BASE PROGRAM LIBRARY FOR PLC APPLICATION DESIGN

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Valkeakoski, Degree Programme in Automation Engineering

Fall 2018

Samuli Multaniemi

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Degree Programme in Automation Engineering
Valkeakoski

Author Samuli Multaniemi **Year** 2018

Title Peruspiirikirjasto PLC-ohjelmoinnissa

Supervisor(s) Hannu Pohjasto

TIIVISTELMÄ

Tämä opinnäytetyö tehtiin Tampereen Automaatiosähkö Oy:n suunnitteluosastolle. Opinnäytetyön tärkein tavoite oli tutkia syvällisesti, suunnitella ja ohjelmoida peruspiirikirjasto teollisuusautomaatiossa käytettäville ohjelmoitaville logiikkaohjaimille. Peruspiirikirjaston tärkeimmät käyttöalueet ovat laajoissa teollisuuslaitoksissa, joissa prosessialueet, laitteistot ja yleinen laitosrakenne sisältävät suuria määriä samankaltaisia piirteitä. Tällöin voidaan hyödyntää tiettyihin pohjiin perustuvia parametroituja toimilaiteohjelmia.

Peruspiirikirjaston luomistyössä syvennyttiin erityisesti Siemensin ohjelmoitaviin logiikkaohjaimiin (PLC), niiden ohjelmointiin, Siemensin suunnittelu-, ja käyttöohjelmistoihin, ja yleisesti teollisuusautomaation tärkeimpiin pääpiirteisiin. Tutkimustyön ja yrityksen työn ohjaajan kokemusten perusteella luotiin määritelmät peruspiirille työn aiheeksi. Opinnäytetyössä päätettiin keskittyä analogiseen mittaukseen, sillä kaikki modernit automaatiojärjestelmät hyödyntävät suuria määriä anturitietoja prosessilta.

Opinnäytetyössä luotiin täysin toimiva analogisen mittauksen toimilaiteohjelma, johon on sisällytetty yleisimpiä toiminnallisuuksia, joita analogisen mittauksen kanssa käytetään. Tämän lisäksi työssä tehtiin kattava tutkimus edellä mainituista aiheista, jota voidaan hyödyntää peruspiirikirjaston jatkamisessa ja yleisesti automaatiotekniikan sovellussuunnittelussa. Työssä rakennettu peruspiiri tullaan ottamaan käyttöön yrityksessä sopivissa projekteissa ja opinnäytetyön tekijä jatkaa niiden parissa työskentelyä.

Avainsanat Ohjelmistosuunnittelu, Ohjelmoitavat Logiikkaohjaimet, Teollisuusautomaatio

Sivut 30 sivua, joista liitteitä 0 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Automation Engineering
Valkeakoski

Author Samuli Multaniemi **Year** 2018

Subject Base program library for PLC application design

Supervisor(s) Hannu Pohjasto

ABSTRACT

This thesis project was commissioned by Tampereen Automaatiosähkö Oy for their engineering and application design department. The main goal for the thesis project was to research, plan and develop a base program block library for industrial PLC systems. Its main purposes are in systems, where devices, process areas and the overall structure has multiple similar elements, and a generative template-based program design can be utilized.

Creation of such a library involved a deep research into the capabilities of PLC products from Siemens, programming features and languages, their software package, and industrial automation around them was conducted. Based on them, and personal experience from the customer company supervisor of the author, predefinitions were formed for a program block to be designed and implemented. The implementation phase of the thesis focused on the Analogue Measurement function block as large amounts of analogue measurements are the bedrock of any functional modern automation system.

The final outcome of the thesis was a fully working, feature-rich function block for Analogue Measurement, and extensive documentation to continue developing further program functions on the principles researched, designed and implemented. The base programs developed will be taken into use in upcoming applicable projects of the customer company and the author will continue working with them in the future.

Keywords Industrial Automation, Programmable Logic Controllers, Programming, Design

Pages 30 pages including appendices 0 pages

CONTENTS

No Appendices

LIST OF ABBREVIATIONS

PLC	Programmable Logic Controller
CPU	Central Processing Unit
HMI	Human Machine Interface
GUI	Graphical User Interface
DCS	Distributed Control System
SCADA	Supervisory Control And Data Acquisition
I/O	Input/Output
DI	Digital Input
DO	Digital Output
AI	Analogue Input
AO	Analogue Output
UDT	User Defined Type
OB	Organisation Block
FB	Function Block
FC	Function
DB	Data Block
CFC	Continuous Function Chart
SCL	Structured Control Language
ST	Structured Text
FBD	Function Block Diagram
LAD	Ladder
SF	System Fault
TIA	Totally Integrated Automation
TAS	Tampereen Automaatiosähkö Oy

1 INTRODUCTION

The topic for this thesis project was commissioned directly by the customer company Tampereen Automaatiosähkö Oy, from here on referred as TAS. The company has been operating in all automation and electrical fields since 1993, mostly in industrial automation customer applications. They have departments of installers, cabinet assemblers and designers and a broad customer base all over the world.

The biggest and most predominant business line for TAS has been its electrical and automation installation services and projects. Nowadays, the company has diversified their business lines and has a well-performing design and engineering department. Supporting the two aforementioned business lines, TAS also assembles electrical cabinets of all sizes.

The commissioning of this thesis work stems from the need to be more competitive in the industrial automation market. For electrical and automation installation projects the situation is rather simple, as long as the installers have good circuit diagrams and tools, they can carry out the work very efficiently. For design projects especially in industrial automation, the case is very different. Traditional design methods, tools and procedures, where all automation system logic was built with separate instructions are becoming too slow for commission projects. Program generation and block template -based program building is a great deal faster especially in larger projects.

TAS has had issues with inefficiencies before as no premade program blocks have existed with broad enough features. This has resulted in previous projects starting from only partial premade sources and not utilizing optimal design tools. Thorough research into the possibilities of modern design tools and what can be automated, was needed.

The program blocks built in the thesis work will be in use of all of the company's application designers and project engineers and in theory, will improve the ability to compete in the bidding of large projects.

This thesis project deals directly with features and operation of modern automation system components found especially in industrial automation applications. In addition, the project topic is not connected to any specific real-life customer project, which means that the work done is quite theoretical and based on ideal conditions. Design choices were made based on previous preferences and wishes for the new library by the customer company. For these reasons, before going more in depth about the project, main principles of automation systems and industrial automation are explained.

2 MAIN PRINCIPLES AND THEORY OF INDUSTRIAL AUTOMATION

2.1 Programmable Logic Controllers

All of the tasks of a modern automation system are executed through some form of microprocessor-based Central Processing Unit (CPU), which is usually called by its general term, Programmable Logic Controller (PLC). PLC's largely replaced cumbersome relay and timer logic in the early 1970's and nowadays can handle extremely complex instructions. They interact with the process through Input/Output (I/O) modules, evaluating signals and executing control actions. System integration by manufacturers allows for flexible design methods to be used and innovation is the limit of automation systems these days. (Gonzales, 2015, MachineDesign)

PLC's operate in what are called scanning cycles, which are formed of three major steps. Firstly, the controller reads states of its various physical inputs and passes them onto the program, then all of the instructions of the PLC's program are executed. Options for these instructions are virtually endless, such as assigning values into memory, all the instruction logic and loading and setting values. After the program instructions are executed, output changes are executed. This means that if for example on this cycle, the program set Digital Output (DO) Q1.5 channel bit as TRUE, the controller will now change the state of that specific channel to TRUE or ON, which means it will give out its ON – state, for example a +24V voltage output in most cases. (Gonzales, 2015, MachineDesign)

The controller operates extremely fast and the cycle time is in the range of milliseconds, but a program too robust can and will slow it down to hundreds of milliseconds if not full seconds in worst cases. As one can image, this will cause severe process control difficulties and can even lead to the process not working at all as intended.

2.2 PLC Program Structure

There are a few different forms of program related data inside the memory of the PLC. In general, PLC program structures follow the standard IEC 61131-3 and languages used in this project are described more in depth in chapter 4.3 "Design and Testing".

Firstly, there is the logic program is what houses the aforementioned logic instructions. The terms used for the different data groups in the PLC memory differ by manufacturer. In the Siemens environment the program consists of Organisation Blocks (OB), which are the highest-level program units. They consist of Functions (FC) and Function Blocks (FB).

FC's and FB's are just compiled program units, which operate in the same way as in all programming; they perform a function or a feature based on

their inputs and publish the result in their outputs. The difference between a function and a function block is that every function block has its own dedicated memory area, called a Data Block (DB), more specifically an "Instance" Data Block which means that it houses all of that specific function block's data points, variables and I/O. It can be accessed by any unit of the program, but its main purpose is data storage and communication for its own function block. Data Blocks can be used for other situations as well, these are called "Shared" or "Global" DB's. These DB's can be used to store any type of data and can be accessed from any program unit or even the control room HMI – software if linked properly and not protected against reading or writing.

2.3 **Industrial Automation**

In industrial automation, where processes and systems are broad and large, any means of optimizing design work are extremely helpful. The project plant or process almost always consists of multiple identical devices, measurements, signals and connections. This allows for a generation-based system design, where a template of a sort can be used for a number of these process positions, instead of building all of them from individual logic instructions. These templates can be called "base program blocks" and they will be the main target and scope of this thesis work.

Another main advantage for building and using such premade general program blocks is that they and their functions have already been tested to be functional for their purpose and so in the design phase of a new project, the designer can be certain that they do not need to spend time making the main logic of a block work. All they need to do is parametrize and connect it to the program according to the instructions.

3 **GOALS OF THE PROJECT**

The scope of a full base program block library is quite vast, and the pre-design work was deemed more important in this commissioning. In practise this meant researching in general how the base program blocks can be combined into the case-specific logic preceding and succeeding them. For example, how parameters, explanatory text strings and signals can be automatically inserted and relayed onto the control room graphical user interface (GUI) of the human machine interface (HMI).

The ideal block library consists of all main components of a typical automation system: Digital and analogue measuring, general control, valve properties and control, motor control either directly online and through a controller, for example a variable speed drive. In addition, a generally

functional modular sequence function block is very commonly found in process control. As the scope of this thesis work needed to be kept reasonable, designing the full library was decided to be rather unreasonable. Instead a very integral block was chosen to focus on, analogue measurement. This block represents the most commonly found and important part of any automation system and should demonstrate design choices and principles the best.

An integral part of the thesis work was to design the logic for these blocks in an efficient way and prioritize features. Having the blocks too robust would affect the cycle time of the controller which is extremely bad as the primary function is to keep them usable in all situations. Also increasing inputs and outputs of a single block would hinder its usability in the design phase. The latter, though is not a primary concern as tools for hiding rarely used I/O points do exist.

Nowadays automation systems have become much more complex and directly in charge of many more things than in the past. This has led to projects having many different parts and trimming down unneeded work is of paramount importance. Having a comprehensive library of base program blocks means that the time spent adding new features according to the customers specifications will be short. For a company specialised in the field being able to offer versatile services and design work quickly is becoming more important every day.

4 PRE-DESIGN PLANNING AND TOOLS

4.1 Physical Equipment

TAS has chosen to use Siemens logic controller-based systems in their projects and thus the thesis work will also be done in the Siemens ecosystem. Siemens is a leading automation solution manufacturer and is one of the most globally well-known automation brands. As TAS has a long history using Siemens in their projects, they were able to offer me their professional insight and equipment to carry out this thesis.

All design work and testing was done with their software and equipment in their headquarters in Tampere. Various Siemens PLC's and I/O – modules were used from both older and newer product families to evaluate their differences and document new features.

Siemens calls their PLC devices and equipment under the name "SIMATIC" derived from the words "Siemens Automatic". The product line currently sold is called S7 and it is the successor to their older S5-line of PLC-devices. The S7 – line is one of the most widely used PLC – equipment families in the world, which makes it greatly suitable for the thesis work and for future

projects. In this thesis, mainly the S7-300 series and lighter ET200S and ET200SP – product line PLC's and modules will be used. S7-300 is currently being phased out and replaced with the new S7-1500 – line. (Siemens AG)

4.2 **Software Environment**

Due to computer software and operating system changes recently, the Siemens automation software package has become quite fragmented. The original STEP 7 – SIMATIC software from 1995 is still largely used but is being updated quite slowly as the whole program architecture is over 20 years old. The main view of the software and a project, SIMATIC Manager is shown in Figure 1 below.

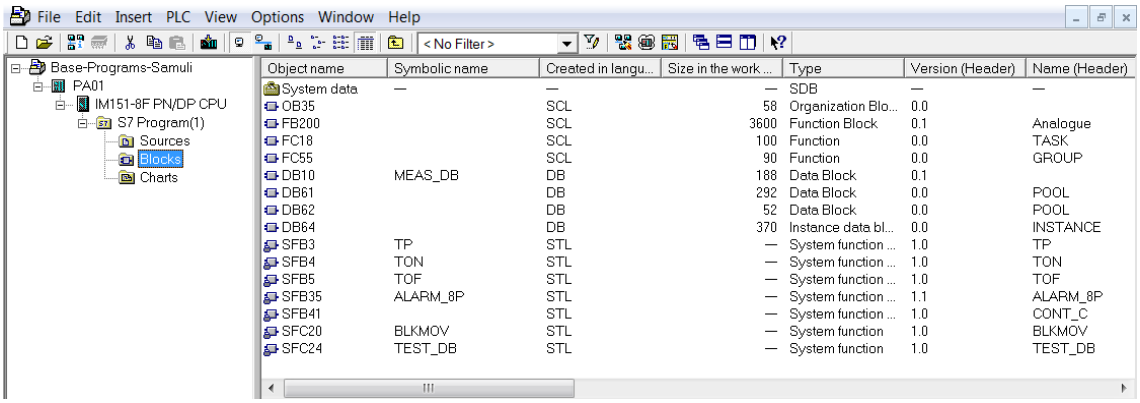

Figure 1. Main view of a project in STEP7 SIMATIC Manager (Samuli Multaniemi, 2018)

A line up aiming for the future called Totally Integrated Automation (TIA) - Portal has been their main focus since 2010 and is the only tool for programming their new product lines, such as the S7-1200 and -1500 – controllers. TIA - Portal does not support all of the functionalities yet so using both in unison is necessary at least for this thesis. In addition, TIA – Portal has been publicly criticized for requiring much more computational power and resources in general from the Windows-machine running it.

This causes some trouble as many customers are against updates for their functional and operational systems, especially in the industrial setting, where any system shutdown results in financial losses. Quite a sizeable portion of industrial plants are still running automation systems from decades ago and so the only way to program, develop and amend these systems is through the old software. Eventually the update will be inevitable though, so the function block library will be needed to be compiled also in the TIA – Portal. The scope of the thesis work project was already large and very broad, so the migration of the base program blocks to TIA was excluded.

The main difference between TIA – Portal and the STEP7 – software family is that Siemens aims to integrate all automation system equipment into a single portal-form software, which is where the name also originates (Totally Integrated Automation). Because of this design plan, TIA – Portal is designed around a one-window GUI principle. This means that opposed to the old STEP7 – software, all programming, configuring, simulating etc. environments are placed in the same program window. The window is divided into modular sections and after getting familiar with it, is very intuitive and convenient to use. The main view of a project in TIA – Portal can be seen in the Figure 2 below.

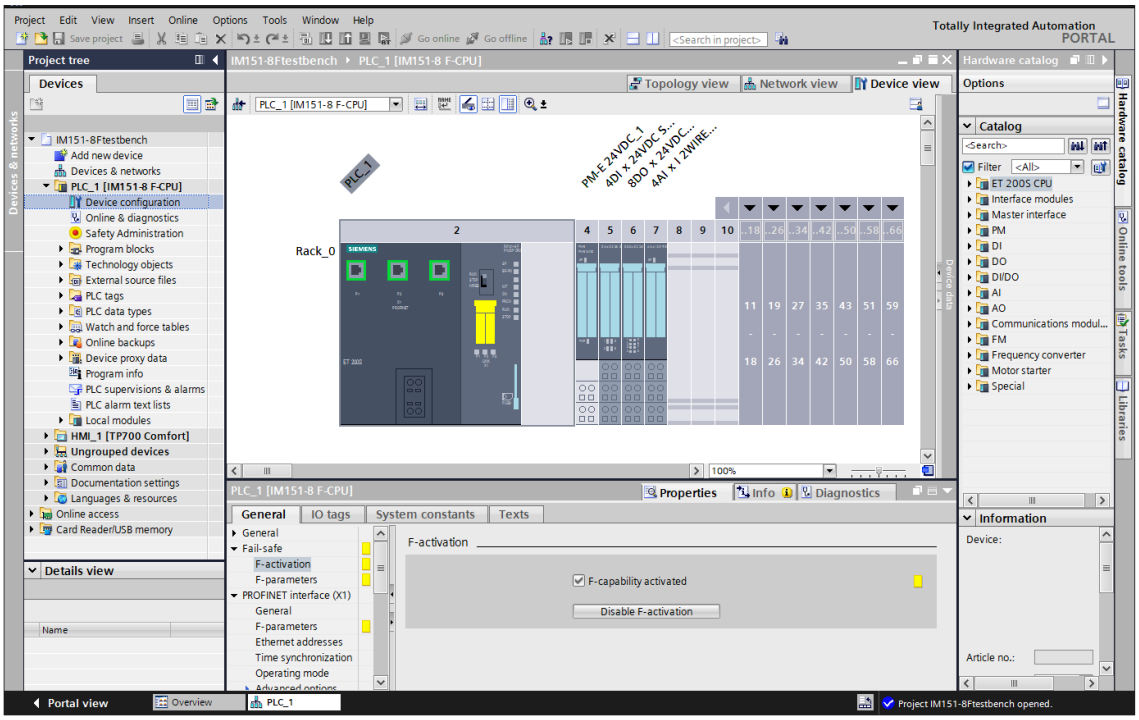

Figure 2. Main view of a project in TIA – Portal (Samuli Multaniemi, 2018)

TIA – Portal is still in its aggressive development phase at the point of writing this thesis, which introduces some unfortunate situations. In addition to some software bugs and compatibility issues, some programming tools, commands, instructions and features have yet to be fully implemented. In addition to this, some features which were implemented in earlier versions of TIA – portal, have now been removed for reasons unknown.

An example of this is indirect addressing of data block contents. This program feature was functional in STEP 7 but has now been replaced with “PEEK” and “POKE” – instructions, which don’t allow all the flexibility of the previous “WORD_TO_BLOCK_DB” – instruction. “PEEK” and “POKE” – instructions work well in their intended use case; transferring large continuous memory structures from one memory area to another. “WORD_TO_BLOCK_DB” is better suited for smaller, uniform sized data movements.

The reception from users and customers towards TIA – Portal has been positive overall and the implemented functions in the current versions of the software work quite well. Once Siemens finishes adding program features, TIA – Portal will become what its name suggests – a Totally Integrated Automation Portal.

4.3 **Design and Testing**

The block design was done with the IEC 61131-3 – defined programming language known universally as ST (Structured Text) and in the Siemens system as SCL (Structured Control Language). SCL is a versatile high-level PLC programming language with a simple syntax. It resembles Pascal and is best suited for building function blocks to be used in a full program. For example, unit conversions, function block calls, loops and clauses and arithmetic operations are simple to implement. The SCL editor is shown in the Figure 3 below.

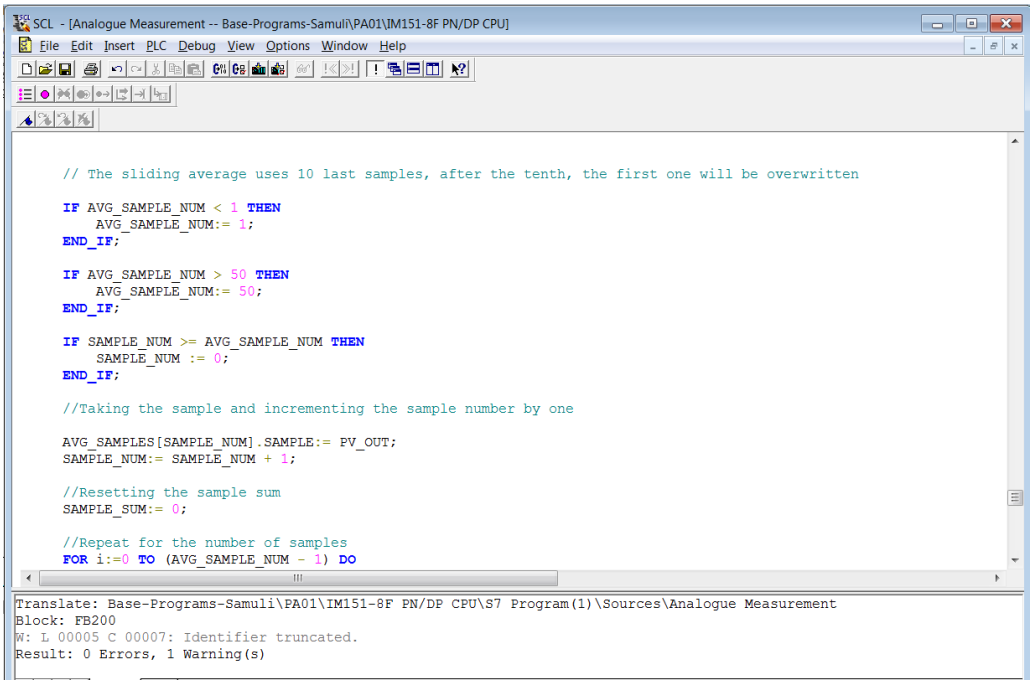

Figure 3. The SCL Programming Editor in STEP7 (Samuli Multaniemi, 2018)

Testing the designed programs was done in STEP 7 with a Siemens proprietary programming language, Continuous Function Chart (CFC), which uses function blocks similarly as the IEC 61131-3 – defined Function Block Diagram (FBD). CFC was developed primarily for Siemens' PCS7, their broader automation system for process control applications. As the base programs blocks serve their best purpose in large systems, which often are programmed with CFC, it is wise to test them straight with CFC.

TIA Portal has no support for CFC yet which was also a reason why the design focus was decided to place on STEP7 still. As mentioned before, the block library will be needed in the future in TIA – Portal as well. Hopefully at that point, Siemens has introduced PCS7 – features such as CFC into TIA – Portal.

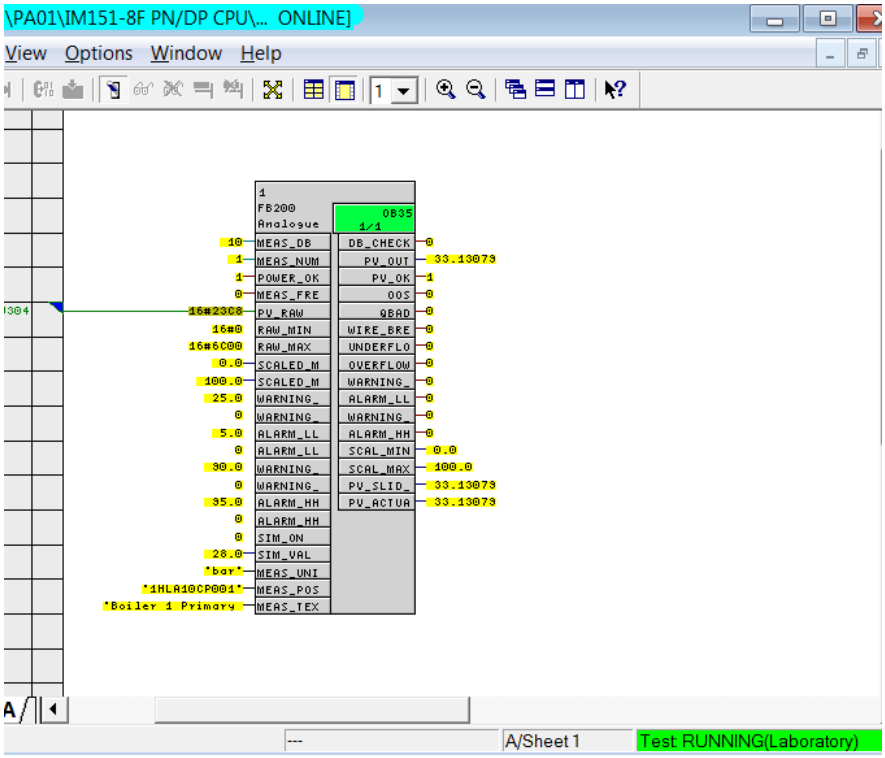

Figure 4. Test Mode in the Siemens CFC Programming environment in STEP7 (Samuli Multaniemi, 2018)

The live monitoring utility in CFC called "Test Mode" is shown in the Figure 4 above. It allows for direct input into the program and immediate live monitoring of all program activity. The user can choose which data points and variable are monitored if some of the are not important for the moment. In Test Mode, the active live connection is signified by the light blue "ONLINE" in the toolbar and the live RUN/STOP state in the bottom corner.

4.4 **Definitions for the program blocks**

Before the actual design work could be started, the program blocks needed definitions on features, inputs and outputs (I/O) and size. These definitions were ideal and their feasibility to implement in full was needed to be checked block by block after design and testing. The definitions consisted of what features, I/O, and data block connections were to be implemented in each block. For example, an analogue measurement block needs to pass on the measured values either in raw form or scaled and transformed. In addition to this, for example high and low alerts, their delays and hystereses are set in the block or read from a setup data block.

5 DESIGN OPTIMIZATION AND EXTERNAL TOOLS

5.1 **Advantages**

There are many angles from which the PLC program design can be optimized and they all link together through the base program blocks. Preferably the designer would only need to build the block related interlockings and other logic specific to the application and import prebuilt sources for the base program blocks and the data blocks related to them. Being able to modify function and data blocks externally for example in Microsoft Excel or any other data editor and then generate them into the project is a great advantage. Making modifications in bulk is much faster in Excel for example.

Data and parameter input is one of the most time consuming and taxing steps of building a process control logic program. Using a pre-designed program block can ease up this step by allowing the designer to input everything into the single program block. From the block, this data is then relayed to wherever it is needed. This also helps tremendously if for example some source measurement or an alarm limit needs to be changed afterwards; changing it from the input of the program block will change it everywhere.

Siemens uses this type of relaying in their PCS 7 – environment and the connection of HMI tags and objects straight from the program blocks is also done automatically if so desired. This means that the tools for enabling this on a smaller scale PLC – system are available. In Siemens' design software, they have implemented "system attributes" for function blocks and even their individual inputs, outputs and variables in general.

5.2 Traditional Design

PLC system design and hardware access is quite restricted for safety reasons and is conducted through the software provided by equipment manufacturers. As mentioned earlier, the equipment and environment chosen is Siemens and the Siemens software, STEP 7 and TIA – Portal are used for this thesis. The software is run on the Windows – platform of mainstream PC-computers.

The first form of PLC program design mainly revolved around the language known as “LADDER” (LAD), which resembles electrical circuits, and is formed of branching lines, logic gates and coils. These are the elements of a traditional relay control circuit and LAD was developed so that all electricians who had worked with relay control before could understand and troubleshoot PLC programs too.

LAD is still used widely for its simple annotation and nowadays prebuilt FC’s and FB’s can also be inserted into LAD, forming a middle ground between LAD and FBD. That said, implementing high level program features in traditional LAD program can sometimes be extremely cumbersome and in some cases, it is much easier to write the program in a FC or a FB in SCL or STL and use that in CFC or FBD.

The cornerstone of PLC programming has always been utilizing premade resources, usually in the form of Functions (FC) or Function Blocks (FB) provided originally from Siemens. They share the same basic principle as base program blocks of being prebuilt pieces to insert in a new program to simply gain functionality. Traditionally the FC’s and FB’s provided only offer functionalities fitting in very detailed and specific situations or only broad functions, such as an On-Delay Timer with no specifications other than input signal and the delay time itself.

Base program blocks take this to a further level, aiming to eliminate more of the specific building process in a new project. The reason why all this is relevant is that most if not all automation systems are formed of inherently identical modules.

5.3 External Tools and Source Utilization

All design is possible to be done through the aforementioned software, but for repetitive building of Data Blocks for example, integration tools for external software have been developed. For the STEP 7 – environment, an official add-in made by Siemens called “SIMATIC” for Microsoft Excel exists. Unfortunately, this add-in was only developed for Microsoft Excel 2003 and seems to have since been discontinued.

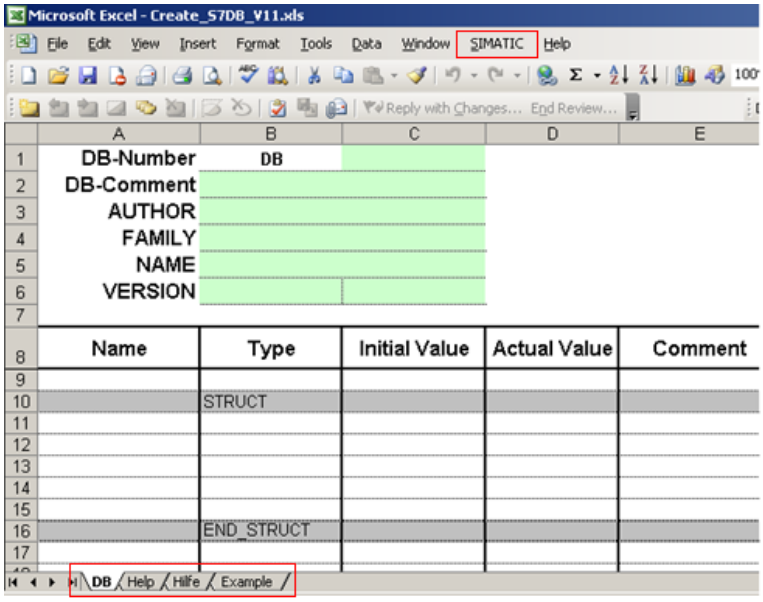

Figure 5. Microsoft Excel 2003 with "SIMATIC" add-in for Data Block creation (Siemens AG, 2013)

The add-in installed automatically into Excel when installing STEP7 on the same computer. It allowed for direct generation of source files from Excel spreadsheets and these sources could be imported directly into STEP 7 and after imported, can used in the project directory like a normal block. It is unclear why Siemens has discontinued this tool as generating Data Blocks through it was the quickest method ever to exist. A display of the add-in can be seen in Figure 5 above.

Importing into STEP7 from new Excel versions is still possible, albeit more laborious. The template base is not generated on its own, instead it needs to be exported as a source-file from STEP7, which can then be opened in Excel. After the contents have been written, the source file can now be saved as an ".awl" file. Importing this into STEP7 is possible, STEP7 recognizes this as a source file and creates the block.

External source importing and exporting is also present in TIA – Portal, but the functionality is slightly different. Due to the lack of an Excel – add-in, making table form spreadsheets and importing them straight into TIA – Portal doesn't work as well as it did with STEP 7. The feature is more useful if a similar project is available and blocks can be exported from it and imported into the new one. This is demonstrated in the figure 6 below.

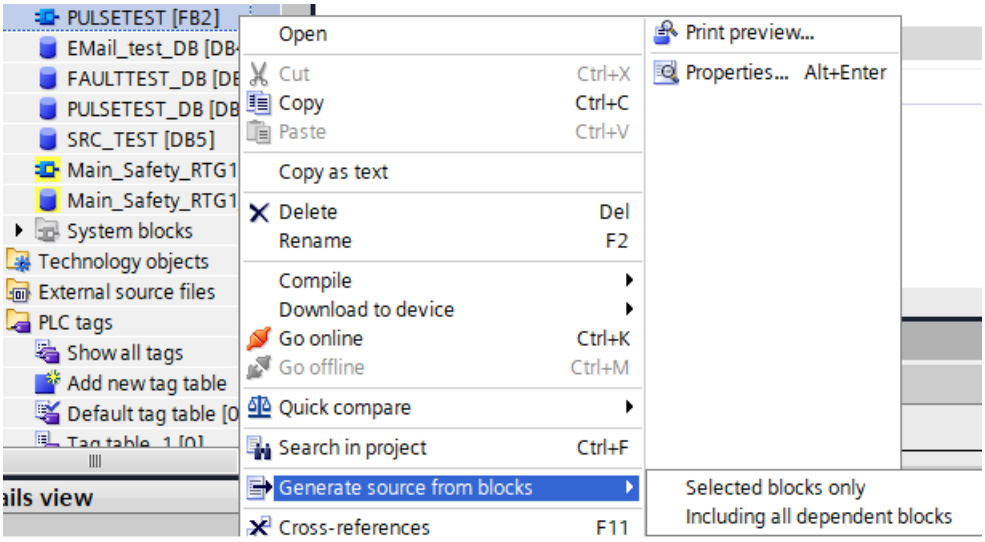

Figure 6. Menu structure of exporting sources in TIA – Portal (Samuli Multaniemi, 2018)

Siemens has also enabled extensive abilities to Copy and Paste data from TIA – Portal and also into it. This allows for variable lists to be created in table form and then imported into TIA – Portal to be created into a Data Block or a Tag List for example. This functionality combined with a predefined UDT – structured data storing makes for a very quick way to build blocks for the base program features. Copying and Pasting actual program blocks works as well, but it is always a good idea to run extensive tests to make sure there are no compatibility or compiling issues with this method.

Tools for creating table form resources are not the only ways that external software can be used in PLC program design. Building blocks in textual languages, mainly SCL can be arduous especially in the classic STEP7 SCL editor as it does not offer any modern programming editor environment features. Features such as highlighting function or instruction endings and a variable table instead of text form input make design work much quicker and improve the whole experience. Now, most of the quality of life features have been introduced to the SCL editor in TIA – Portal, but for SCL development meant for STEP7, external text processors or programming editors can be utilized. As the code is in text-form, it can simply be copy-pasted into the editor in STEP7 or saved as a source file externally and imported into the project as a block from the source.

5.4 **Block and Variable Attributes**

The aforementioned attributes for blocks and individual variables offer some powerful features for design purposes. They were not studied in depth as the scope of this thesis work is quite broad as is. That said, their

advantages are introduced here for future reference if some continuation work is done on the base programs later.

System attributes are premade functions for variables and some of them even whole function blocks. They are more visible and found on the CFC – programming environment but do work equally on other programming languages. Most commonly used ones are described as follows:

“S7_dynamic” is usually activated on every input/output variable and will cause automatic registration of testing mode when it is turned on. This means that its value can be monitored and controlled during the testing mode in the S7 – CFC – environment.

“S7_link” allows the designer to choose whether it is possible to interconnect a signal from outside of the block into the variable or not. This is useful in cases where the value or signal will not be needed to be changed easily and quickly. Such case could be for example a substitute value in the case of an error.

“S7_visible” simply controls whether the variable is visible in the programming environment or not. This is useful for removing clutter in the programming environment and ensuring that the value will not be modified by accident. Base program blocks tend to have dozens of inputs as they have a lot of functions and parameters. A comparison between a block with all inputs visible and a block with only most commonly modified ones visible can be seen below in Figure 7. It is clear to see that in the program view environment it is better to use the more compact block rather than having all inputs visible. The hidden inputs can be modified from the Properties – window of the block and usability is improved while maintaining all functionality.

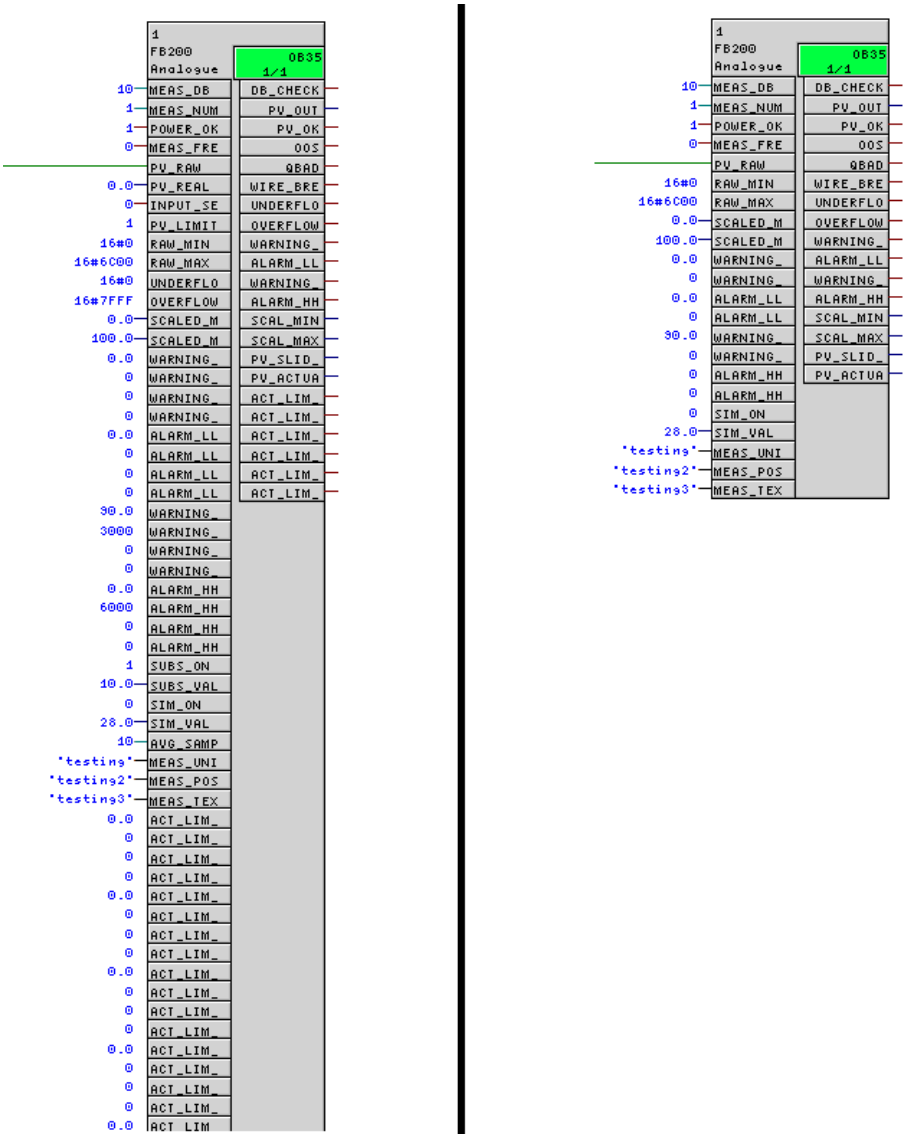

Figure 7. A comparison between all inputs visible and only most commonly used in operation visible (Samuli Multaniemi, 2018)

5.5 **Data Storing and User Defined Datatypes**

Having all data stored in a structured and predefined form helps out with the whole design process. Assigning structured form for data, is a way of optimising design flow. Structure defined for a specific use case where multiple instances of process equipment have identical features and I/O. For example, a structure for an analogue measurement would hold all data needed for a single measurement, such as raw value in, raw value scale minimum and maximum, transformed and scaled value, its range, high and low alarms and much more. The Data Block (DB) holding these will then have an array of uniform data structures which can be accessed easily by the base program function block.

The program block can be built to automatically operate based off of only the running order number of the function. For example, for measurement number 8, the block finds the 8th data structure from the predefined measurements DB and will store and use the values from the right structure. This is demonstrated in the Figure 8 below.

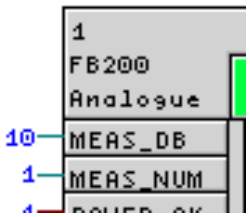

Figure 8. Analogue Measurement Data Block and measurement number designation inputs (Samuli Multaniemi, 2018)

In the figure, the "MEAS_DB" is an input for entering the Data Block number for the measurement data for the project. "MEAS_NUM" is the aforementioned running number, which is used to structure each measurements data in its correct memory area in the data block. Similarly, a separate DB can be created if only some of the data needs to be relayed onto the control room Human Machine Interface (HMI).

The structuring of data entities can be also taken one level further and User Defined Types (UDT) can be utilized in data block building. A UDT is a similar way of structuring data as described earlier, but in the case of UDT, the structure is made into a template of a sort and named as such. For example, a UDT called "Measurement" could hold the data described in the previous paragraphs. Its main advantage is easier data handling and designating. Due to personal preferences in the customer company, the UDT principles were not implemented in this thesis work. A structured form of data storing was used instead.

6 PLC I/O DIAGNOSTICS

Siemens has always had great tools for diagnostics purposes especially in their PLC – equipment, as reliable operation and safety is a number one priority in automation. The robust diagnostics in the S7 – line were studied in this thesis starting with the 300- and 400-series and continuing with its new 1200- and 1500-series based controllers and I/O – systems. The older modules had System Fault (SF) diagnostic signal lights on most modules which is demonstrated in the Figure 9 below.

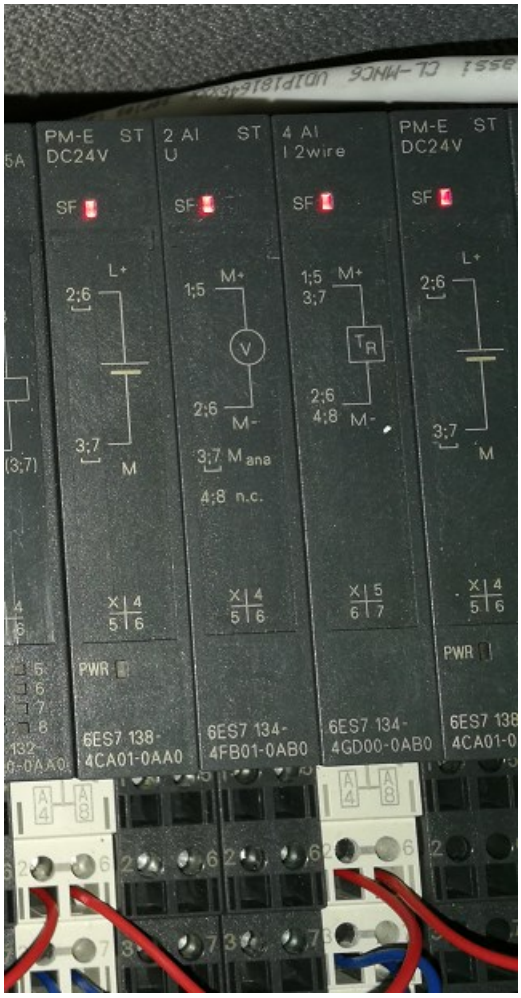

Figure 9. ET200S – I/O modules with their SF – lights on (Samuli Multaniemi, 2018)

The new 1200- and 1500- modules have replaced the SF – lights with “DIAG”- lights on all modules and will report diagnostics information very extensively. The diagnostics and fault states reported on older equipment could be mixed with System Fault conditions arising from other sources, which could cause confusion to the user and maintenance personnel.

Fault signals vary between different module types, as their operation is based on different electrical properties. Especially for AI- and AO-modules, a quite wide range of diagnostics information is available for the module itself and the channels separately. This information is extremely helpful in troubleshooting situations. The diagnostics data can be relayed into the control room HMI straight from the base program block and the operator can see right away what type of fault condition is active and the location of the fault as well. With large and complex systems, its often hard to find a wire break for example and knowing which wire to follow and check saves a lot of time.

6.1 **Module diagnostics**

"No supply voltage L+" is given if the I/O-module is not receiving sufficient supply 24V voltage. This is needed for the operation of the module, and if this condition is true, it is possible that none of the channels work at all. This condition gives out an error code of 11 in hexadecimal base, which translates to 17 in the decimal base. These codes can be read from the PLC CPU diagnostic buffer. (Manual A5E34941201-AA, Siemens, AG, 2015)

"Error" is given if the module encounters some internal fault or error. This will halt the operation of the module which needs to be replaced. The error code given is 9 in hexadecimal base, which translates to 9 in decimal base. (Manual A5E34941201-AA, Siemens, AG, 2015)

"Wire break" is the most common diagnostics alarm encountered in the real life and can be caused by various conditions. The impedance of the connected circuit of the encoder can be too high and therefore not allowing enough current to flow to the module. In addition, a disconnected wire in any channel will also trigger this alarm. The error code given is 6 in hexadecimal base, which translates to 6 in decimal base. (Manual A5E34941201-AA, Siemens, AG, 2015)

"Short-circuit" is given if the encoder supply is fed to ground, which means that its purely a wiring error. The error code given is 1 in hexadecimal base, which translates to 1 in decimal base. (Manual A5E34941201-AA, Siemens, AG, 2015)

"Channel temporarily unavailable" is given only in the case of a firmware update and shouldn't be encountered during normal use. The error code given is 1F in hexadecimal base, which translates to 31 in decimal base. (Manual A5E34941201-AA, Siemens, AG, 2015)

In addition, output modules have a setting for a substitute value which will be set in the case of any fault state. Initially the substitute value is 0. (Manual A5E34941201-AA, Siemens, AG, 2015)

6.2 **Channel diagnostics**

Channel faults and diagnostics can be read from the signal value, this can be evaluated in the program blocks and in some cases the error can be identified. In any case, the error can be reported. (Manual A5E34941201-AA, Siemens, AG, 2015)

"High limit violated" is given if the signal exceeds the factory-defined high limit for the signal, for example in 4-20 mA modules, 22.81 mA. In percent-grade, the overflow occurs after the signal hits 117.589% of the scale. In this case, the channel signal readout will be 7FFF in hexadecimal which

translates to 32767 in decimal. The area between 100% of the scale and the overflow threshold is called the “overrange” and while it indicates faulty calibration or process values, will not inherently trigger any diagnostics actions. (Manual A5E34941201-AA, Siemens, AG, 2015)

“Low limit violated” is given if the signal falls below the factory-defined low limit for the signal, for example in 4-20 mA modules, 1.185 mA. In percent-grade, the overflow occurs after the signal hits -17.589% of the scale. In this case, the channel signal readout will be 8000 in hexadecimal which translates to -32768 in decimal. This signal is shared with the wire break fault, that said, the measuring circuit would have to have a major fault to only produce less than 1.185 mA in any other condition than the wire break. The area between 0% of the scale and the overflow threshold is called the “underrange” and while it indicates faulty calibration or process values, will not inherently trigger any diagnostics actions. (Manual A5E34941201-AA, Siemens, AG, 2015)

6.3 **Use cases for diagnostics**

The diagnostics gathered from modules and the rest of the physical equipment can be effectively used to shorten downtimes and assist in troubleshooting situations. Designing the functionality of diagnosis gathering and processing into the base program blocks allows for the data to be forwarded into an alarm logic program and the SCADA – system in the control room. A simple way would be just to report a general fault signal, activating from any fault or alarm. Separating the faults into their own signals brings much more perks than it is cumbersome.

As the individual fault and diagnostics signal are displayed to the operator, the problem solving of fault situations is easy and straightforward. In the typical situation where only a general system fault is given, the operator or maintenance technician has to spend time investigating the source of the fault.

It is also unfortunately common in real-life process environments to have active fault signals clogging up the diagnostics buffer and even the actual HMI alarm list. This happens for example in process areas that have been dismantled, but automation components are still present and report fault signals. It is much more difficult to find the fault from here compared to reading it from the control room screen. Not to mention that the Siemens designer software is necessary for reading the diagnostics buffer of the CPU.

7 PROGRAM BLOCK DESIGN

7.1 General

As mentioned earlier, the base program function blocks were designed with design optimization first in mind. This means that most of the functions housed in them usually are implemented before or after them in individual logic instructions. With the Siemens design software, unnecessary I/O can be made invisible and therefore they won't interfere with design flow. The benefits of having everything in the same place largely outweighs the drawbacks of visual clutter that may occur if all of the functions are activated from the function block.

In addition to this, inputs and outputs that are connected to the HMI system are modified from the Data Block. This is to continue the principle that data is entered and used from one place only, in order to keep the whole system tidy and organized.

7.2 Analogue Input and Measurement

7.2.1 Role in a system and functions

Analogue Measurement is a key part of any process related automation system. The role of the program block is simple: to introduce measurement values into the system in a correct manner and to report if there are errors or faults. The base program block for analogue measurement handles transforming the raw value in integer form from the analogue electrical signal from the transmitter into the real value the sensor is sensing and feeding the transmitter.

As mentioned before, measurements are how the PLC monitors everything in the process. This means that especially in industrial process settings there will be often dozens of analogue measurements. With modular design, customizable limits, alarms, delays and other features, a single base function block type can be used for all of them.

A broad base program block for analogue measurement will also handle a great deal of other functions which include at least the following: raw value range, scaled real value range, fault signal detection and reporting from the Analogue Input (AI) I/O – module, high and low limits and alarms, substitute mode and value, simulation mode and value. In addition, the beforementioned principle of the designer inputting the running number of the measurement will then store all data related to that measurement based on the structure of the UDT. This way the data is in a logical order and will be easy to locate and use.

As mentioned earlier the operation cycle time for all of the blocks is important, but as measurements are the basis of all safety related automation, the analogue measurement block simply cannot operate slow or delayed. Process values can change incredibly fast and the difference between a few millisecond cycle time and a few hundred millisecond cycle time is enormous. On modern larger controllers this is rarely an issue, but the base programs should work on all controllers so attention needs to be paid to the operation cycle times.

7.2.2 Signal Input and Processing

Analogue Input I/O – modules operate in the way of measuring either current or voltage of a channel, usually in the range of either 4-20mA or 1-5V. This value is then transformed linearly into an integer value in the range of -27648 to 0 for negative signal values and 0 to +27648 for positive signal values. Negative signal is not used in normal operation. The whole integer scale for a 16-bit integer is from -32768 to +32767, and some of the range is left as a buffer so over- and underflow can be measured. In addition, fault signals will give out a case-specific integer value.

In order to make the base program blocks use range broader, multiple different signal types can be input. These include raw analogue signal in integer form, already scaled and transformed real form or 0-100% values although not commonly used. An input for the block is also used to choose the signal input type.

Direct signal from the Analogue Input (AI) module will be in integer form as described earlier. The program block will also need transforming and scaling in form of limits to transform this integer value into the desired real-life signal values. This scaling is done to match the range of the transmitter after which the signal will be what the sensor is sensing and sending to the PLC through the transmitter. For example, a 4-20 mA signal is in the range of 0-27648 and if the desired scaled real-life value range is 0-200 Pa, the real-life value for a signal integer value of 15000 is calculated as follows:

$$X_{scaled} = X * \frac{I_{scaledmax} - I_{scaledmin}}{I_{rawmax} - I_{rawmin}}$$

$$X_{scaled} = 15000 * \frac{200 - 0}{27648 - 0} = 108.507$$

The scaling function of the block should only used if the signal is in the “raw” integer form. This means that it is in the same form that is received from the input address of an Analogue Input card.

The simulation function of the block is directly related to the signal value. It allows to use a simulated value as the output of the block for testing and troubleshooting situations. The block has two inputs for this function, the SIM_ON and the SIM_VAL. If the SIM_ON – input signal is 1, the simulation function is active and the actual signal input is replaced with the simulated signal which is read from the SIM_VAL – input.

The substitute value function is very similar to the simulation one, though it serves a very different cause. In the substitution function, the input signal is also replaced, this time by the value in SUBS_VAL. The purpose of this is to produce a known, safe signal in case of equipment malfunction or fault. Generally, a substitute value function is kept active, as in process industry faulty measurements can lead to unwanted automation system behaviour. For example, if a flow measurement faults and shows a signal of no flow which the automation system sees as no material in the process line. This might activate logic of stopping a succeeding conveyor and could cause a build-up if material actually does still flow in the process line. In all cases, automation systems should rely on multiple signals and have fail-safe logic preventing action like this. That said, unknown situations are always a major hazard to the automation system as every fail condition is hard to simulate and test in the commissioning phase.

The block is also equipped with six additional “action threshold” – limit functions. This means, that when the signal either falls below or exceeds predefined limit, after the set delay time a corresponding output signal is set to 1, until the condition ceases, which is defined with a hysteresis value. This feature is found on Distributed Control Systems (DCS) and saves the designer time by existing inside the measurement block. This function and its inputs are disabled by default as they have a lot of parameters and might not be used in some cases. They are configured similarly as the warnings and alarms explained in the next chapter.

In metrology, signal-to-noise ratio is extremely important. To combat noise, the program block has an inbuilt sliding average from 1 to 40 last measurement cycles, based on designer or operator choice. This can be used if a more stable, albeit slower process value signal is needed. It uses an array of samples, each taken on a program cycle, and then calculates an average of these.

The program block also has functionality for freezing the measurement signal based on an input signal. This can be used if for example some unwanted changes in the operation environment cause the measurement to show invalid signal values. A practical example could be a level measurement in a freight ship. If the freight ship tilts in a direction too much, the level measurement can show incorrect values. A signal from a sensor measuring the tilt can trigger the measurement freeze to stop possible fault conditions caused by the level measurement invalid signal.

7.2.3 Warnings and alarms

The block is equipped with warning and alarm thresholds. Warnings are designated with an “H” for a high warning and an “L” for a low warning. Alarms are designated with an “HH” for a high alarm and an “LL” for a low alarm. The block also has separate outputs for signalling alarm conditions.

All warnings and alarms have their individual bypass states, activation delays, and hystereses. In this case, hysteresis is value representing the amount, that the process value needs to surpass from the alarm or warning limit to deactivate the alarm signal. For example, if a high alarm HH is set at 90 degrees Celsius, and the hysteresis is set at 4 degrees Celsius, after tripping the alarm, the process value needs to return to 90 – 4 = 86 degrees Celsius for the signal to deactivate. This is done so the alarm state will not repeatedly activate and deactivate while close to the limit, as in real life, process values tend to be quite unstable and environmental noise is present.

Warnings are essentially notifications to the operator and the system itself that the process value is nearing the limit and unsafe process values. They will give out a warning signal to the HMI/SCADA. Alarms are defined in a way that they should not ever be triggered in normal operation as they signify a fault condition or an unsafe process value. Alarms can act as interlockings for a device control program.

For the measurement circuit, the block has three separated condition signals. Underflow is triggered for falling below a set limit, overflow for exceeding a set limit, and wire break. Under- and overflow limits vary by the used Input Module Card, which is the reason for their user-input limit values. Wire break on the other hand will always give out the signal of 8000 in hexadecimal which translates to -32768 in decimal. Wire break usually means that the measurement circuit has a broken wire connection. Other causes were listed in chapter 5.1 “Module Diagnostics”.

The block also has a combined “Measurement OK” and “QBAD” – functionality. Measurement OK – signal is given when no measurement faults or alarm states are active. The inverse of this, QBAD, simply signifies that some fault condition is active. This is an output of the base program block and is found for example on the Siemens Safety PLC program logic channel driver blocks.

7.2.4 String Identification

While not directly needed for operation, introducing string inputs to the program block helps with data management. These string inputs include a process position (for example 1HLA10CT001), a short explanation (Boiler 1 Temperature 1) and a measurement unit (degrees Celsius). Designing a SCADA system is a lot faster, when all data comes from a same source already in structured form. Also, a prebuilt template, usually called faceplate can be used if all of the data can be read from variables and nothing is written from the SCADA system itself.

In practise, entering the string would happen usually into the input of the program block from where it gets passed onto the HMI communication Data Block. Figure 10 found below is an example of the string data from the HMI communication measurement data block.

+12.0	MEAS_1_UNIT	STRING[12]	'bar'
+26.0	MEAS_1_POS	STRING[28]	'1HLA10CP001'
+56.0	MEAS_1_TEXT	STRING[40]	'Boiler 1 Primary Air Pressure'

Figure 10. Measurement unit, position and text in Measurement Data Block in STEP7 (Samuli Multaniemi, 2018)

8 PROGRAM BLOCK FUNCTIONALITY TESTING

8.1 Simulation Testing

The testing of the base program blocks was quite straightforward. A very capable simulator for PLC's called "PLCSIM" is included in the Siemens software environment. It can be used to troubleshoot, test, validate and demonstrate features and even full programs. It works by being a clean slate for a PLC, which is then formed from the hardware configuration built within the SIMATIC software. The hardware configuration is a direct component and module listing from the catalogue in SIMATIC. In the Siemens environment, hardware configuration is formed through a catalog and additional device configuration files can be found from device manufacturers sites to make hardware configuration extremely simple. Shown below in Figure 11 is the hardware configuration view window.

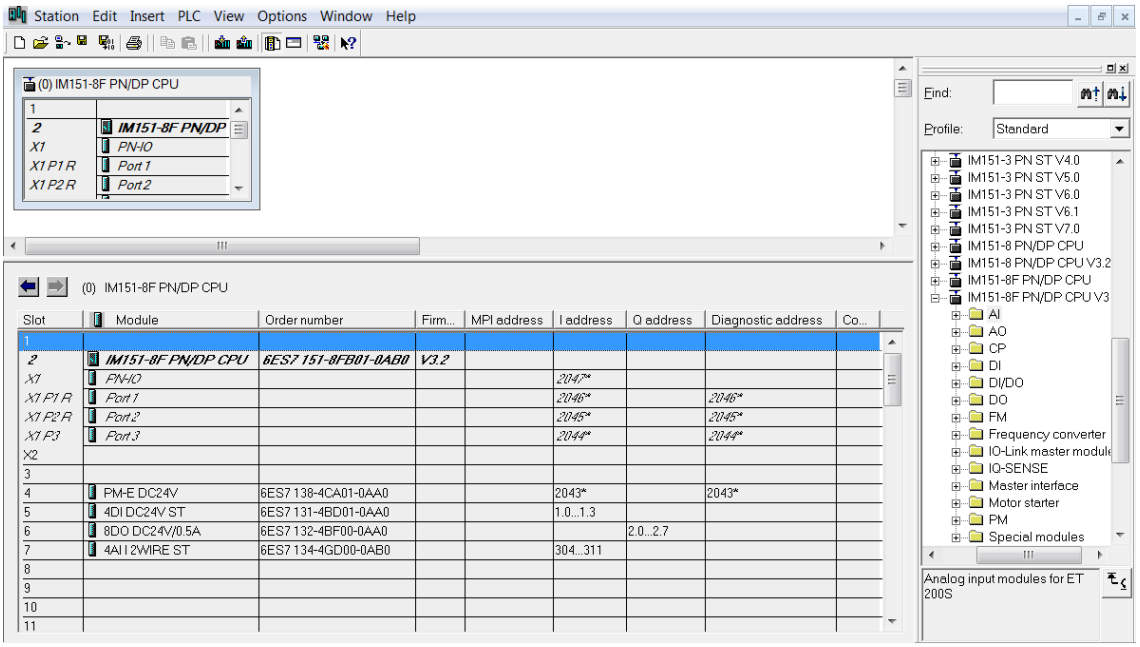

Figure 11. Hardware Configuration of the testing equipment in STEP7 (Samuli Multaniemi, 2018)

The hardware configuration is downloaded onto the simulator just like for a physical PLC. The simulator PLC is not perfect and can sometimes behave differently than a physical PLC. It is always advised to test programs in a physical PLC as well.

Connection type used in the thesis work was “Profinet” which is a fieldbus standard developed by Siemens and is based on the older “Profibus DP” standard, also developed by Siemens which is the most widely used fieldbus in the world. Profinet is built around Ethernet and CAT-cabling as its modem of connection and is rather simple to use for connections. (Siemens AG, 2018)

The only information needed for connection is the IP-address range of the device that the connection needs to be made to. In the newer software environment TIA-Portal, even this step is not that crucial as the software can add an IP-address in the correct area. That said, it is always important to be aware of the address of your devices. Siemens has a tool for checking the addressing, “Edit Ethernet Node -> Browse”. From here it is also possible to modify the addressing and parameters of a device. The Ethernet Node management tools is shown in the Figure 12 below.

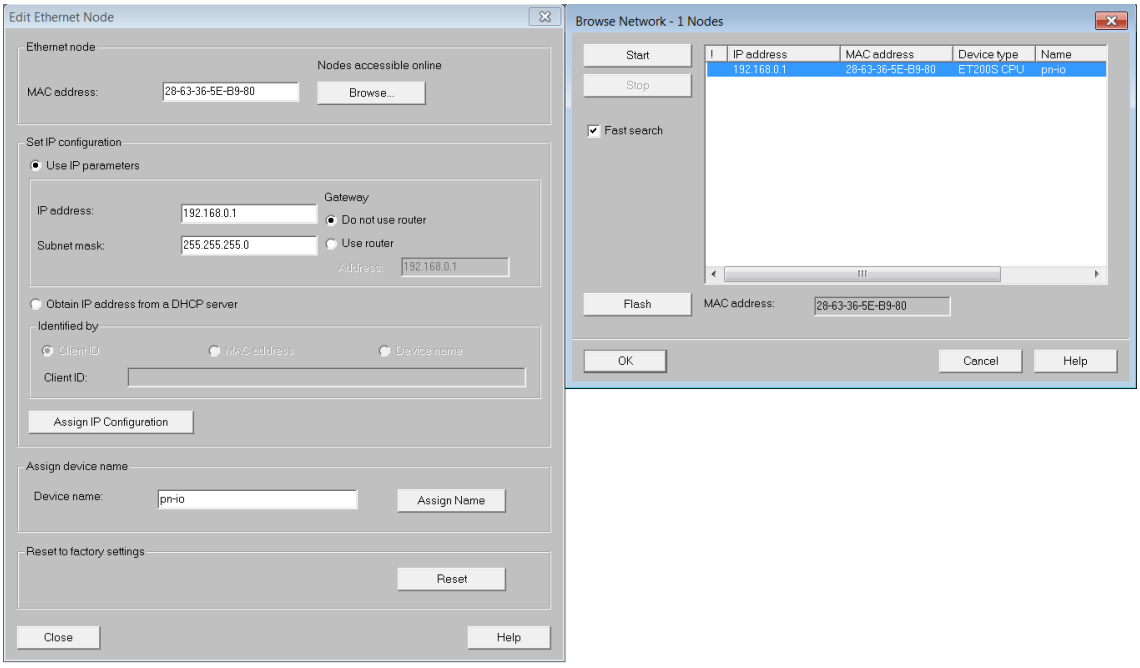

Figure 12. Ethernet Node Display and Searching in STEP7 (Samuli Multaniemi, 2018)

After loading the desired hardware configuration onto the simulated PLC, the program itself can be downloaded. As mentioned earlier, the testing of this thesis work was conducted in the Continuous Function Chart (CFC) language environment. Downloading a program onto the simulated PLC does not differ from downloading onto a physical PLC at all.

After downloading and changing the state of the PLC to “RUN” the program can be monitored and tested with the “Test Mode” where program units can be monitored and modified. Test Mode is explained more in depth in Chapter 4.3 “Design and Testing”.

8.2 **Physical Testing Equipment**

Simulation testing was done during the design phase and programming errors were fixed. After the block performance was fully functional, the physical testing equipment was taken into use. The testing equipment consisted of a Siemens ET200S rack, including the IM151-8F PN/DP CPU, which is more than capable of running the tested programs. The ET200S is a S7-300 – based line of slim, DIN-rail mounted modules. ET200S modules can be used as a distributed I/O – rack for a CPU placed in a distance as well, for example in a main electrical room and the distributed I/O in a control cabinet in the process field.

A compact PLC rack and a 24V power supply used to power equipment and signals used in testing of this thesis work is shown in the Figure 13 below.

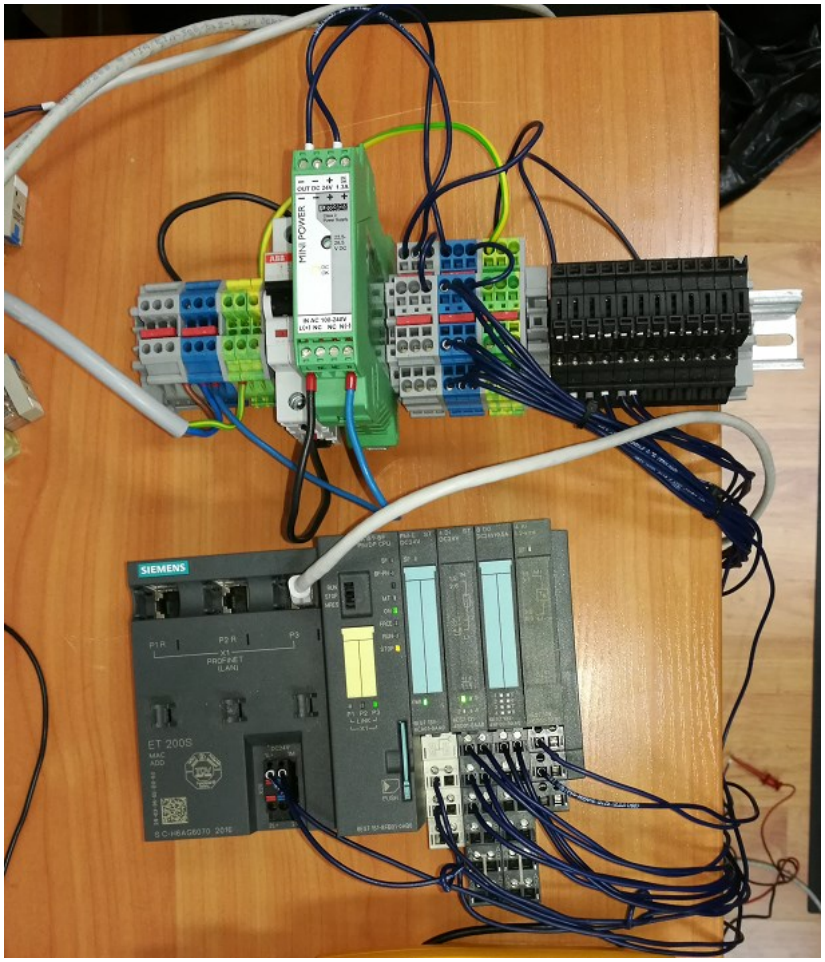

Figure 13. Siemens S7 ET200S PLC – rack and a 24V power supply testing equipment (Samuli Multaniemi, 2018)

8.3 **Analogue Measurement Testing**

Analogue Measurement from a functionality standpoint is simple. For the testing purposes the signal was generated from a Fluke 773 Milliamp Process Clamp Meter with simulation functionality. The meter allows to feed up to 24 mA of DC current into a circuit. This is specifically designed to be used in simulating and testing environments. The simulating function of the meter allows to feed the signal even by increments of 0.01mA at a time, which is more than sufficient for most testing scenarios. Some meters and calibrators with similar functionalities offer up to 0.001mA increments but using such accurate signals was not necessary for the testing situations of this thesis work. The Fluke 773 meter can be seen in a testing scenario below in Figure 14.

Figure 14. Fluke 773 Process Meter with milliampere transmitting functionality (Samuli Multaniemi, 2018)

As for the program block testing, the analogue electrical signal is fed to the AI – module and the desired parameters are set in the program. After these, the block should give out the correct measurement value. The I/O modules from Siemens seem to be very high quality and measurements were very accurate. Additional functionality, warning, alarm and fault testing was successful and the block performed correctly. For confidential reasons, the programming or testing functionality cannot be shown more in depth

9 CHALLENGES DURING THE PROJECT

The commissioned project was very enjoyable to work on and reflected on the authors studies perfectly. PLC's are the heart of all modern automation systems and getting to work on the design of them for a project of this size was extremely educating. That said, the project was not all smooth sailing and had plenty of challenges.

Firstly, designing a full library of base program blocks is a very large undertaking. The sheer amount of preparatory research, planning and defining needed turned out to take quite a bit longer than anticipated. Especially the research work did reveal some helpful information but was not fully thought of in the planning of the work.

Even though designers at TAS, including the authors supervisor, had designed such program blocks earlier, compiling a full library of them turned out to involve a lot of design choices, new additions and even modifications of the premade definitions. Even simple sounding details like choosing the programming environment led to difficulties, as mentioned earlier the Siemens software package for the S7 – product family is so fragmented nowadays. Hopefully Siemens can find solutions to these software issues, as competition in the automation field is stronger than ever and high-quality hardware might not be enough to keep Siemens at their industry leader spot in the European market.

Programming work itself was challenging in its own ways as well. Building the program blocks required some new knowledge especially of dealing with the CFC – language environment, as the author was not very familiar with it beforehand. Siemens products have been used for decades now and this offers a lot of support as many other people have been dealing with similar programming situations and troubleshooting online is made very easy with great resources available even on the official support forum of Siemens.

Finally, spending resources and time on developing new program tools with software originally developed over 20 years ago is always a bit risky, especially as the successor product has already been available for 7 years now. In this case, it definitely seems like the STEP7 – software package is going to be relevant for long time and it is still worth investing in research and development like the thesis work conducted here.

10 CONCLUSION

The project goals were met in a sense that a lot of the research conducted was worthwhile and the block designed and compiled performed extremely well. Unfortunately, some of the functions planned were not possible or feasible to be implemented and some limitations were met in the Siemens software. Being aware of such things is an advantage itself and can influence design choices and troubleshooting in the future.

Being able to finish a full library of base program blocks for this thesis work was the initial goal. As it turned out to be much more extensive than thought and the research work needed ended up being quite labour-intensive, the goals were altered. Documenting the research, design methodology and producing a finished base program block for demonstration and example purposes was a great result and sufficient for the scope of a Bachelor's thesis project.

The rest of the blocks will be designed with the same principles in mind and for the purposes of this thesis work, a full completion of the library is rather irrelevant. The structured, feature-based implementation is applied to all of them, just with different functions and operational details.

The development of the base program library will be continued by the author and the library will be used in applicable projects as planned beforehand. The existence of such resources can make or break major bidding deals in the future and investing in building the library was extremely needed.

REFERENCES

”Tampereen Automaatiosähkö Oy”, Tampereen Automaatiosähkö Oy, Retrieved 9.5.2018, https://www.tas-automation.fi/

”Ohjelmoitavat logiikat”, Siemens AG, Retrieved 12.5.2018, http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/ohjelmoitavat_logiikat_simatic.php

”Engineering Essentials: What Is a Programmable Logic Controller”, MachineDesign, Carlos Gonzales, 2015, Retrieved 28.5.2018, http://www.machinedesign.com/engineering-essentials/engineering-essentials-what-programmable-logic-controller

SIMATIC Programming Instructions, Creating Blocks for PCS 7, Siemens AG, Retrieved 9.5.2018, https://cache.industry.siemens.com/dl/files/962/7213962/att_79822/v1/S7Prog_e.pdf

“Automatic generation of an S7 data block with STEP 7 V5.x from an MS Excel 2003 table”, Siemens AG, Retrieved 18.5.2018, https://support.industry.siemens.com/cs/document/15162450/automatic-generation-of-an-s7-data-block-with-step-7-v5-x-from-an-ms-excel-2003-table?dti=0&lc=en-WW

Manual A5E34941201-AA, Siemens AG, Retrieved 15.5.2018, https://cache.industry.siemens.com/dl/files/183/109475183/att_837325/v1/et200sp_ai_8xi_2_4_wire_ba_manual_en-US_en-US.pdf

“Profinet”, Siemens AG, Retrieved 22.5.2018, http://www.siemens.fi/fi/industry/teollisuuden_tuotteet_ja_ratkaisut/tuotesivut/automaatiotekniikka/teollinen_tiedonsiirto_esim_profinet/profinet.htm