YRKESHÖGSKOLAN
NOVIA

Development of Siemens S7 Protocol to Modbus Converter

Timmy Lund

Bachelor's Thesis

Electrical and Automation Engineering

Vaasa 2022

YRKESHÖGSKOLAN NOVIA
www.novia.fi

AB YRKESHÖGSKOLAN VID ÅBO AKADEMI

Prefekt Kristian Blomqvists beslut

Nr	Datum	
02/22	12.1.2022	Studerande Timmy Lund (studerandenummer 1700445) vid utbildningen i el- och automationsteknik har per den 12.1.2022 inkommit med en anhållan om att få sitt examensarbete sekretessbelagt i enlighet med rutin *GQAP26 Examensarbete vid Yrkeshögskolan Novia*. Examensarbetet bör sekretessbeläggas med hänvisning till att det innehåller affärshemligheter. **Beslut:** Lunds examensarbete skall innehålla en offentlig del och en sekretessbelagd del. Den sekretessbelagda delen är konfidentiell fram till den 12.1.2027.

Vasa 12.1.2022

Kristian Blomqvist
Prefekt

Distribution:
Sakägare
Utbildningsledare
Handledare
Studerandeservice Vasa
Tritonia

DEGREE THESIS

Author: Timmy Lund
Degree Programme and place of study: Electrical and Automation Engineering, Vaasa
Specialization: Automation
Supervisor(s): Joachim Böling – Novia University of Applied Sciences
Anders Franzén – ABB Energy Industries

Title: Development of Siemens S7 Protocol to Modbus Converter

---

Date: 10.4.2022 Number of pages: 30 Appendices: 0

Abstract

This thesis work is done for ABB Energy Industries and will be used in Wärtsilä marine systems. Wärtsilä is changing their monitoring system from WOIS to WDCU. The monitoring system displays information from the engine control unit using Modbus and from the engine-specific PLC: s using S7 protocol. The old WOIS was capable of reading both protocols, but the new WDCU is only capable of reading Modbus, therefore the S7 communication has to be converted to Modbus.

The main tasks were to develop a device for converting S7 protocol from each engine UCP to Modbus protocol and feeding it to the WDCU. This included finding a suitable device for the task, developing converting software, and standardizing the communication lists. The requirements for the device were a Siemens PLC of suitable model and the PLC software were to be done in TIA portal. Generating of the communication lists were standardized and automated with Microsoft Excel and macros using the programming language VBA.

The finished product is easy to implement on existing projects without the need to re-configuring existing hardware. When upgrading from WOIS to WDCU the old communication list is used to generate the new lists which are used in the WDCU. The converter is to be developed and tested with simulated communication to begin with, later tested in its real environment using PLC: s and WDCU.

Language: English
Key Words: protocol conversion, PLC, WDCU, WOIS

EXAMENSARBETE

Författare: Timmy Lund
Utbildning och ort: El- och automationsteknik, Vasa
Inriktning: Automationsteknik
Handledare: Joachim Böling – Yrkeshögskolan Novia
Anders Franzén – ABB Energy Industries

Titel: Utveckling av Siemens S7-protokoll till Modbus Konverter

---

Datum 10.04.2022 Sidantal 30 Bilagor 0

Abstrakt

Detta examensarbete har gjorts på uppdrag av ABB Energy Industries. Wärtsilä implementerar deras nya övervakningssystem, som kallas WDCU, för marina applikationer. Det tidigare övervakningssystemet som kallas WOIS läser både Modbus och S7-protokoll men WDCU läser endast Modbus. Motorstyrenheten kommunicerar via Modbus protokoll och de motorspecifika PLC:n kommunicerar via S7-protokoll. För att överföra S7-kommunikationen till WDCU:n måste den först konverteras till Modbus protokoll.

De huvudsakliga målen var att utveckla en anordning för konvertering av S7-protokoll från varje enskild motor UCP till Modbus-protokoll och överföra det till WDCU: n. Detta inkluderade att hitta en lämplig apparat för uppgiften, utveckla konverteringsmjukvara och standardisera kommunikationslistorna. Kraven för apparaten var en Siemens PLC av lämplig modell och att PLC-mjukvaran skulle skapas i TIA portal. Generering av kommunikationslistorna standardiserades och automatiserades med Microsoft Excel samt makron baserat på programmeringsspråket VBA.

Den färdiga produkten är enkel att implementera i existerande projekt utan att behöva omkonfigurera existerande hårdvara. Vid uppgradering från WOIS till WDCU används den gamla kommunikationslistan för att generera den nya som används i WDCU. Konvertern utvecklas och testas med simulerad kommunikation till en början och senare i dess naturliga miljö bestående av PLC:n och WDCU.

Språk: engelska
Nyckelord: kommunikationsprotokollskonvertering, PLC, WDCU, WOIS

Table of Contents

1 Abbreviations

PLC – Programmable Logic Controller

WDCU – Wärtsilä Data Collection Unit

WOIS – Wärtsilä Operator Interface System

UCP – Unit Control Panel

ECU – Engine Control Unit

TIA - Totally Integrated Automation

VBA – Visual Basic for Applications

MB – Modbus

CPU – Central Processing Unit

HMI – Human Machine Interface

DB – Data Block

FB – Function Block

TCP/IP – Transmission Control Protocol/Internet Protocol

IP – Internet Protocol

V – Version

NAT – Network Address Translation

ID – Identification

INT- Integer

DINT – Double Integer

2 Introduction

This thesis work is about developing a protocol converter for Siemens S7 protocol to Modbus protocol and standardizing the communication lists for the conversion. The data in need of conversion is the communication from the engine specific PLC to the monitoring system. The task is given by ABB Energy Industries in Vaasa and the prototype is expected to be done in week 14 of 2022.

The protocol converter is going to be sold mainly as an upgrade from WOIS to WDCU in marine applications but can also be implemented in newbuild marine applications with WDCU as a monitoring system.

2.1 ABB organization

ABB is a global technology company with a history of over 130 years. Their main production is power equipment, robotics, and automation solutions for industries & homes. ABB consists of 105 000 employees around the world divided into different divisions. Relevant to this thesis work is process automation and digitalization which focuses mainly on industrial automation. The division consists of amongst others, energy industries which is focused on automation solutions for powerplants. The segment consists of four main teams and about 3000 employees worldwide. The teams are hydropower, nuclear & thermal power, engine power, and service.

The engine power team consists of about 30 employees situated in Vaasa, Finland. The team provides automation solutions, including HMI, PLC, and hardware, to gas, diesel, and gas-diesel dual fuel powerplants manufactured by Wärtsilä, both for powerplant and marine applications. [1]

2.2 Disposition

The document is structured according to the following. Chapter 2 is a brief introduction to the thesis work and ABB as an organization. Chapter 3 explains why the thesis work is done, what the goals were, and the problem statements of the project. Chapter 4 gives the reader an understanding of the theoretical starting point and the knowledge of both the person making the thesis work and the company. Chapter 5 is the basic theoretical background of

subjects to be used in the thesis work. Chapter 6 is the procedure of the practical work, including a user manual for the Excel macro. In this chapter also all the functions and code are explained in detail. Chapter 6.4 and 7 is about the testing and implementation of the work, both in individual parts and as a working unit. Chapter 8 is a discussion of how the project went, problems during the development, and suggestions for upcoming improvements.

3 Purpose and problem statement

This bachelor's thesis is ordered by ABB in Vaasa and the finished product will be used mainly in Wärtsilä marine projects.

Wärtsilä is renewing the monitoring system in their marine powerplants, switching from the old WOIS to the new WDCU. The monitoring system displays information from both the engine control unit and the engine specific UCP PLC: s. WOIS reads information from the ECU through Modbus and from the UCP PLC: s through S7. The WDCU is only capable of reading Modbus and therefore the S7 communication needs to be converted to Modbus. In new-build projects of the new system, the converter must be integrated into the UCP PLC but in service projects of the new system the converter cannot be integrated into the UCP and therefore an additional converter device is needed. (Figure 1)

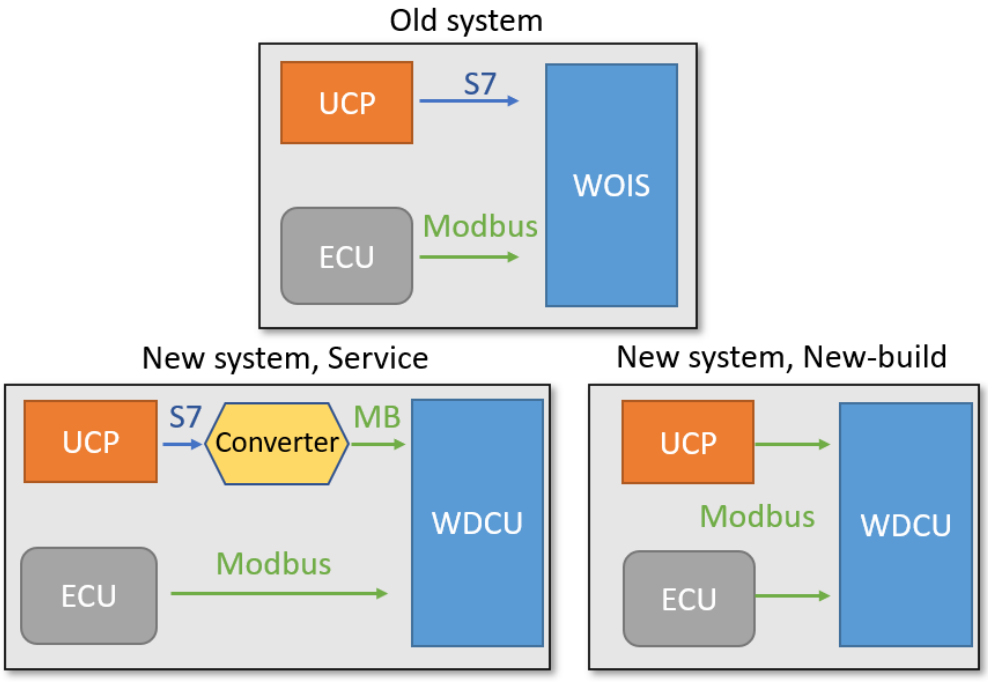


Figure 1 Flowchart, old and new monitoring system

The main purpose of the work is to develop a converter to ensure flawless communication between several engine specific PLC: s and the new monitoring system.

The individual data tags have a standardized pre-given name and address in the engine PLC. When configuring the WDCU the same data tags are defined, in order to keep all project's data tag addresses the same the addresses in the WDCU are standardized. This is accomplished by using an Excel macro when generating the new communication lists for the WDCU.

With standardized communication lists, implementation of the new monitoring system will be faster and simpler. This also favors service work in the future when all the projects follow the same standard.

The final goal is to simplify the protocol converter software enough for it to be implemented directly into the existing hardware of new build projects, meaning the engine specific PLC would communicate internally through S7 protocol and externally through Modbus. This would exclude the need for an additional converting device in newbuilds. In service projects, the additional PLC will be added.

4 Theoretical starting points

There are plenty of protocol converters on the market of all kinds, but the goal of this converter was to keep it flexible and easily re-configurable. The converter also has to be customizable to work with the converting macro. The easiest solution was to develop the converter and not buy a third-party device.

There was a theoretical idea of the program function, syntax of inputs and outputs but the detailed PLC program requires research and testing research and testing to achieve the theoretical idea.

The theoretical starting points of developing the macro were strong due to earlier experience of both excel and VBA. The syntax of the conversion had to be sorted out with the company specialists.

5 Theoretical background

This chapter contains brief theoretical information relevant to the thesis work.

5.1 PLC

PLC stands for programmable logic controller and is a crucial device in automation. The main device is the CPU which runs a logic-based program based on virtual or physical in and outputs. The controller is flexible due to the possibility to simply reprogram it and extend the device with different modules, for example, more in and outputs and power supplies matching the specific need.

The CPU of the PLC cycles the loaded program while in running mode, using the scan cycle (Figure 2). The cycle starts with an internal check, after the inputs are read, using the inputs the program is executed and the result is updated outputs. This is one cycle and the time it takes is called scan time. [2]

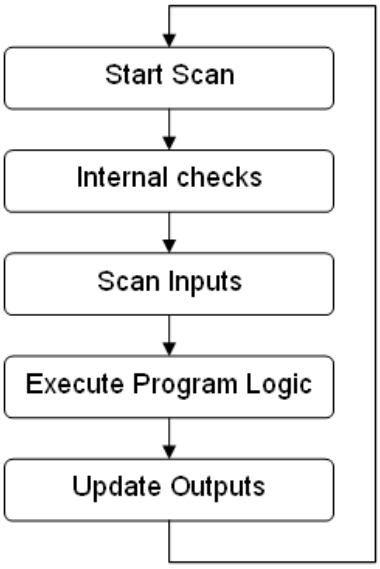


Figure 2 Flowchart, PLC Work cycle

A PLC can be programmed using one or a combination of different programming languages. The languages are, among others,

- Function block diagram (FBD)
- Ladder diagram (LD)
- Structured text (ST)
- Sequential flow chart (SFC)
- Instruction List (IL)

Of these function block diagram and structured text is the most used for larger programs and ladder diagram for smaller programs. [3]

5.2 Siemens TIA portal

TIA portal, standing for Totally Integrated Automation portal is an engineering tool by Siemens for Siemens hardware and software, like PLC: s and HMI: s. TIA portal supports three different programming languages, being structured text, ladder, and function block diagram. The programming languages follow IEC 61131-3 standard.

A SIMATIC-PLC program in function block diagram is structured by four types of blocks: Organization block (OB), Function block (FB), Function (FC), and Data block (DB). The organization block is the block calling the lower blocks. Function block is a block containing one or multiple functions and function is one single function. Every block except the function needs a data block which is a block containing data in form of variables. The data block can be configured as instance or global data block. [4]

5.3 Excel and Macro

Excel is a spreadsheet-based program developed by Microsoft. It is used for organization and calculation of data by using formulas and functions. To make more complex and automated sets of functions in Excel a so-called macroinstruction can be made. This is done using Excels developing tool “Visual Basic for applications” or abbreviated VBA. VBA is an event-driven programing language developed by Microsoft for all Office programs. The VBA code is made and stored in a different window but uses the excel workbook as a user interface, this means that the user of the macro never sees the code of the program but triggers it by actions in the workbook. All built-in excel functions can also be used in VBA.

6 Methods and procedures

This chapter explains the methods and procedures used in the thesis work. First comes the PLC part of the project starting with the choice of hardware, followed by development of the PLC program. As the second part of the chapter comes a user manual of the Excel macro, followed by development of the macro. The last part explains the testing and implementation of the converter.

6.1 Choice of hardware

In order to develop a prototype of the protocol converter, a separate device was needed. The project specifications included some requirements on the device but otherwise the choice of hardware was free. In the planning stage of the project there emerged more requirements on the device. These requirements were the following:

- Capable of communication in both Modbus and Siemens S7 protocol through ethernet (RJ-45)
- Programmable with function block diagram and structured text
- Programmable or compatible with Siemens TIA portal
- Powered by standalone power supply or powered by 24VDC
- DIN-rail mountable
- Reasonable price and performance according to the area of use

Considering the requirements, the choice of hardware was a Siemens S7-1200 series PLC which fulfills all the requirements. Due to the need for only internal/virtual I/O, there was no need for additional modules to the CPU. The precise model of the CPU was based on estimations and comparison to earlier PLC projects, on how big and demanding the program would be and the selected model was the 1214C DC/DC/RLY CPU. The DIN-rail mounted PLC has a built-in program/data memory of 100kB and operates on 20,4-28,8 VDC. It has 14 digital inputs, 10 digital outputs operated by relay, 2 analog inputs, and 2 analog outputs ranging from 0-10VDC. As earlier stated, the physical I/O will not be needed, and the choice of the I/O specifications are therefore based on organization standards. [5]

The PLC stores the program on an external memory in the form of a Siemens SD card. The selection of SD cards ranges from 4 MB to 32 GB. Suitable to this specific use is the 12 MB SD card. The size needed was determined by estimation and comparison to earlier projects.

The chosen PLC has only one integrated ethernet port and because of this, there is a need for an external switch. The model or specifications of the switch is not a crucial detail because it will only act as a junction for the engines, the WDCU, and the PLC itself. The PLC is limited to a maximum of eight simultaneous connections.

The hardware of the prototype will be mounted in a separate box next to or close to the project specific engine unit control panels because of the need for a power supply from the panel.

6.2 Development of PLC software

With the project requirements determined and the hardware chosen the next step was developing the PLC software for converting the data from S7 to Modbus. Based on the manufacturer of the PLC, project requirements, and organization standards the software was developed in Siemens TIA portal V16, and the program is structured with function block diagram.

The standardization part done in excel is explained both from the user perspective and development perspective.

6.2.1 Program structure

The program is structured theoretically according to the following. The needed data for the WDCU is taken from the UCP PLC in S7 format and stored in a temporary data block in the converter. Double integer variables from the partner PLC are split into individual integers when stored locally in the converter and therefore the split variables are restored as a double integer using the “INT to DINT” function. The engine specific Modbus converting block takes the engine specific data from the temporary data block and broadcasts it as Modbus according to preset parameters which include IP address, port number, and the starting address of the Modbus communication. Due to the limitations of the Modbus block every engine must have a specific block, the individual blocks can be enabled or disabled depending on the number of engines. The structure is graphically represented in Figure 3.

To make the converting software as easy to implement as possible all functions are located in one single function block and the function parameters are stored in an instance data block.

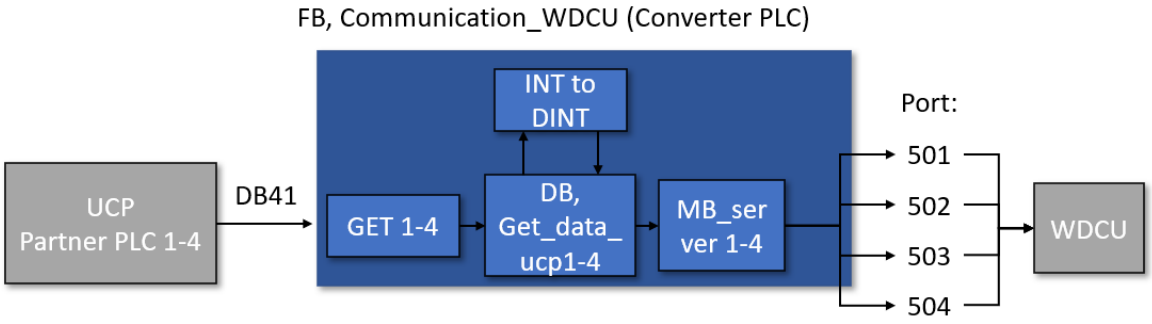


Figure 3 Flowchart, PLC program structure

The program structure depends on if the project is a new-build or a service project and therefore the more detailed explanation of the separate functions are structured according to service or new-build. In a new-build project, the converting software can be loaded directly into the UCP PLC while other configurations are done, which means that the data to be converted already exists in the PLC and does not need to be fetched from a partner PLC. The data can be fed directly to the Modbus block and broadcasted.

If the project is a service project the implementation to the existing project must be as simple as possible and therefore the external converting PLC is used. The external PLC receives data as earlier explained and from there the program structure is the same as a new-build project.

6.2.2 Creating a new PLC program and setting up connections

The development of the PLC program started with the creation of a new project. TIA portal version 16 was installed with belonging licenses. A new project was created with the project name “PLC_converter”, saving path and version were chosen. (Figure 4) [6]

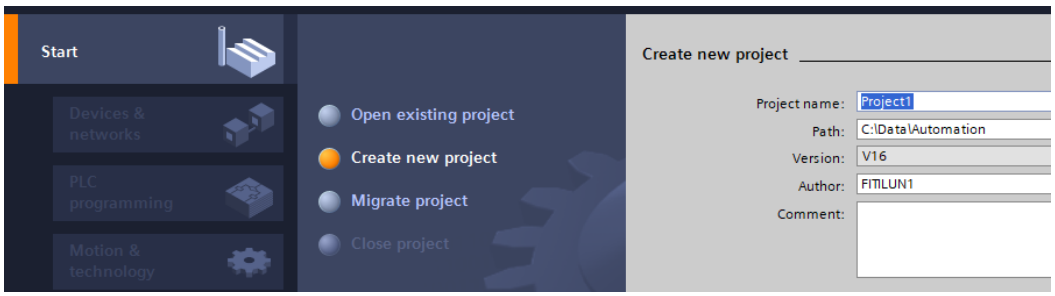


Figure 4 Create a new project, TIA portal

The next step was adding the converting PLC CPU. As earlier stated, the suitable model was the 1214C DC/DC/RLY CPU with 100kB work memory, version 4.4, which was the newest version. (Figure 5)

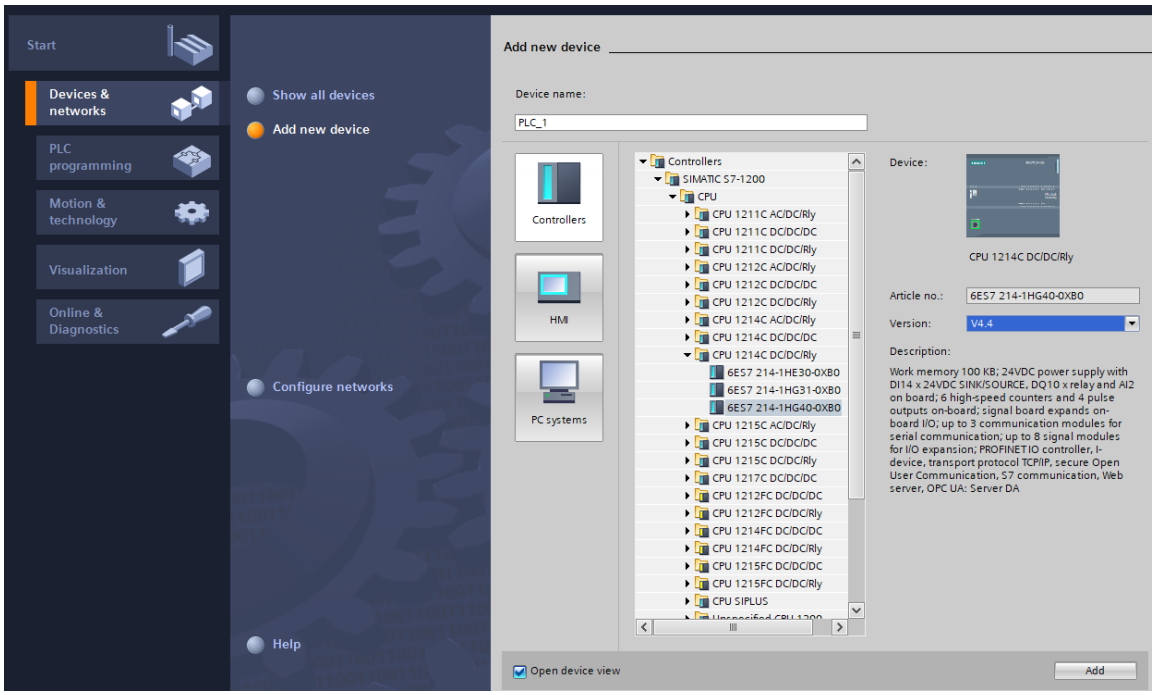


Figure 5 Add PLC CPU

With the converting CPU added, the connections were configured. First, a new subnet was added, this will later link the partner PLC: s to the converter PLC (Figure 6). The configuration is done under the general tab in the converter PLC. Under the same tab, the PLC IP address is configured, this is project specific but has to be in the same IP space as the rest of the system, otherwise the subnet space has to be enlarged or a NAT rule has to be made in the firewall between. (Figure 7)

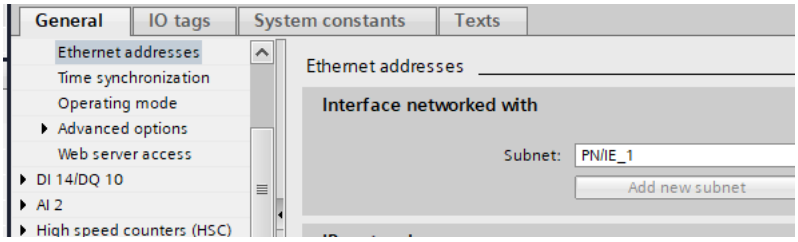


Figure 6 Assign subnet

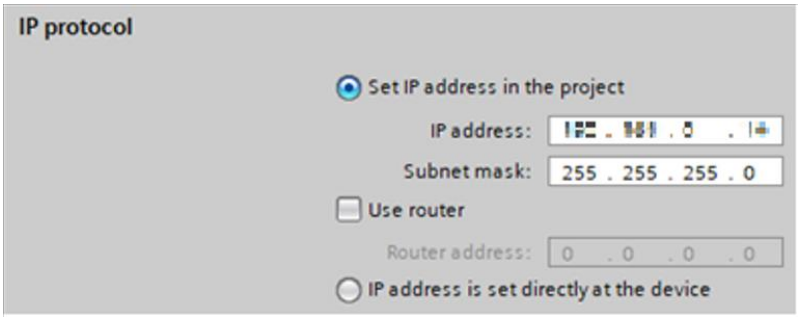


Figure 7 Assign IP address

Because all four partner PLC: s will not necessarily be connected to the converting PLC at all times the connections have to be established pre-hand under the connections tab on the converter PLC. The right number of connections are added, appropriate names and a unique local ID is given. Local endpoint where setup as the converting PLC, the partner configurations will be explained in the next paragraph. (Figure 8)

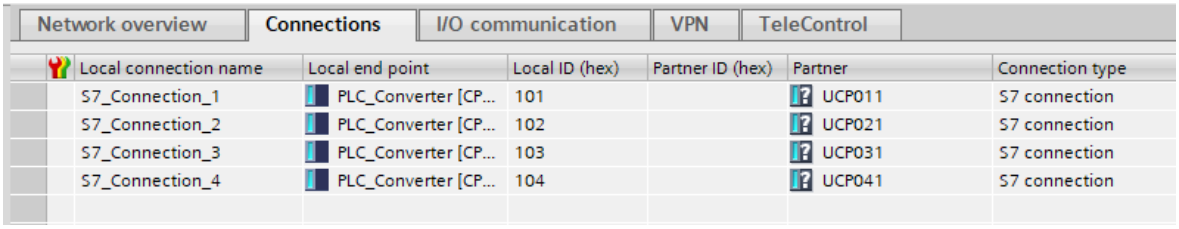


Figure 8 Partner PLC connections

The partner PLC is configurated by the general tab, here the partner PLC endpoint was given an engine specific name and an engine specific IP address, which is project and engine specific but must be in the same IP space as the converter and the rest of the system. The subnet is connected when the connection is established. (Figure 9)

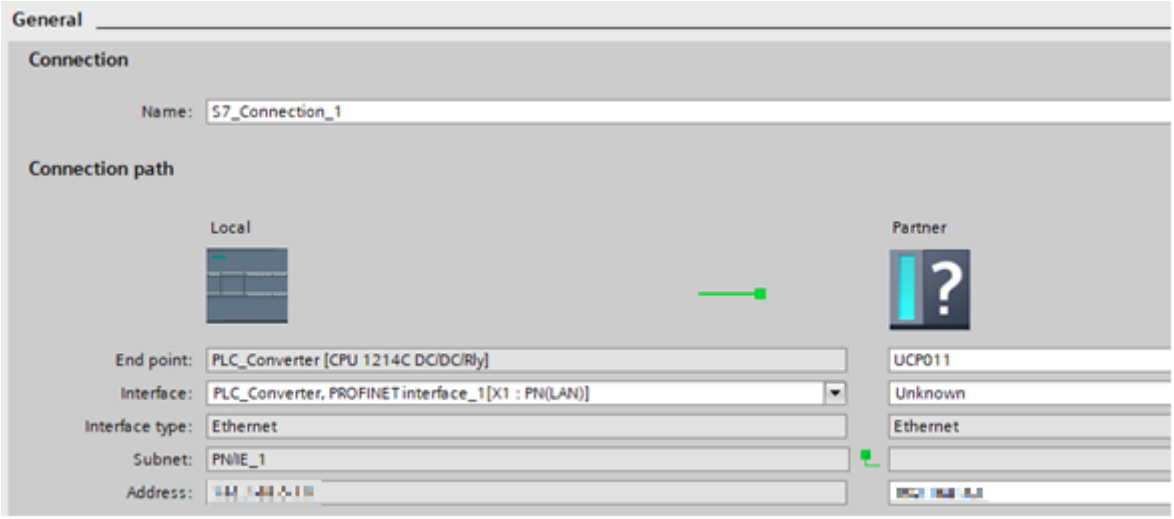


Figure 9 Connection configuration 1/2

The second thing to be configured on the partner PLC where the rack and slot number. The default numbers for a PLC CPU are rack 0, slot 2 (Figure 10). The needed configurations and connections are now done and the making of the function can begin.

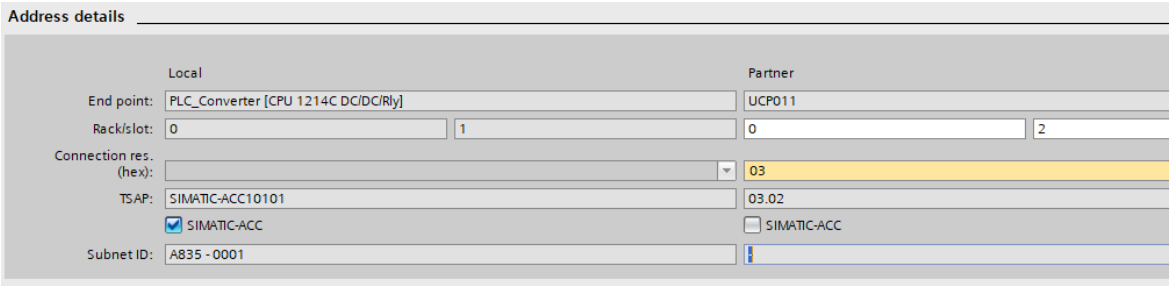


Figure 10 Connection configuration 2/2

6.2.3 Retrieving data (service project)

Classified

6.2.4 Retrieving data (new build)

Classified

6.2.5 Restoring INT to DINT

Classified

6.2.6 Converting & sending data

Classified

6.2.7 Main, FB1

As earlier stated, all the functions are placed in one function block with belonging instance DB. This means the whole conversion from start to finish can be implemented as a single block in any PLC software made in TIA portal V16. The main block (OB1) is set to cyclic sweep and therefore calls the function block “Communication_WDCU” every sweep. The function block in OB 1 works as a “user interface” for the user where UCP: s can be enabled or disabled and a unique UCP ID is set. Here are also, as earlier stated, the high word and low word inputs for the DINT: s. (Figure 11)

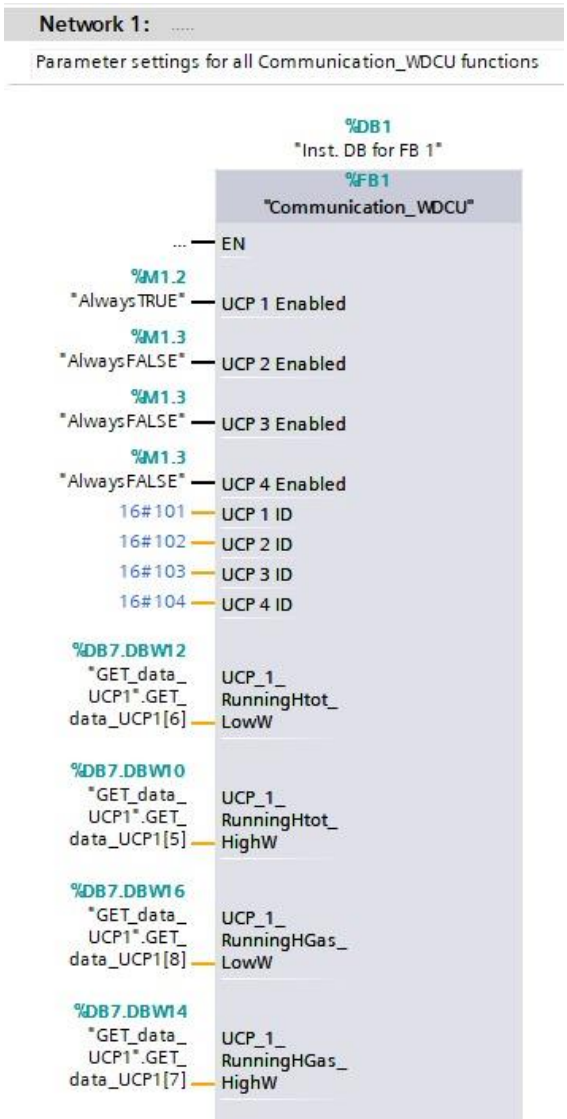


Figure 11 FB 1 in Main OB 1

When using the block in a PLC software it is recommended to also copy the "GET_data_UCP1-4" global databases to ensure they stay compatible with the block "Communication_WDCU". (Figure 12)

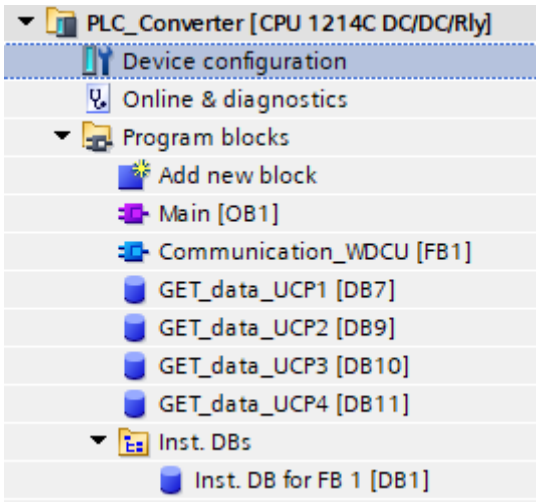


Figure 12 Converting program

6.3 Development of Excel macros for generating communication lists

The conversion from Siemens S7 protocol to Modbus is executed and standardized in Microsoft excel using VBA-based macros. The macro-based Excel program is a tool where the user chooses parameters on the cover page according to the project-specific needs and loads the WOIS database. The macros generate engine specific communication lists in Modbus according to both the conversion standards and the standards of the WDCU communication list structure.

The excel macro is designed for users who are somewhat familiar with the WOIS communication lists. The cover page of the macro works as an interface between the user and the converting macro, here the user can operate all functions and insert parameters. The macro informs the user of the state of the program or errors through message boxes.

Because the macro is made for users familiar with the subject nothing is locked, and this is so the user can modify the lists or use only parts of them.

6.3.1 Conversion syntax

Classified

6.3.2 User manual

Due to the file being macro-based, Excel will ask if you allow macros upon opening the XLSM file, this must be accepted otherwise the macro will not work.

With that done the user is presented with the first page of the Excel file, being the cover page (Figure 13). The box to the left is the user interface divided into different steps. The box to the right is instructions of the macro and important notes.

The Excel file contains all needed sheets which are displayed in Figure 14. “DB” is where the loaded DB will be stored, “IODisc”, “IOInt” and “IOReal” are the sorted analog and digital data tags, and “MB G1-4” are the converted engine specific Modbus lists ready for exporting.

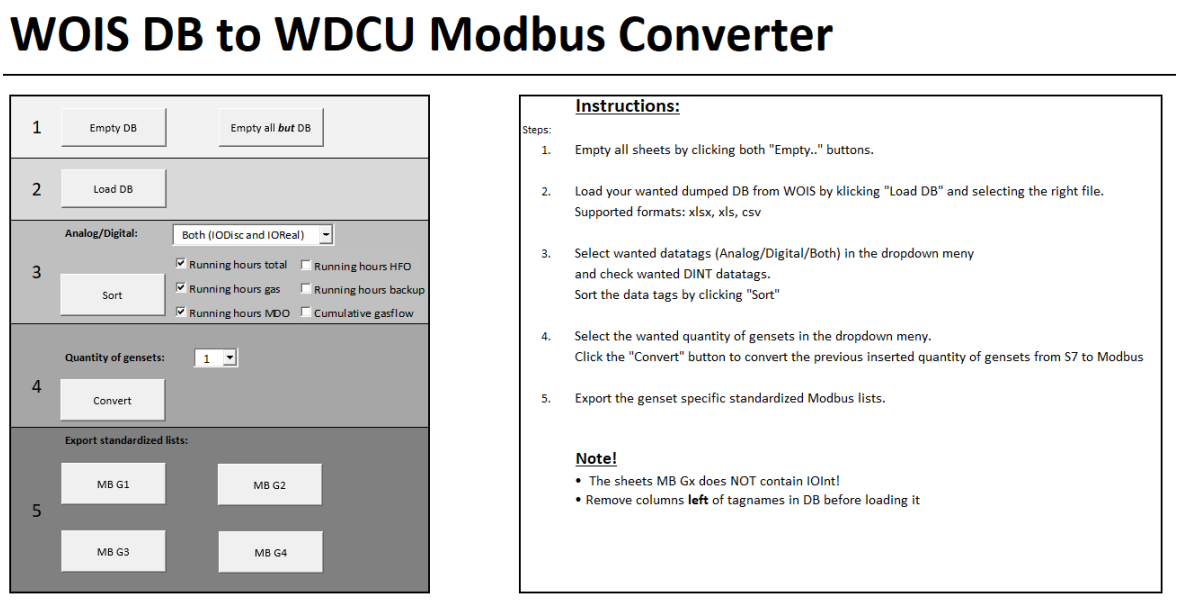


Figure 13 Conversion macro cover page

Figure 14 Converter macro tabs

To use the macro the user simply has to follow each step of the instruction:

1. To make sure all sheets are empty the user is prompted to empty all sheets by clicking both the buttons of step 1. Sheets containing old data may interfere with the conversion.

2. Load the WOIS DB in its original format without excessive columns by clicking the button of step 2. This will open the file explorer and ask the user to select the wanted file (Figure 15). Supported file formats are CSV, XLS, and XLSX.

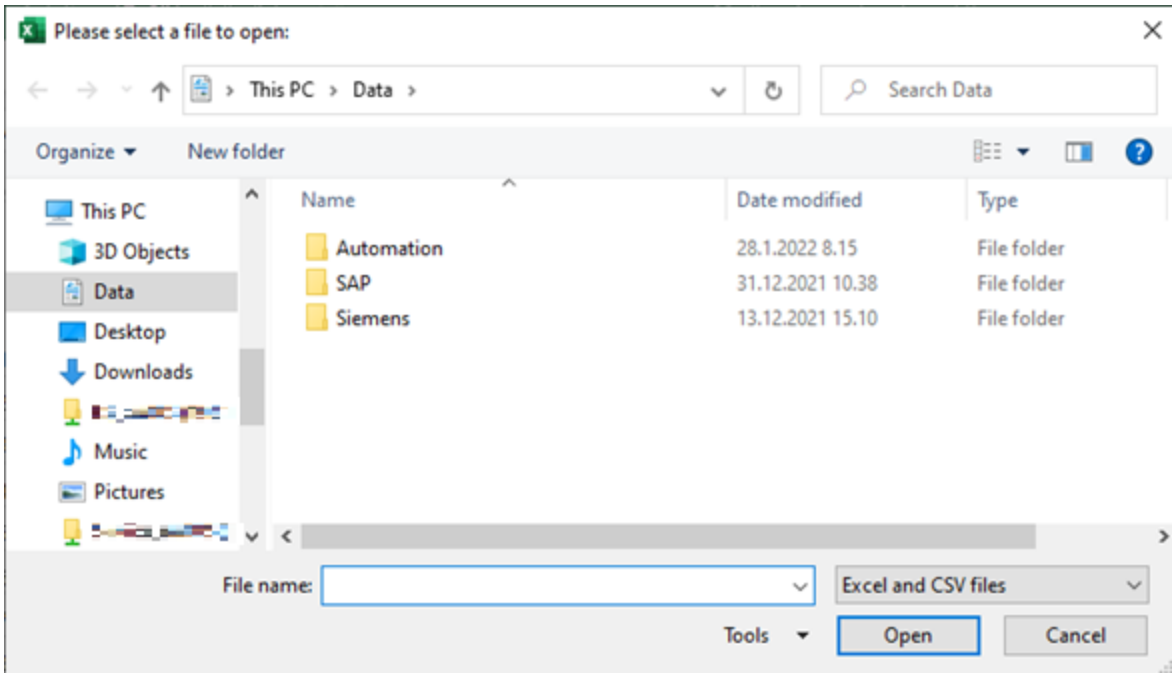


Figure 15 File explorer, load DB

3. Select the wanted data tags to be sorted from the DB by choosing an alternative in the dropdown menu (Figure 16) and check the checkboxes of wanted DINT tags, when this is done, click the button of step 3. The dropdown menu contains four options which are "Digital" for digital signals, "Analog" for analog signals, "Both" for both earlier stated, and "IOInt" which contains integers but is not needed for the final Modbus list. When Excel informs "Sort is done!" the earlier chosen tags are sorted and found in the respective sheet.

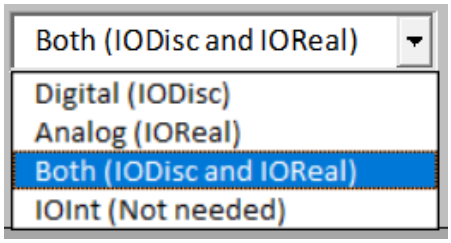


Figure 16 Sort selection

4. Select the wanted quantity of engines to be converted from the sorted lists by choosing an alternative from the dropdown menu (Figure 17) and clicking the button of step 4. The dropdown contains options from 1 to 4 engines. The number of engines chosen in step 4 cannot be greater than the number of engines in the WOIS application from where the DB

was taken. When excel informs “Conversion of engines done!” the converted communication lists can be found in sheets MB Gx, where x is the number of the engine. The list is sorted as digital signals first and then analog signals.

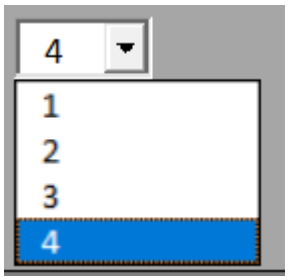


Figure 17 Engine quantity selection

5. When the communication list is sorted and converted the engine specific Modbus lists can be saved to a new separate Excel workbook. In step 5 there is one button for each engine, when clicking the button, the macro will open file explorer (Figure 18) asking where to save the file, file name, and extension. When the user has successfully saved the file Excel will provide the path to the saved location through a message box. The final step is not mandatory due to the possibility of manually copying the converted Modbus list. The generated Modbus list is now done can now be loaded into the WDCU.

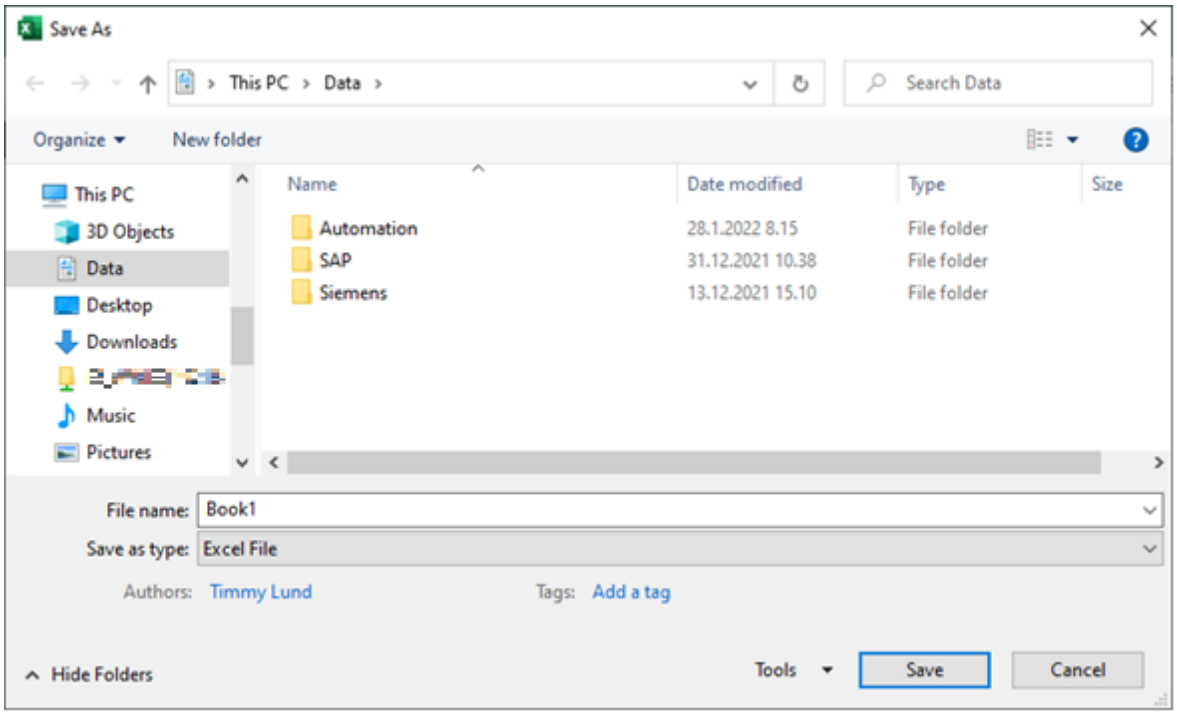


Figure 18 File explorer, export communication list

6.3.3 Macro

The functions were organized in modules according to the functionality of the steps on the cover page. The structure of the macro is graphically represented in Figure 19, where the different modules are listed in the top part with their belonging procedures, and the different sheets are in the bottom part in yellow, structured as a flowchart from left to right.

When the user loads the WOIS DB, it is done using the module “Import_export”, procedure “importDB()” which loads the DB to the sheet DB. From here the module “Sort” sorts the needed tags from the sheet DB to the respective sheet of “IODisc”, “IOInt”, and “IOReal”, still in S7 format. Next the conversion from S7 to Modbus is done using the module “Convert” with its procedures “convert_g1-4()”, this converts the chosen engines and places them in the correct Modbus list. The final module, which is the same as the first, exports the Modbus lists using the procedure “ExportG1-4()”.

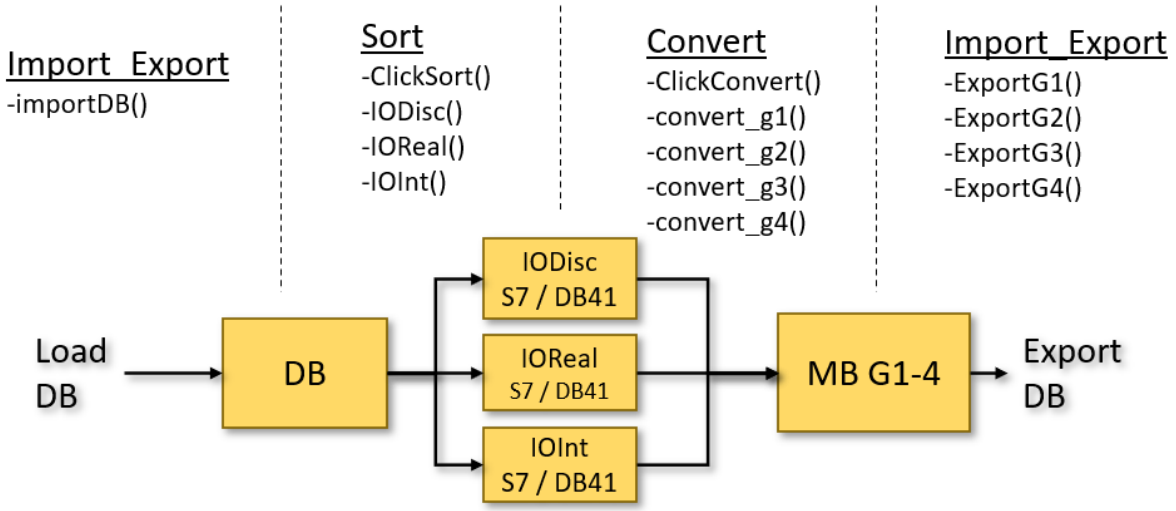


Figure 19 Flowchart, the structure of converting macro

For all functions the VBA function “Application.ScreenUpdate” was used to paus screen updating, this is due to a shortage of resources and flickering of the screen while the macro runs.

In the following chapters the individual functions will be explained in more detail. [7]

6.3.4 Empty sheet’s function

Classified

6.3.5 Load DB Function

Classified

6.3.6 Sort function

Classified

6.3.7 Convert function

Classified

6.3.8 Export function

Classified

6.4 Implementation and testing

With the converting macro and the PLC software developed the next step was testing, first if they work as intended individually and later tested together. The implementation and testing were done in ABB's testing area in Strömberg Park, Vaasa, with some help and supervision of experts in the subject. Validation of the PLC software was done through calculations on excel and validation of the macro was done using the conversion software on the PLC.

6.4.1 Implementation and testing of the converter PLC

In the following chapter IP addresses are classified and therefore exchanged for X and Y.

The testing of the PLC software was done in two main steps, the first was compiling the program to the converting PLC: s CPU to make sure the syntax of the software was correct and the second step was setting up the real environment including another PLC, acting as UCP and a device to read Modbus, acting as the WDCU.

Figure 20 is a graphical presentation of the environment of the second test. Device 1 is a regular computer running TIA portal V16 in monitoring mode, with a project linked to both

the UCP and the converter PLC, simultaneously running a Modbus scanner reading Modbus communication on port 501, representing engine 1. The computer IP address is set to the same IP space as the PLC: s, being XXX.XXX.XXX.YYY subnet 255.255.255.0. From the computer, on the UCP project values of the DB41 are set using a watchlist.

Device 2 is the UCP PLC, in this case, a Siemens S7 300 containing only DB41, this to make adjustments in the DB values as easy as possible. The UCP PLC represents engine 1 and therefore the IP address is set to XXX.XXX.XXX.YYY subnet 255.255.255.0 where the Y represents the engine number.

Device 3 is a simple unmanaged switch which must be used due to the lack of ports on either of the other device. Because the device is unmanaged it simply distributes the data between the ports.

The final device, device 4 is the converting PLC, in this case, a Siemens S7 1200 containing the converting software. Device 4 is set to IP address XXX.XXX.XXX.YYY subnet 255.255.255.0. The GET function in the converting PLC retrieves the data from DB41 in the UCP PLC in S7 format, converts it, and broadcasts it to Modbus which is read by the computer on Modbus port 501.

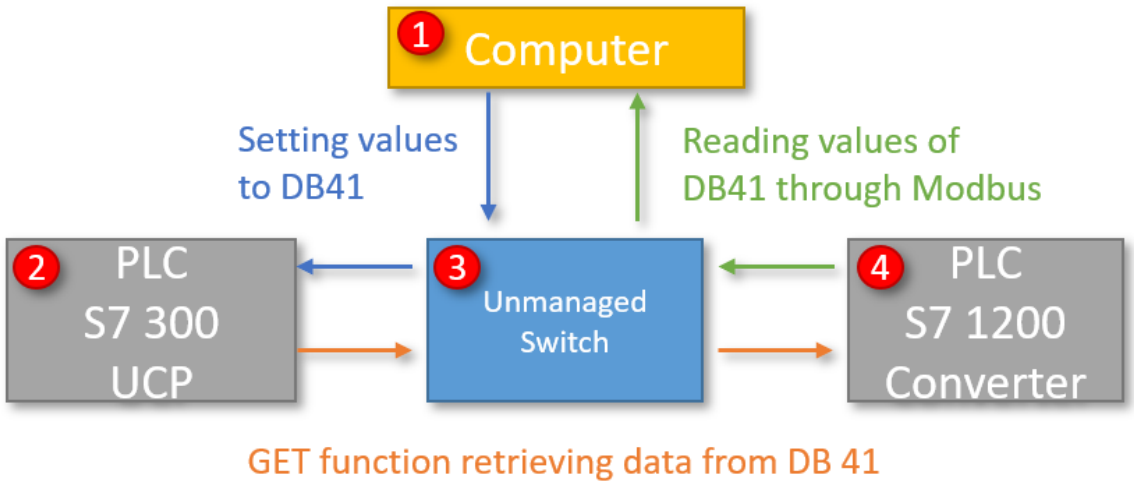


Figure 20 Flow chart, test setup

The address and value of the input S7 data are compared to the read data from the Modbus scanner. To make sure this is correct the calculations explained in chapter “6.3.1 Conversion syntax” were done.

The setup represented in Figure 20 was also re-configured as engines 2,3 and 4 to validate the functionality of all 4 possible engines.

6.4.2 Testing of the Excel macro

The testing of the Excel macro was done in the same order that the sorting and converting is done, because if the sorting fails, the conversion also fails. The first step was to make sure the sorting function worked as intended, to test this the tags from the specific header in the DB were copied and so were the sorted tags and both pasted into a temporary sheet. Conditional formatting for duplicated cells was used to see if all the needed tags had indeed been sorted from the DB. This was done for all the sorting functions. Details like sorting in groups of engines, address number, and so on were inspected visually.

With the sorting functions validated the converting function was tested. The first goal was getting the syntax of the Modbus list right due to the WDCU demanding a certain structure of the sheet. For this, a model Modbus list was used as a comparison. When the structure was correct the conversion of the addresses was validated through the help of calculation following the same principle as explained in chapter “6.3.1 Conversion syntax” and with the help of the converting software on the PLC.

As a last step of validation, the Excel macro different combinations of lists and conversions were generated along with different DB: s were used ranging from the year 2006-2021, during the tests the previously explained tests of comparison were done. The converted Modbus lists were also uploaded to a pre-configured WDCU, with the lists successfully uploaded the syntax was inspected visually.

7 Results

The result of the thesis work is an overall success. The converter device converts S7 protocol to Modbus and the Excel macro converts and generates the communication list for the WDCU as intended.

To summarise the work and represent both the converter PLC and the converter macro in its real environment the following structure was made. In the example, an old project is used to get a real data base and its belonging UCP PLC program.

First, the dumped database from the existing WOIS application, including all the data tags, is loaded to the converting macro, and parameters are set. The macro generates the engine specific communication list, which is loaded into the pre-configured WDCU as a data source, this is needed to bind the incoming Modbus traffic to the WDCU to the right data tag to be displayed. The WDCU is given a suitable IP address and each data source is given the converter PLC IP address and a unique port number, for example, engine 1 is given 501, engine 2 is given 502, and so on.

The second step is to configure the converter PLC. The device is given a suitable IP address and the S7 connections are assigned the engine specific UCP PLC IP addresses and ports. The GET blocks are dimensioned according to the "as built" information, this is crucial due to the size of the retrieved data array must be of the same size as the partner PLC location where the data is to be retrieved from.

The configurations of the converter and WDCU are now done, left is setting up the environment. The structure is according to Figure 21.

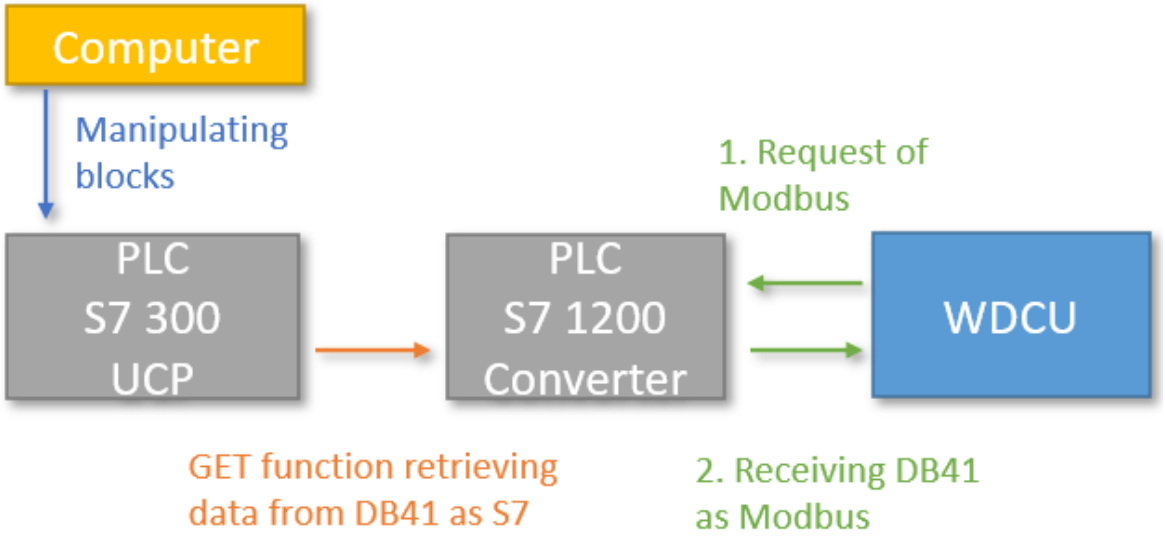


Figure 21 Example structure

To trigger events and alarms a computer is connected to the UCP PLC. To simulate analog values the input tag is removed, and the raw value is fed to the block. In Figure 22 the raw input, ranging from 0 to 27648, is manually set to 10000 on input IN, this results in an output of 1085 which is stored on word 50 in DB41 of the UCP PLC.

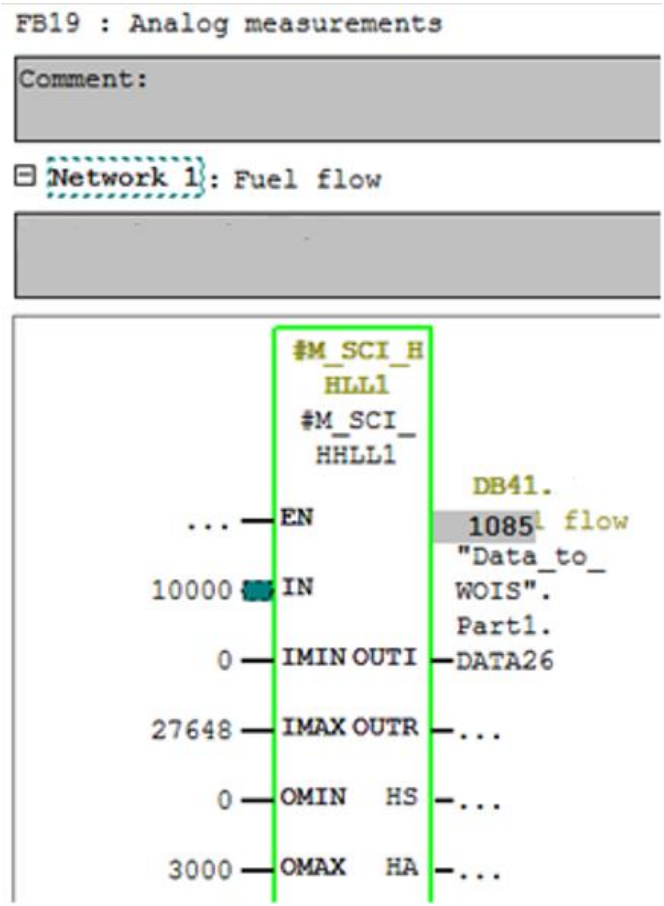


Figure 22 Manipulation of analog values

To prove that the manipulation of an input on the UCP PLC reaches the actual monitoring system, the same tag is displayed in the WDCU trend window. (Figure 23 and Figure 24)

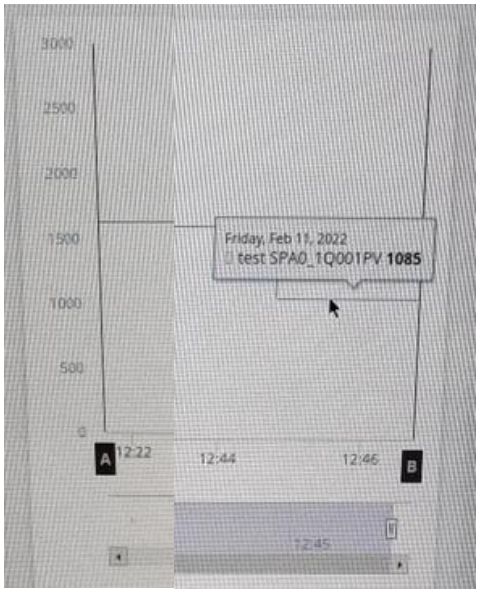


Figure 23 Display of analog values, trend

MEASUREMENT	UNIT	FREQUENCY	LATEST
Fuel flow	l/h	Live	1085

Figure 24 Display of analog values

The same is done to a digital signal. The input of signal “Engine start order from UCP” is inverted (Figure 25), triggering the event in the WDCU (Figure 26).

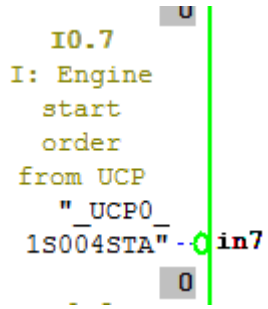
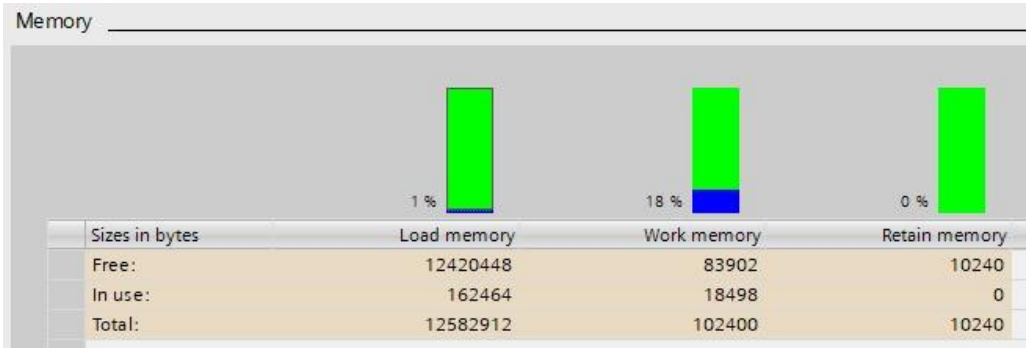


Figure 25 Manipulation of digital values

STATE	DATE	TIME	DESCRIPTION	EQUIPMENT	STANDARD TAG	FUNCTION
ON	2022-02-11	12:27:15	Engine start order from UCP	test	UCP0_1S004STA	STATUS

Figure 26 Display of digital values

The size of the PLC was based on estimations, and a bit oversized on purpose. With the finished program loaded and running the cycle time is approximately 4 milliseconds and the memory usage is according to Figure 27.



Sizes in bytes	Load memory	Work memory	Retain memory
Free:	12420448	83902	10240
In use:	162464	18498	0
Total:	12582912	102400	10240

Figure 27 Memory usage

8 Discussion / Conclusion

Despite the success of the finished project, there were some problems on the way. Originally the converter was supposed to handle up to 10 engines simultaneously but after some research, it would have demanded two S7-1200 CPU: s or one S7-1500 CPU. The price of these setups would have been unnecessarily big and therefore it was decided to only make the converter with one S7-1200 CPU as a proof of concept. With the S7-1200 CPU, 4 engines are the maximum but if upcoming projects demand more than that a second S7-1200 PLC can easily be added with the same software, but different ports and IP addresses.

The optimal way of determining the size of the PLC and the memory card would have been to use an in-house PLC for the developing stages of the project and with the PLC software done, checked the size and how demanding the software is and order a suitable PLC. This was not possible due to a major semiconductor shortage and overall problem to get hardware because of the COVID-19 pandemic, with a delivery time of 75 days or more. The solution was to estimate the needed size and order the device as soon as possible. A PLC better optimized according to size can be used in upcoming projects now that the program size and performance needs are known.

Further improvements to make commissioning and troubleshooting easier would be to broadcast the error and status messages of the GET blocks to Modbus and add the possible errors and statuses as default to the communication lists using the Excel macro. With this improvement, the errors of the GET blocks would be displayed in the WDCU which would make pinpointing the problem easier.

As conclusion the converting device and the converting macro is working as intended for the first prototype. As the converter is used in upcoming projects it will be improved according to the needs.

The collaboration with employees of ABB Energy Industries during the thesis work has been great, information and help concerning the project have been easy to get.

9 Bibliography

[1] ABB Oy, "ABB Oy, Energy Industries," [Online]. [Accessed January 2022].

[2] PLCdev, "PLC Basics," PLCdev, [Online]. Available: http://www.plcdev.com/book/export/html/9. [Accessed January 2022].

[3] V. Romanov, "Top 5 Most Popular Types of PLC Programming Languages," SolisPLC, 2020. [Online]. Available: https://www.solisplc.com/blog/plc-programming-languages. [Accessed January 2022].

[4] Siemens AG, "Your gateway to automation in the Digital Enterprise," 2018. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:0c66eeeb-b67d-4c98-aa0d-290f82d5c0d2/version:1575896550/7801-09-tia-p-ipdf-en-181029-1.pdf. [Accessed January 2022].

[5] Siemens AG, "SIMATIC S7-1200 Easy Book," Januray 2015. [Online]. Available: https://euroec.by/assets/files/siemens/s71200_easy_book_en-US_en-US.pdf. [Accessed January 2022].

[6] Siemens AG, "SIMATIC S7-1200 Getting started with S7-1200," November 2009. [Online]. Available: https://support.industry.siemens.com/cs/attachments/39644875/s71200_getting_started_en-US_en-US.pdf?download=true. [Accessed January 2022].

[7] Microsoft, "Functions (Visual Basic for Applications)," Microsoft, 13 09 2021. [Online]. Available: https://docs.microsoft.com/en-us/office/vba/language/reference/functions-visual-basic-for-applications. [Accessed January 2022].