46:10]
PMOSPIPE.dll	1.6.7.14	[28.06.12 23:17:36]
pmoszip.dll	1.0.0.0	[09.12.02 22:35:00]

Appendix M Publications

Gasser T., Dominic Palmer-Brown, David Xiao (2009), Paradigm shift in PLC programming, AC&T 2009: *Precedings of the AC&T 2009*, pp 43 - 48