

RTU 560 with IEC 61850

Vaasa Engineering Oy

Joakim Ainasoja

Bachelor's thesis
Electrical Engineering
Vaasa 2010





BACHELOR'S THESIS

Author: Joakim Ainasoja
Degree programme: Electrical Engineering
Specialization: Power Engineering
Supervisor: Erik Englund

Title: *RTU 560 with IEC 61850*

Date: 1.12.2010 Number of pages: 44 Appendices: 2

Abstract

This thesis work was made in cooperation with Vaasa Engineering Oy. The thesis deals with the procedure for a configuration of an ABB RTU 560G for communication with protective relays from ABB and VAMP to a network control system in the form of MicroSCADA. The goal of this work is to provide a method for configuring RTU 560 so that it serves as a central unit between the protocols IEC 61850 and IEC 60870-5-101. The approach is explained together with descriptions and solutions to problems that arose during the configuration process. Firstly background and theory are explained and then the methods used during configuration. The final result and the conclusion are also described and discussed.

Language: english Key words: IEC 61850, RTU 560, substation automation

Filed at the electronic library Theseus.fi or at the Tritonia Academic Library, Vaasa.

EXAMENSARBETE

Författare: Joakim Ainasoja
Utbildningsprogram och ort: Elektroteknik Vasa
Inriktningsalternativ/Fördjupning: Elkraftsteknik
Handledare: Erik Englund

Titel: *RTU 560 med IEC 61850*

Datum: 1.12.2010

Sidantal: 44

Bilagor: 2

Abstrakt

Detta examensarbete har gjorts i samarbete med Vaasa Engineering Oy. Examensarbetet behandlar förfarandet vid en konfiguration av en ABB RTU 560G för kommunikation med skyddsreläer från ABB och VAMP, fram till ett övervakningssystem i form av MicroSCADA. Målet med arbetet var att ge en metod till konfiguration av RTU 560 så att den fungerar som en centralenhet mellan protokollen IEC 61850 över till IEC 60870-5-101. Tillvägagångssättet förklaras tillsammans med beskrivningar och lösningar på problem, som uppstod under konfigureringsprocessen. Inledningsvis tas bakgrund och teori fram för att sedan övergå till metoderna använda vid konfigurationen. Avslutningsvis beskrivs resultatet och slutsatsen diskuteras.

Språk: engelska Nyckelord: IEC 61850, RTU 560, understations automation

Förvaras på webbiblioteket Theseus.fi eller vid biblioteket Tritonia, Vasa.

OPINNÄYTETYÖ

Tekijä: Joakim Ainasoja
Koulutusohjelma ja paikkakunta: Sähkötekniikka Vaasa
Suuntautumisvaihtoehto/Syventävät opinnot: Voimatekniikka
Ohjaaja: Erik Englund

Nimike: *RTU 560 IEC 61850-protokollalla*

Päivämäärä: 1.12.2010 Sivumäärä: 44

Liitteet: 2

Tiivistelmä

Tämä opinnäytetyö tehtiin yhteistyössä Vaasa Engineering Oy:n kanssa. Opinnäytetyö käsittelee menettelyä ABB RTU 560G:n konfiguraatiosta yhteydenpitoon ABB:n ja VAMPin soujareleistä valvomojärjestelmiin MicroSCADA:n muodossa. Työn tavoitteena on tarjota RTU 560:n konfigurointiin menetelmä siten että se toimii keskusyksikkönä IEC 61850:n ja IEC 60870-5-101:n välillä. Lähestymistapa on selitetty yhdessä kuvauksin ja ratkaisuin ongelmiin, jotka ilmestyivät määrittämisprosessin aikana. Aluksi tausta ja teoria on selitetty siirtyäkseen määrittämisprosessin aikana käytettyihin menetelmiin. Lopuksi kuvaillaan lopputulosta ja keskustellaan päätelmästä.

Kieli: englanti Avainsanat: IEC 61850, RTU 560, ela-asema automaatio

Arkistoidaan verkkokirjastossa Theseus.fi tai kirjastossa Tritonia, Vaasa.

Table of contents

Abstract

Abstrakt

Tiivistelmä

Abbreviations

1	Introduction.....	1
1.1	Background.....	2
1.2	Goal	2
1.3	Vaasa Engineering Oy	2
2	The electrical power system and its control.....	3
3	Communication in substation automation	5
3.1	Basics of data communication.....	5
3.2	LAN and WAN networks.....	6
3.3	The OSI model.....	7
3.4	Common protocols for data communication	8
3.4.1	TCP/IP.....	8
3.4.2	Modbus	9
3.4.3	DNP 3.0	9
3.4.4	SPA-bus.....	10
3.4.5	IEC 61850	10
4	Hardware	13
4.1	Remote Terminal Units	13
4.1.1	RTU models from different manufacturers	13
4.1.2	RTU 560.....	13
4.1.3	Netcon 500	14
4.1.4	Siemens AK 1703 ACP	15
4.2	Protective relays	16
4.2.1	VAMP	16
4.2.2	Relion	16
4.2.3	Siprotec.....	16
4.2.4	GE.....	17
4.3	Control center software	17
4.3.1	ABB MicroSCADA.....	17
5	Configuration process.....	18
5.1	REF 541 and CAP 505	20
5.2	CET and SPA-ZC configuration.....	20

5.2.1	Problems occurring with CET.....	21
5.3	Vampset	22
5.3.1	Problems with VAMP	22
5.4	RTUtil	23
5.4.1	Problems with RTUtil.....	24
5.5	PCM 600 configuration.....	25
5.6	CCT/IET.....	26
5.6.1	Problems with CCT.....	28
5.7	Back to RTUtil	28
5.8	Multiprog WT.....	30
5.9	HMI Editor.....	32
5.10	MicroSCADA.....	35
5.10.1	Problems with MicroSCADA	40
5.11	Results	41
6	Discussion and conclusion.....	42
7	List of sources.....	44

Appendices

RTU 560 configuration guide

I/O list

Abbreviations

ASCII	American Standard Code for Information Interchange
BRCB	Buffered Report Control Block
CAP 505	Computer Aided Programming 505
CCT	Communication Configuration Tool
CET	Communication Engineering Tool
DCS	Distributed Control System
DHCP	Dynamic Host Configuration Protocol
DNP 3.0	Distributed Network Protocol 3.0
EPRI	Electric Power Research Institute
GOOSE	Generic Object Oriented Substation Event
GSE	Generic Substation Event
GSSE	Generic Substation State Event
I/O	Input/Output
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IEEE	Institute of Electrical and Electronics Engineers
IET	Integrated Engineering Tool
ISO	International Organization for Standardization
ISP	Internet Service Provider
L/R	Local /Remote
LAN	Local Area Network
MAC address	Machine Address Code
MMS	Manufacturing Message Specification
NO/NC	Normally Open/Normally Closed
OSI	Open Systems Interconnection
PC	Personal Computer
PCM 600	Protection and Control IED Manager
PLC	Programmable Logic Controller
RCB	Report Control Block
RTU	Remote Terminal Unit
SCADA	Supervisory Control And Data Acquisition
SNTP	Simple Network Time Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
UCA	Utility Communications Architecture
URCB	Unbuffered Report Control Block
VEO	Vaasa Engineering Oy
VMD	Virtual Manufacturing Device
WAN	Wide Area Network
WLAN	Wireless Local Area Network

1 Introduction

What if there were no standards published yet to this day. People would roam around while producing unique ideas which would lead to a very wide array of individual products that could not merge with each other in any way since they are all unique. Imagine owning an elegant phone with a lot of features. Without standards this phone would be rendered useless since no other phone would use the same features. Standards make merging and coexistence of products a possibility as they tell us how things should be made. The best standards are developed by selecting the most efficient, simplest and easiest ways for doing things. Standards make everyday life a lot easier.

Languages have no international standards saying that everyone has to speak English just to make conversations abroad easier, but it would be a lot simpler if they had. Dictionaries and grammar tell people what words to use and how to inflect them within a certain language. This example can also be used when referring to communication protocols used by intelligent devices within a certain process or station. The protocols are standards that can be seen as languages spoken by the devices, and the protocols should match perfectly on both sides for the communication to work.

Steady research and development concerning electronic devices give them new and improved functions and give new demands for communication between devices. This makes it hard for a protocol to stay up-to-date with the demand for features and functions made by users. The average lifetime of a protocol is 10 years, depending on the type of industry it is used in. The automotive industry has a fast change in demands whereas the process industry is more conservative and thus stretches the lifetime of a protocol.

A relatively new protocol in digital communication is the IEC 61850, which is based on an international standard and meant to simplify the installation and engineering of electrical substation automation. The protocol helps engineers utilize the latest systems for connection like optical fiber and Ethernet cables.

1.1 Background

This thesis work was assigned to me by Vaasa Engineering Oy. My work began in the beginning of February 2010. The popularity of the IEC 61850 protocol is growing in the Finnish electrical substation market but the implementation of the protocol is not yet perfect, meaning that errors and flaws may occur when establishing communication between devices from various vendors.

1.2 Goal

The goal with my work is to test the RTU 560G connected to one VAMP- and one ABB protection device and test their communication according to IEC 61850 and IEC 60870-5-101 with the 560G as gateway. I will document my procedures for establishing the connections and when testing basic but vital functions like alarms, clock synchronization etc. All errors and problems encountered will also be documented and if they are resolved I will give step by step procedures on how to avoid these problems. I will also give basic information on previous popular protocols and some information on devices mostly used by Vaasa Engineering Oy for substation automation. This thesis work will give me the basic understanding needed for working with substation automation and the belonging protocols.

1.3 Vaasa Engineering Oy

Vaasa Engineering Oy provides automation and electrification solutions for energy production, transmission, distribution and use to customers worldwide. They deliver turnkey projects and project components, design and engineering, procurement and supply, project management, installation, start-up and commissioning and user training. Plant modernizations, maintenance, system updates and switchgears are also offered. The cooperation with leading manufacturers throughout the field enables Vaasa Engineering to remain at the leading edge of development. Vaasa Engineering Oy is often referred to as VEO. (9)

2 The electrical power system and its control

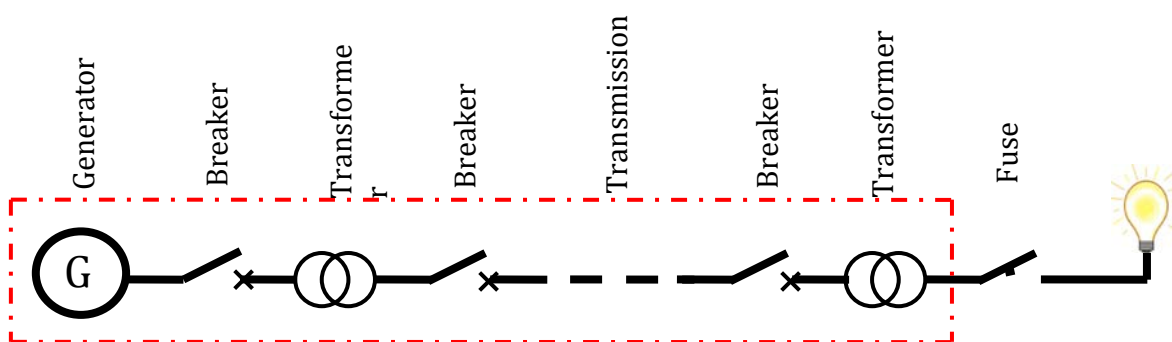


Figure 1. Basic generation and distribution of electricity. Components inside the red rectangle are in need of intelligent protection devices.

The basic idea of electrical distribution is to find a power source such as light, wind, water or diesel engines. These powers are then harnessed so that they turn the axle of a generator which in return generates electricity. Power lines are then connected to this generator so that electricity can be transferred to the end user. Breakers and fuses are used to protect the electrical equipment in case faults or error occur while transformers are used for increasing (step-up) or decreasing (step-down) the voltage levels.

Substations are secondary stations where the feeding lines are connected to breakers, disconnectors, transformers and busbars so that electricity can be controlled and divided into distribution lines.

To be able to provide continuous electrical power to customers, power plants and substations need to be equipped with IEDs such as protection relays for supervision and control of the bigger components. These protection devices are a lot smaller than the breakers that switch the loads, but as the big breakers are motorized the smaller IEDs are able to control them. The IEDs read measurements and state indications from the larger components and then act in accordance with the configuration that they have been given. These measurements and state indications should then be passed on to the network control centers so that humans can be informed on the substation condition. This is where the data communication functions as a means to transfer the information and receive commands both locally within the substation and onwards to control centers.

Before computer aid in process control was developed and affordable, many control rooms consisted of a control table hardwired to all the adjustable devices in the process and a large screen from where alarms and conditions were readable and mostly all different commands had to be sent through its own wire making cabling very expensive.

In recent years, process control has taken a big step forward thanks to computer aid. The digital age has removed much of the hardwiring previously necessary for sending control signals (in some cases even all communication wires thanks to wireless technology). Digital communication gives users the ability to address these commands and send all of them through one cable, but it requires hardware and software that are able to recognize these addresses and commands so that they can be applied to the correct device. Protocols give hardware and software these necessary capabilities to understand the data that is being sent and received. The new international standards also give devices from different vendors the ability to cooperate without needing special modifications.

IEDs and their advanced and extensive functionality have given the ability to create more reliable stations as more functions can be added without a huge expense to customers. Many of these devices also support PLC logics so that special requirements and functions that are station-specific can easily be added.

SCADA systems are used for supervisory systems for control with humans as operators. The primary control is not done by SCADA but by intelligent devices and surrounding control systems. These devices collect process specific data and later on convey this data to executive control equipment for supervision. (3)

3 Communication in substation automation

3.1 Basics of data communication

People have learnt how to regulate voltage quickly and efficiently, which is a crucial knowledge within data communication. George Boole is the person credited for coming up with the idea of having two different states defining yes or no / on or off / TRUE or FALSE (hence the name *BOOLEAN value*). Later on it was realized that Boole's idea could be implemented into computer logic. By regulating voltage levels over or below a predefined setting you can give a logic 1 or a logic 0, where 1 stands for TRUE and 0 stands for FALSE. By having a definite time interval you can change the voltage level for each time-interval and have the receiving end read the value once every interval. This voltage level at a specific time is called a *bit*, one bit can be 1 or 0. By sending these bits one after another you get something called a *bit array*, a series of ones and zeroes. After that a standard tells us what a certain amount and order of these values correspond to, for example the ASCII defines the printable letter A as 01000001, the symbol & is coded as 00100110. See figure 2 for an illustration of the symbol & as an electrical signal.

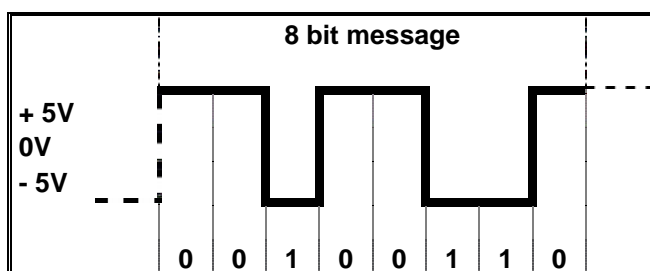


Figure 2. Illustration of the symbol “&” as an eight bit data message according to the ASCII code. Time is presented horizontally and voltage vertically.

By putting several of these bit arrays after each other it is possible to send e.g. text messages over electrical conductors (other mediums like optic fiber, radio waves etc. are also possible) using the ASCII standard. A few complementary bits need to be added to the message for functional data exchange, depending on which protocol is used for communication. The other bits can be for addressing, error detection (parity check), encryption etc.

3.2 LAN and WAN networks

LAN networks can be found today at homes, offices, industry etc. A LAN network is built within a small area that does not need an Internet connection for transmitting data between devices that are interconnected. Ethernet cable or wireless (WLAN) connection from the device to the router is needed to create a LAN. WAN is the complementary connection between multiple LANs or other networks, so that information can be obtained between users over longer distances. To get WAN access you often need an ISP that handles connections via telephone or similar physical links. LAN and WAN can be explained rather as a physical way to connect devices than a way of electronic communication. (10)

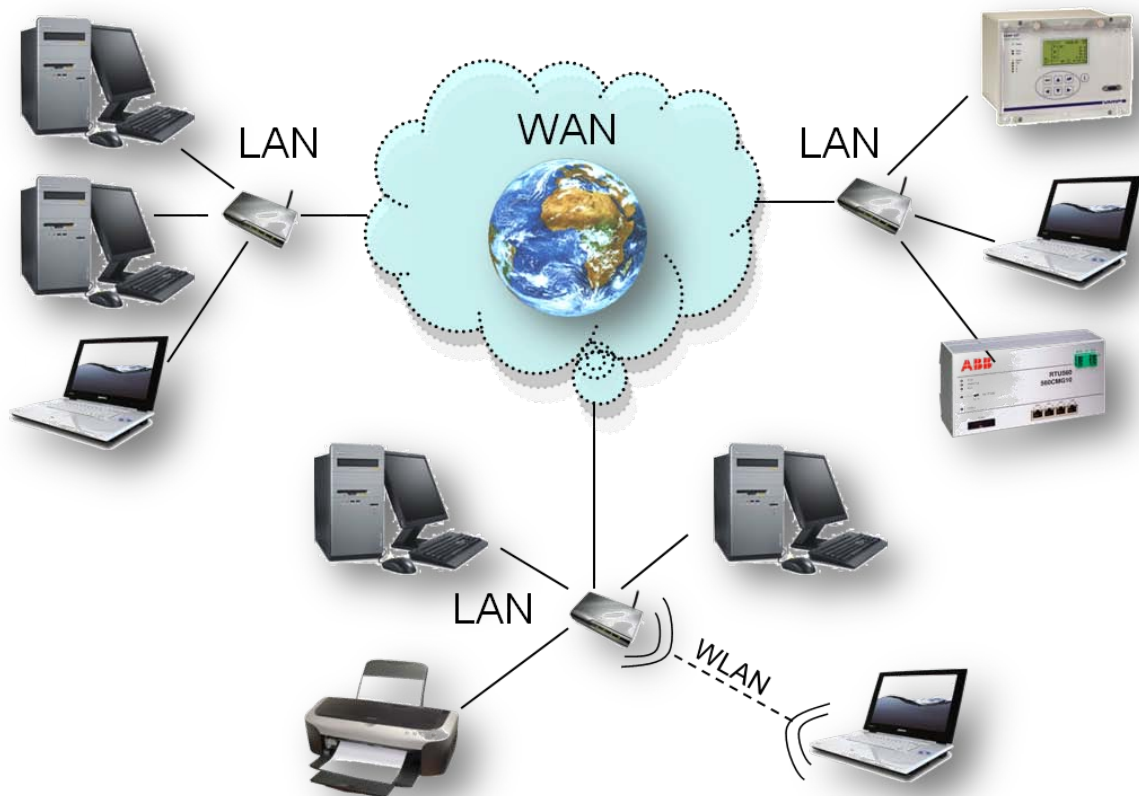


Figure 3. Example of interconnection with LAN and WAN.

3.3 The OSI model

Many protocols are built upon the seven layer OSI (Open Systems Interconnection) model which is a reference on how to build up the messages to be sent. The model works so that the information/data needed to be sent is first processed through the seven layers before being sent, with each layer adding or modifying the data within the message as for example receiver address or encryption. Once the message has been received the process is reversed and the data retakes its original form so that the end application can present it for the user, while some protocols enable an entity in one host to interact with a corresponding entity at the same layer in another host. The OSI protocol hierarchy is illustrated in figure 4. (10)

Layer	Function
#7, Application	Handles data produced in an application, e.g. E-mail.
#6, Presentation	Performs data transformation, including services such as reformatting, data compression and encryption.
#5, Session	Establishes, maintains and terminates user connections. Also handles grouping of data.
#4, Transport	Ensures accurate delivery of data through flow control, segmentation and reassembly, error correction and acknowledgement
#3, Network	Establishes network connections, translates network addresses into their physical counterparts and determines routing
#2, Data link	Packages data in frames appropriate to network transmission method and ensures the reliability of the physical link established at layer 1.
#1, Physical	Controls transmission of the raw bit stream over the transmission medium. Determines amount of voltage swing, duration of voltages (bits), and so on.

Figure 4. The 7 layer OSI model.

3.4 Common protocols for data communication

Below some of the types of protocols used today are described briefly. All countless protocols available are not mentioned, only the ones that are commonly used and that are not manufacturer-specific. It is also hard to compare the protocols side by side, therefore only some of their individual features are brought up. The information given here is only scratching the surface by brief explanations for these standards. They are much more complex when analyzing what is going on at the bit-level.

- TCP/IP
- Modbus
- DNP 3.0
- SPAbus
- IEC 61850
 - GSE
 - MMS

3.4.1 TCP/IP

TCP/IP is a protocol that takes care of addressing and error checking in data communication between two devices. Units may have a static IP address or they may be given a dynamic address from the router's DHCP protocol. Each network card inside a unit has a unique MAC address which makes it possible for data to be sent to the proper recipients. TCP/IP consists of four layers from the OSI model, - application, transport, Internet (same as network layer) and the link layer (physical). Every layer adds different bits for control, addresses headers etc. Shown in figure 5 is the actual data originally sent (marked with blue). Notice that the data is just a small part of the whole message.

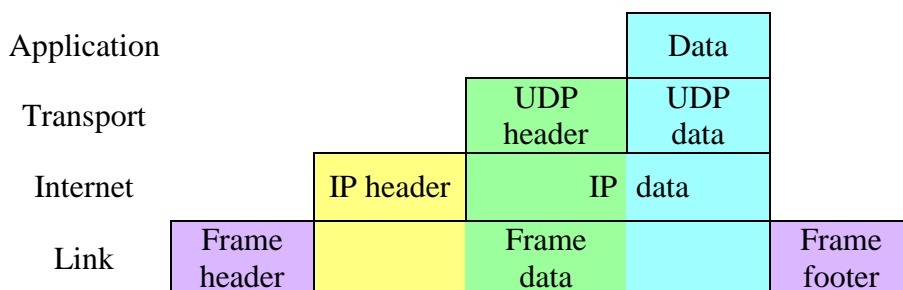


Figure 5. Protocol-overhead from an Internet stack.

The TCP/IP protocol has become so widely used and important that it can be viewed as a reference model. (10)

3.4.2 Modbus

Modbus was developed by Modicon ® in 1979 to be used for communications with PLC devices. It is an open protocol which means that developers can design products that make use of the Modbus standard, without limitations. It grew rapidly in popularity and is now one of the most widely used methods for connecting industrial electronic devices. In a standard Modbus system there is one *master* unit and there may be up to 247 *slave* units. The error control called *Cyclic Redundancy Check* is used, which means that after every message there are two additional bytes. These bytes are the result of a calculation (checksum) from all the bits in the message. In case these bytes are not summarized correctly, the receiving units know that there has been an error within the message.

Modbus and many other protocols make use of the serial ports (RS-232 and RS-485) as physical links. The serial communication port that has been around for almost 40 years was often found on most PCs before, but has nowadays mostly been replaced by USB ports. The common serial port may be viewed in figure 6. (8)

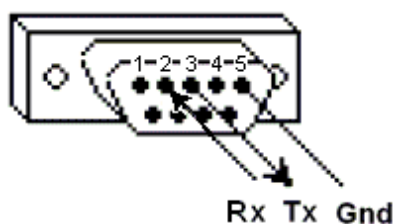


Figure 6. Serial port design (Rx – Receive, Tx -Transmit), and an example of digital signaling. (4)

3.4.3 DNP 3.0

The DNP 3.0 protocol is similar to Modbus but since it was developed in more modern times (early 90s) it has been given support for more functions. DNP 3.0 is an open protocol and that is a benefit to developers who wish to make modifications or extensions enabling better support with newer functions for use in SCADA systems. (4)

3.4.4 SPA-bus

The SPA-bus uses an asynchronous serial communications protocol (1 start bit, 7 data bits + even parity, 1 stop bit) with a common data transfer rate of 9600 b/s. Messages on the bus consist of ASCII characters, which are easier for a human to interpret than hex-code which is normally used by many other protocols. The bus can support one master and several slaves. (1)

3.4.5 IEC 61850

For numerous years the goal has been to define a communication infrastructure that will allow seamless integration of the IEDs into higher level devices - an infrastructure that is vendor independent and will allow devices from multiple vendors to be integrated together. That goal is what the IEC 61850 standard has been built upon.

A few years ago, DNP and IEC 60870-5 (also referred to as IEC 870-5) were the most widely supported of the traditional SCADA protocols, but UCA and IEC 61850 were meant to represent a new approach to utility communications. In 1994, EPRI/IEEE started working on the next phase of UCA, namely UCA 2.0. This time the focus was put on the Station Bus. In 1996, Technical Committee 57 of the IEC began to work on IEC 61850 with a similar charter defining a Station Bus. In 1997, the two groups agreed to work together to define a common international standard that would combine the work of both groups. IEC 61850 is a set of UCA 2.0, i.e. it contains almost all of the UCA 2.0 specifications, plus offers additional features. The results of the harmonization efforts are the current IEC 61860 specification which wants to define three things:

1. Which data is available and how is this named and described (IEC 61850-7-4, -7-3, and -7-2),
2. How can this data be accessed and exchanged (IEC 61850-7-2), and
3. How can devices be connected to communication networks (IEC 61850-8-x and -9-x).

Some of the features included in IEC 61850 are:

1. *Data Modeling* - Primary process objects as well as protection and control functionality in the substation are modeled into different standard logical nodes, which can be grouped under different logical devices. There are logical nodes for data/functions related to the logical device (*LLNO*) and physical device (*LPHD*).
2. *Reporting Schemes* - There are various reporting schemes (*BRCB* & *URCB*) for reporting data from server through a server-client relationship which can be triggered based on pre-defined trigger conditions.
3. *Fast Transfer of events* - GSE are defined for fast transfer of data for a peer-to-peer communication mode. This is again divided into *GOOSE* & *GSSE*.
4. *Setting Groups* - The setting group control Blocks (*SGCB*) are defined to handle the setting groups so that the user can switch to any active group.
5. *Sampled Data Transfer* - Schemes are also defined to handle transfer of sampled values using Sampled Value Control blocks (*SVCB*).
6. *Commands* - Various command types are also supported by IEC 61850.
7. *Data Storage* - SCL (Substation Configuration Language) is defined for complete storage of configured data of the substation in a specific format.

The abstract data models defined in IEC 61850 can be mapped to a number of protocols. Current mappings in the standard are to MMS, GOOSE, SMV, and to Web Services. These protocols can run over TCP/IP networks and substation LANs using high speed switched Ethernet to obtain the necessary response times of < 4 ms for protective relaying.

The GSE control model is further subdivided into GOOSE and GSSE. GOOSE is a control model mechanism in which any format of data (status, value) is grouped into a data set and transmitted within a time period of 4 milliseconds. The following mechanisms are used to assure specified transmission speed and reliability. GOOSE messaging is mostly used for horizontal communication, e.g. breaker state indications between protection relays for interlocking purposes.

GSSE is an extension of event transfer mechanism in UCA2.0. Only Status data can be exchanged through GSSE and it uses a status list (string of bits) rather than dataset used in GOOSE. GSSE message is transmitted over MMS based stack (base stack without using TCP/IP), which necessitates more time for transmission & processing in comparison with GOOSE messages.

MMS is an international standard (ISO 9506) dealing with messaging system for transferring real time process data and supervisory control information between networked devices and/or computer applications. MMS defines the following:

1. A set of standard objects that must exist in every device and on which operations like read, write, event signaling etc can be executed. VMD is the main object and all other objects like variables, domains, journals, files etc come under VMD.
2. A set of standard messages exchanged between a client and a server stations for the purpose of monitoring and/or controlling these objects.
3. A set of encoding rules for mapping these messages to bits and bytes when transmitted.

The protocol is based on LAN technology where the nodes are linked through a switch. The main advantage of the protocol is that it makes fast and reliable data transmissions for events and commands within a substation. One goal is also that all devices supporting this protocol will be “plug and play” compatible, which will save the end user a lot of time. Devices may also act as master or as slave, this is meant as a model of communication where one device or process (master) has control over one or more other devices (slaves). Some of the devices mentioned in this thesis may support different protocols depending on communication types where they act as master or as slave, while most protocols are available in both master and slave modes. (6)

4 Hardware

4.1 Remote Terminal Units

RTU is a physical device designed to send and receive data from the physical protection and control units and then deliver data to a control room computer. An RTU may have I/Os for measuring and sending analogue and digital signals, such as power measurement at 4 - 20 mA or voltage measurement of 0 - 10 V, which today are common ranges of analogue signals. This analogue data is interpreted and encoded into digital data that is then sent to the monitoring equipment. An RTU can often be compared with PLC and DCS controllers, but the PLC is more aimed for use with local processes and makes use primarily of physical actuators. The PLC is also more flexible since an RTU has a more focused "task." A modern RTU can have a completely integrated HMI that can be accessed via network communication. PC software needed for HMI access is often only a standard web browser with java as a plugin.

4.1.1 RTU models from different manufacturers

As of today many RTU models are available, some of the more advanced models can even be called a computer. The manufacturers for these devices often have a wide product line but covering all of those in this thesis is unnecessary. My focus on these devices is their main features and what types of connection protocols they support. The products often used by VEO are:

- ABB - RTU560
- Netcontrol - Netcon@500
- SIEMENS - SICAM 1703

4.1.2 RTU 560

The RTU 560 family from ABB has a flexible and modular design which allows for easy station installations and upgrades. The RTU 560 can be integrated into existing infrastructures, as it supports all modern international and most third party telecontrol communication protocols. Integrated HMI is available, and PLC programming languages according to IEC 61131-3. Time synchronization can be made via protocols, GPS, DCF77, IRIG-B or SNTP. Configuration files and licenses are stored on a flash drive, meaning that a faulty device can easily be replaced by a new one without the need for re-upload of files.

RTU 560s are often big rack mounted units with many I/O connections depending on their applications. A new smaller unit, the DIN-rail mounted RTU 560G, will be analyzed and tested further on in this thesis.



Figure 7. ABB RTU560G remote terminal unit. (2)

4.1.3 Netcon 500

The Netcon 500 has been designed to act as a data concentrator and a protocol converter in station automation systems. The unit can be applied at outstations in which large numbers of protection relays, programmable logic controllers, telecontrol outstations and other intelligent devices with a serial line or a LAN interface are connected to a SCADA system, and where the required I/O points are many. Netcon 500 has the time synchronization via GPS and can also be synced by master stations. It can communicate with several master stations simultaneously, and utilizes a Windows based parameterization software tool.



Figure 8. The Netcon 500 unit. (5)

4.1.4 Siemens AK 1703 ACP

Siemens AK 1703 ACP from the SICAM products offers automation, telecontrol and communication functions for combined flexibly and in full compliance with IEC 61850. Especially noteworthy is the possibility of offering client and server functionality on only one Ethernet interface. You can use the AK 1703 ACP as:

- a central unit or telecontrol substation,
- data node or front-end
- automation unit
 - with autonomous function groups
 - with local or remote peripheral equipment.

Configuration and all parameters are saved on a flash card. This means that in an event of a fault, a replacement device can be put into operation immediately without the need for PC or resetting of parameters. TOOLBOX II is the program used for configuration, loading, system diagnostics, testing and documentation. The TOOLBOX II implements the IEC 61850 standard, meaning that devices of other manufacturers can be handled in projects as well. Up to 66 serial interfaces for local and remote communication are supported.



Figure 9. The Siemens AK 1703 ACP unit. (7)

4.2 Protective relays

Not all protective units from all different vendors will be mentioned below, just the ones commonly used by VEO.

A protective relay (also in this context referred to as IED) is the unit for supervising electric circuits ranging from low voltage to high voltage installations. Its main objectives are fault detection and reading measurements supplied from connected devices, such as current transformers, voltage transformers, position indicators etc. When a measurement is out of specification the relay then gives trip commands to a circuit breaker. A trip alarm is also triggered which control personnel should be able to receive and process.

- VAMP standard protocols
- ABB – Relion series
- Siemens - SIPROTEC product family
- GE – Multilin

4.2.1 VAMP

Vamp Ltd specializes in protection relays, arc flash protection and measuring and monitoring units for power systems. Their products are suitable for a wide range of protective applications, and the required functionality of the devices can be selected with the Vampset configuration software.

4.2.2 Relion

Relion series from ABB are feeder protection and management relays. These products provide versatile communications as well as sophisticated functionality for event, alarm and fault analysis.

4.2.3 Siprotec

Siemens SIPROTEC product family for line, motor and generator protection offers an integrated solution which starts with extensive protection and control functionality, flexible communication possibilities and uses one single operation program, DIGSI, for all SIPROTEC protection relays.

4.2.4 GE

GE (General Electric) Multilin 3 series providing protection, control, monitoring and metering, and both local and remote user interfaces in one assembly. They give various protocol support through front USB, rear serial, Ethernet and fiber ports.

4.3 Control center software

4.3.1 ABB MicroSCADA

MicroSCADA is a SCADA system made by ABB for managing and supervising an entire distribution network in utility and industry environments. It gives access to real-time information from measuring units. It can be implemented in both electrical and non electrical processes. It also provides a wide arrange of protocol support which makes it easy to implement with devices from many different vendors. Support for protocols can be called modular because MicroSCADA is designed so that you can install libraries and connectivity packages specifically meant for a project. By continuous development MicroSCADA has been able to keep up with the ever growing and changing computer market. During my further tests I will get familiar with MicroSCADA. (2)

5 Configuration process

The following configuration steps describe the procedures used when trying to accomplish the goals of this thesis work. The configuration process means going through one software after another in a certain order and will be explained in that same order. As none of the first attempts after a software configuration was instantly successful all these steps had to be figured out one by one. Problems and mistakes first made are explained along with their solutions. The proper configuration steps and screen shots from the different software are explained in appendix 1.

At the beginning of the configuration all units and software were at default settings clean from any earlier configurations except for the REF 541 relay which had been preconfigured with the necessary functions and mimic display. But the bus connection module SPA-ZC 400 attached to the REF relay was also clean from configuration.

To be able to completely configure the RTU 560 for use with IEC 61850 with PLC and HMI functions the following software is required:

- RTUtil
- Web browser with Java plugin (e.g. Internet Explorer)
- Multiprog WT
- HMI editor
- PCM 600 with necessary connectivity packages
- IET/CCT

Additional useful software and accessories:

- Windows Notepad
- Compact Flash card reader for PC
- Microsoft Excel

In this thesis work the REF 541 relay used the SPA-ZC 400 module for SPA to IEC 61850 conversion. To be able to configure the SPA-ZC 400 unit the CET software is needed. To change or acquire the configuration of the REF 541 relay itself a software named CAP505 is needed. And for VAMP relay configuration the Vampset tool is required.

Before any work was begun the station topology needed to be defined. This would help in seeing the big picture on how the test station should function and it was also easier to keep track on what solutions are possible and all the needed hardware. The station built is illustrated below in figure 10.

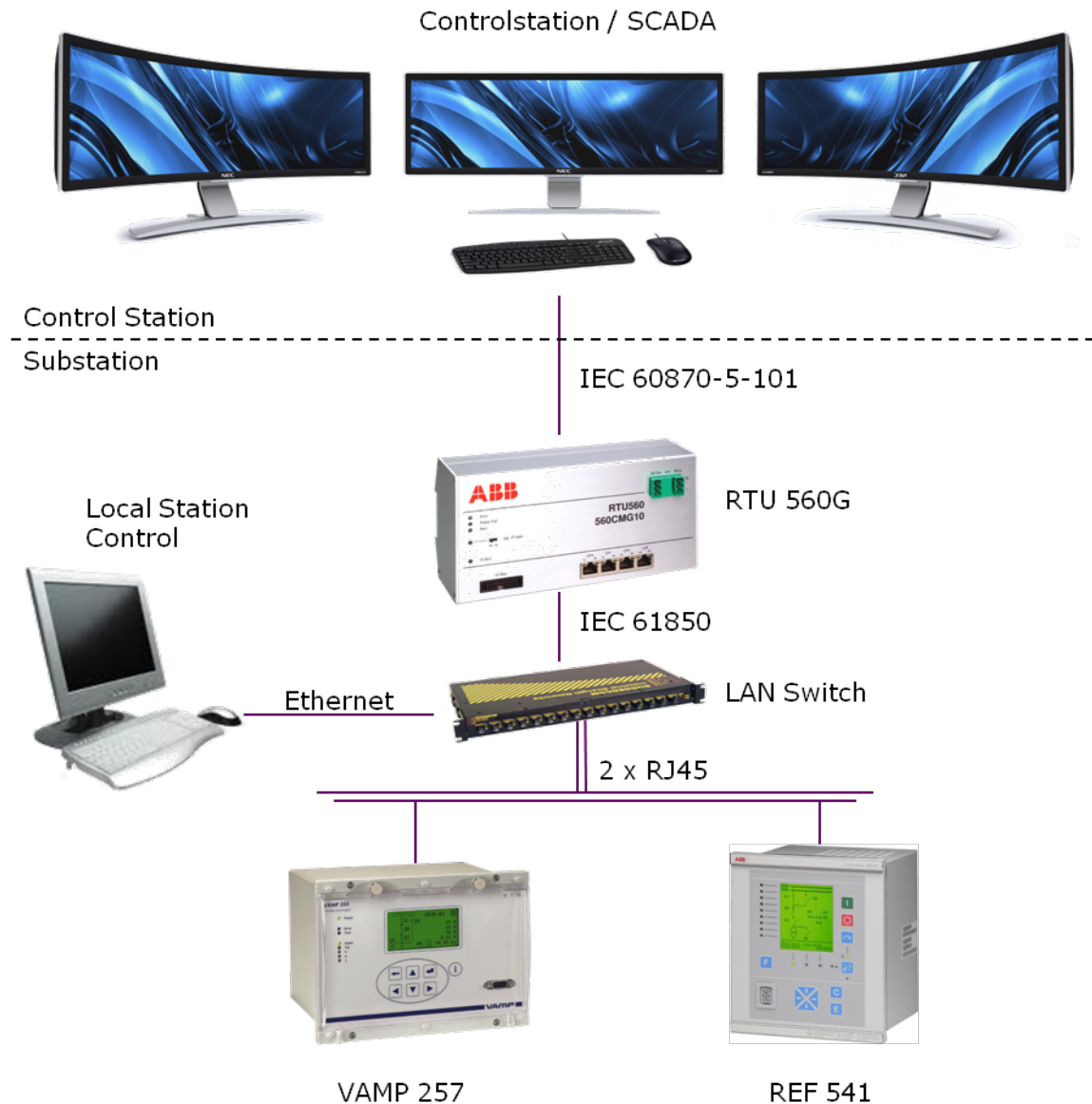


Figure 10. Thesis project system overview

A so called I/O list was also made which defines which signals to use and their addresses for the IEC 60870-5-101 communication. Signal names and their state definitions were also added. The signal list works as a reference when configuring the communication just so that it is easier to keep track of the required signals. See appendix 2 for the I/O list.

5.1 REF 541 and CAP 505

The CET tool, which is used to configure the SPA-ZC 400 needed to have the REF 541 configuration file (file format .ar) imported. REF 541 can natively communicate over LON, SPA and DNP 3.0 protocols and since the SPA-ZC 400 module was attached to the REF 541 it was possible to simply set the SPA-ZC 400s IP address and receive the .ar file through Ethernet TCP-IP connection using CAP 505.

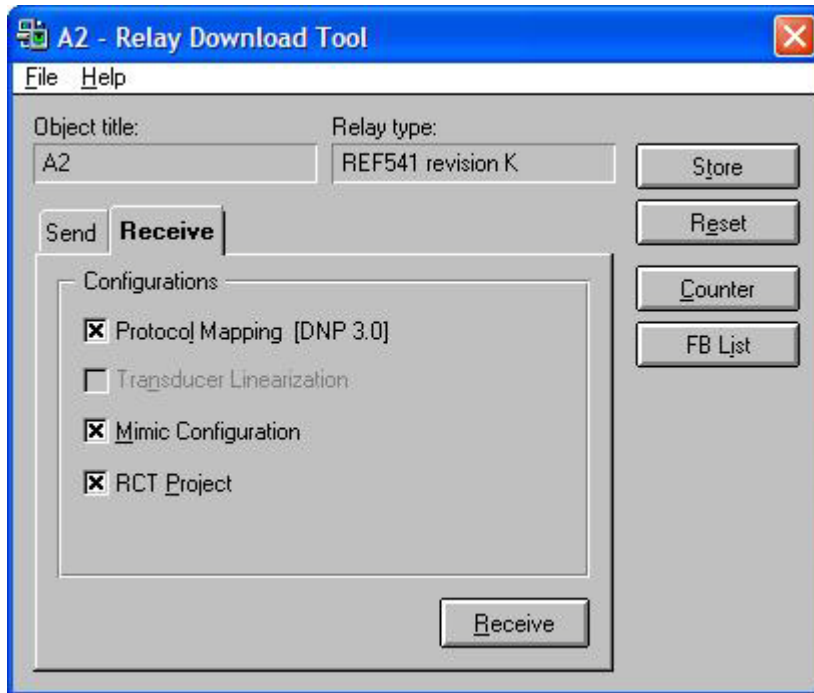


Figure 11. CAP 505 relay download tool.

5.2 CET and SPA-ZC configuration

When the .ar file had been acquired it was possible to begin with the SPA-ZC 400 configuration, and that was made possible with the CET programming tool. A new project was initialized and the .ar file was imported and IEC 61850 process objects were generated automatically with all their corresponding SPA addresses pre-defined. Some small adjustments to properties were made such as report control block ID name and the Ethernet addresses of the SPA-ZC unit. A .cid file containing important communication configuration data was exported and renamed to .icd for later use with the CCT tool. Renaming .cid to .icd had to be done for the CCT tool to recognize the file before import.

5.2.1 Problems occurring with CET

The first SPA-ZC unit used had an older firmware than the firmware set in CET, thus the Ethernet address given in CET would not apply to the SPA-ZC unit but instead it would assume the address given as its primary SNTP server, which is the address of the RTU 560. For the communication to work the units could not have the same addresses. This problem was resolved by getting a newer SPA-ZC unit with firmware version 2.x making it possible for the settings given to apply correctly.

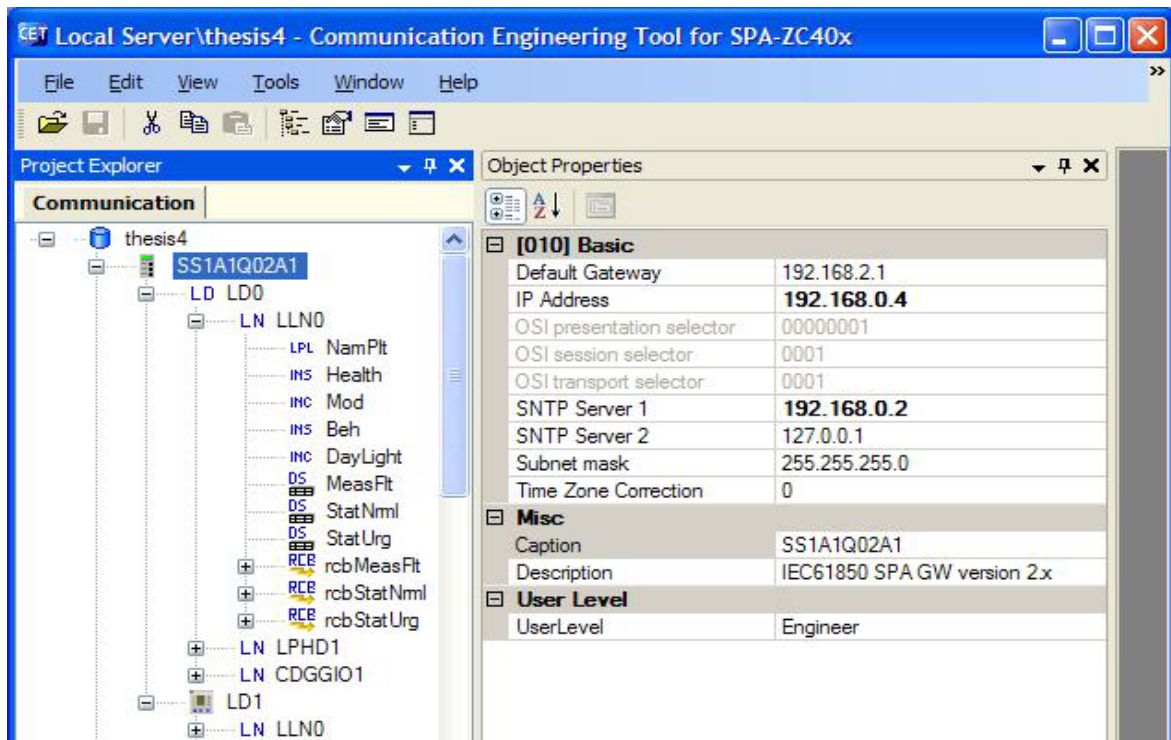


Figure 12. Communication engineering tool and Ethernet addresses.

Another important thing noticed was that when exporting the configuration both to the unit and to the .cid file, the checkbox *Export Datasets DO level* had to be checked or otherwise communication with the RTU 560 would fail.

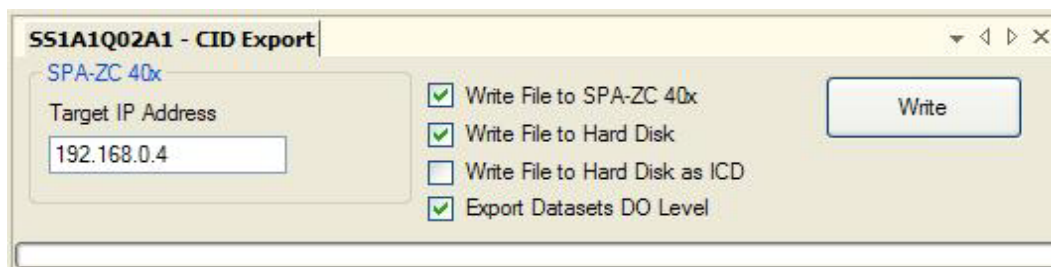


Figure 13. CID export from CET.

5.3 Vampset

The VAMP 257 has native support for IEC 61850 and an Ethernet port on its backside, so no extra equipment was needed to configure the protection and communication settings. Only a few of the protection capabilities were enabled, since most of them function in the same way from a communication viewpoint. VAMP 257 supports the use of three datasets to which the user can address any signal or object he wants. A common procedure is to assign measuring data to one dataset, indications to another etc. Since there were not many signals to use so all of them were assigned to one dataset, DS1. The datasets are then sent to the master station as report control blocks, *RCB*. There is also the possibility of assigning the datasets to *BRCBs* or *URCBs*, where buffered is where data will be stored even if no communication is going on. This makes it the preferred setting although *URCBs* won't store data changes during a communication break. It is important to keep the report ID names the same in both sub device and master station for the information exchange to succeed. The configuration was uploaded to the relay and after that an *.icd* file could be downloaded and exported for later use in CCT.

5.3.1 Problems with VAMP

Attempts to import the *.icd* file generated by Vampset into CCT would fail and cause errors, so by exporting an *.iid* file instead and rename that file to *.icd* and import it to CCT it worked, as did also the *SPA-ZC .icd* file. The *.iid* file contains all the necessary information for that to work although it may not be the recommended approach.

Later in the station testing phase, the local/remote setting was reversed in MicroSCADA compared to the VAMP setting. Digital input 15 was used as the external switch for changing unit local/remote control, and changing the input from normally closed to normally open in Vampset would not change the switching states. Nor was there any change when attempting the same NO to NC switch for bay A1 L/R object in MicroSCADA. The answer to this was to make a small logic program with Vampset so that digital input 15 would trigger an NOT that would later on change state on the VO6 (virtual output 6). VO6 would then change the unit L/R setting.



Figure 14. Logic program in Vampset for inversed local/remote setting.

The same problem was encountered with the REF 541 unit and it was resolved in the same manner using the RTU 560 PLC for inverting the local/remote signal state from REF 541.

5.4 RTUtil

The RTUtil tool has been developed by ABB specifically for use with the RTU 560 models and it is the main software during the configuration process. In RTUtil there are three main views, *Network Tree*, *Hardware Tree* and *Signal Tree*. Network tree is where the station topology is defined by adding the main RTU and lines to other station nodes. In the hardware tree the main RTU is linked and the hardware of the RTU is added so that the RTUs internal communication structure is made. In the signal tree all the process objects (event signals) are added and given their unique object identifier names.

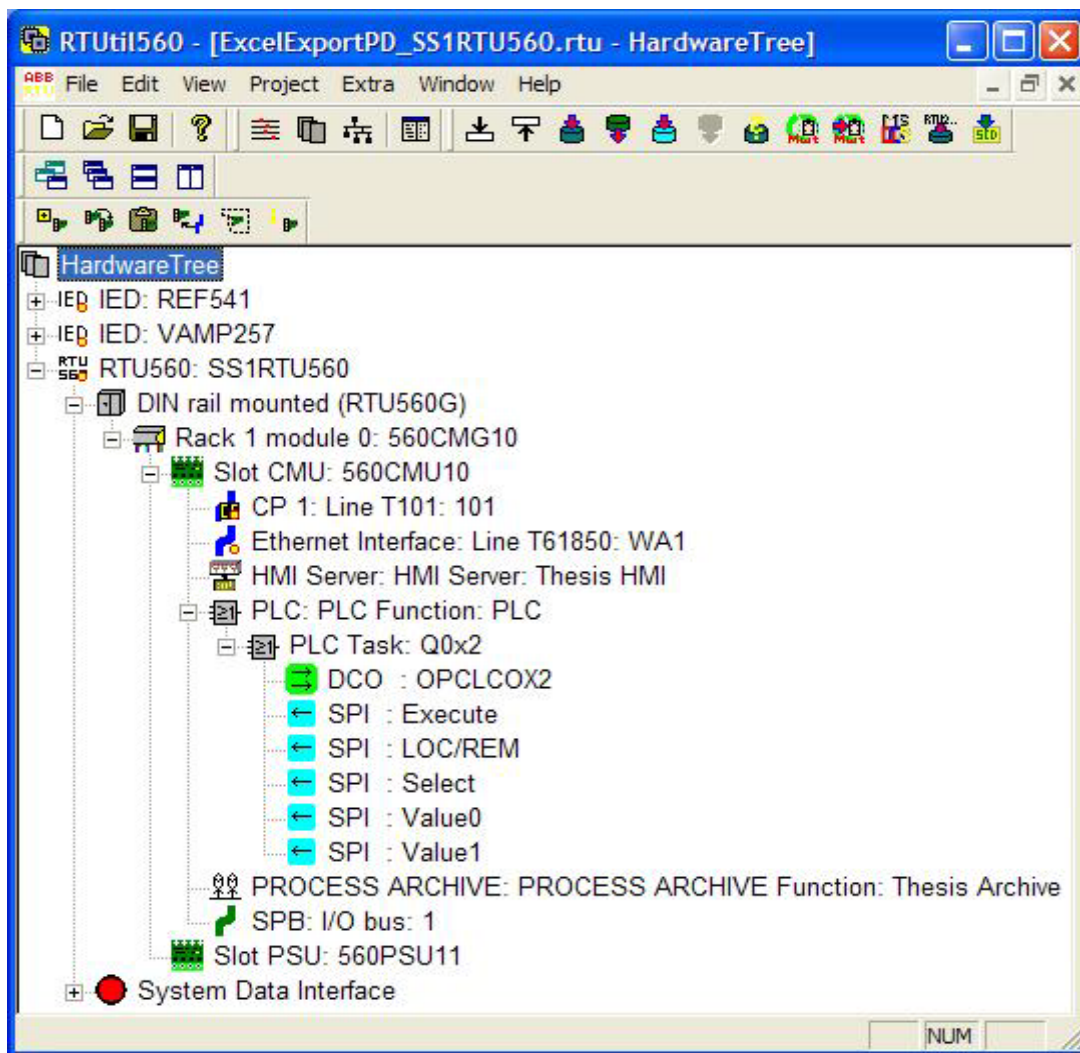


Figure 15. RTUtil Hardware Tree.

At the beginning the hardware and protocols to be used were defined in RTUtil by adding the items to the network tree and the hardware tree. In the network tree, the RTU and all the outgoing lines were defined and also the IEDs connected at the other end of those lines. In the hardware tree the type of RTU was defined and the connected lines were linked. An I/O bus had to be added to the CMU board in the RTU since it is crucial for the internal communication to work. PLC, HMI and process archive functions were added to the CMU. The PLC was also given all the necessary “virtual” objects necessary for the duplication or modification of signals. Any specific details of the local control PC were not added as it wasn’t necessary. All added items were given unique names and most were kept at default settings. Important settings made were IED name, IP address (192.168.0.2) and the parameter settings for the IEC -101 protocol such as transmission with full timestamp and how the RTU was to be time synced from the control station. The RTU was also set as SNMP server so that the relays could be time synced by the RTU. The control authority timeout for the HMI had a very low setting at default. It was therefore increased to 3600 seconds and that is the time that the HMI user will be able to control the station before having to request for control authority again.

Once all items and settings were done in RTUtil the next step was to export the Excel files from RTUtil. These files contain all settings made making it possible to import them into another project and give that project the same settings. The main idea for the Excel files is that process objects can be added manually by typing in all object settings row by row, and when engineering for IEC 61850 communication the process object are inserted automatically by synchronizing a SCL (.scl) file with the Excel file using RTUtil. After the export the projects Excel initialization had to be made. The Excel export and Excel initialization procedure may be viewed in Appendix 1 on page 11 and the synchronization is explained in section 5.7. An .iid file, containing RTU specific information such as name, addresses and capabilities, was also exported and renamed to .icd as it is needed later on when the data sets are assigned to clients in CCT.

5.4.1 Problems with RTUtil

One mistake made early on was using version 9.7.1.0 of RTUtil and since the firmware on the RTU 560G was 10.0.1.0, the RTU would not accept the configuration files made with the older version. By acquiring version 10 of RTUtil the problem was solved.

Another problem encountered was that with the RTUtil software installation a template .icd file for the RTU functions came bundled at the installation directory. That file is meant for import into the CCT tool and when doing so a faulty configuration would be generated. Instead it was important to extract a new .iid file and rename it to .icd for the configuration to be successful, i.e. the same procedures as with the CET and Vampset exports.

5.5 PCM 600 configuration

Protection and Control IED Manager 600, a software developed by ABB is a diverse and useful tool when configuring substation relays for IEC 61850 communication. The software has many great functions for relay configuration upload and download, and for exporting .scd files. However, it requires connectivity packages for each type of relay that is being configured, so that it can handle data properly for each device in use. No connectivity package was available for the RTU or the VAMP relay, which made it troublesome to import their configuration files. Only the model type, the revision and the unit name could be imported, which was not of much use for the .scd file to be exported. Therefore only generic IEC 61850 IEDs were defined, even for the REF 541, with names and IP addresses for the three logical units to be used in the project. The IEDs have to be placed under a bay and that was also done for the RTU even if it is not physically in a bay.

The PCM 600 tool would not be fully utilized in this project. It was merely used just for defining the station topology extracted within the .scd file later needed in the CCT tool.

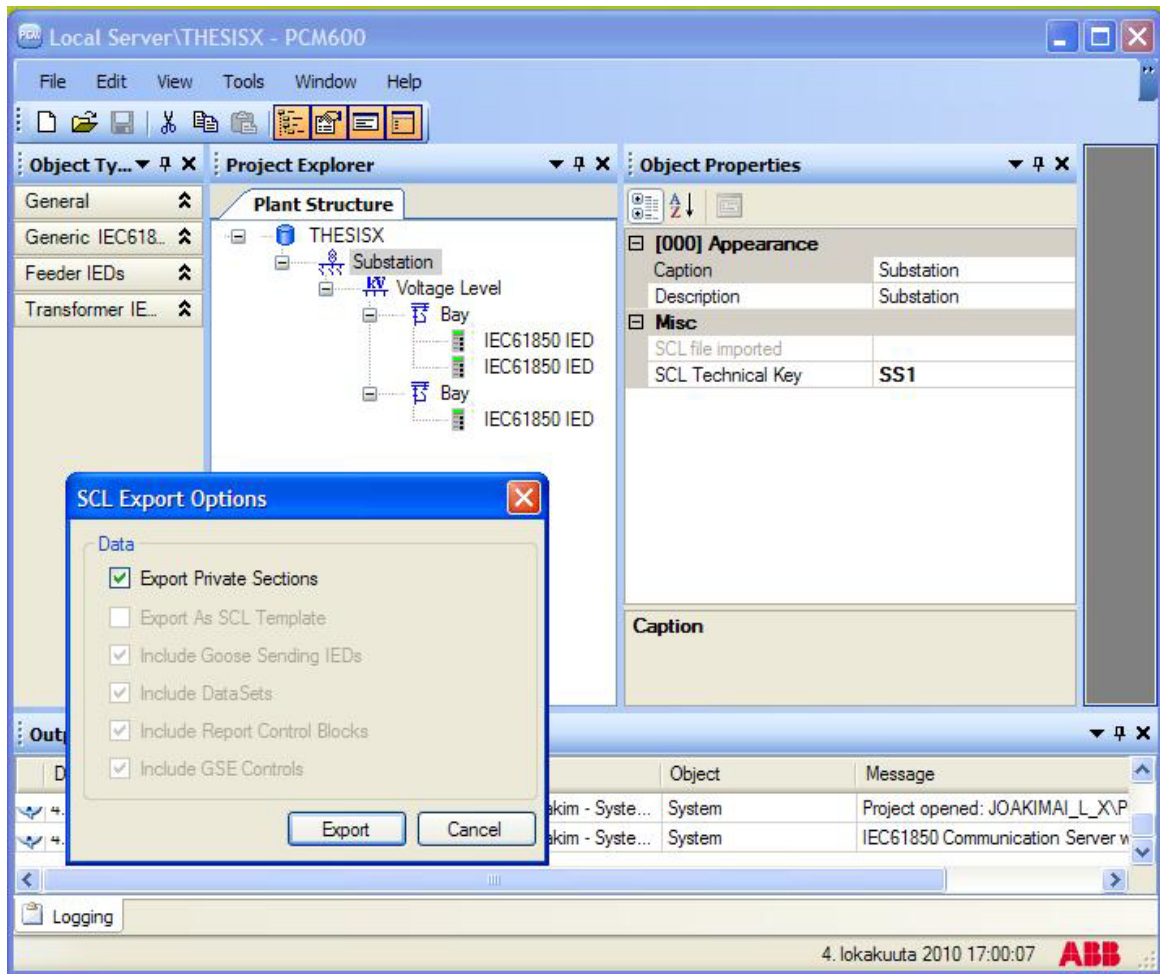


Figure 16. PCM 600

5.6 CCT/IET

CCT is a licensed tool created by ABB. It comes bundled with the PCM600 Engineer Pro software package and is meant for handling process objects and report control blocks throughout the station. The version used (CCT 3.2.1) for designing this project was a bit unstable and tended to crash quite often. It also did not show all IEDs on some occasions, which led to a lot of retries.

Firstly a new project was created and the previously exported .scd file was imported so that the station structure would appear automatically along with a station bus, the bus representing the actual Ethernet network used. Afterwards the three generic IEDs appeared as empty devices, so into each of those their corresponding .icd files were imported, which would add their names, report control blocks and process data signals.

It was then important to check that all the units had correct IP addresses and that the devices all had the same bus connection selected.

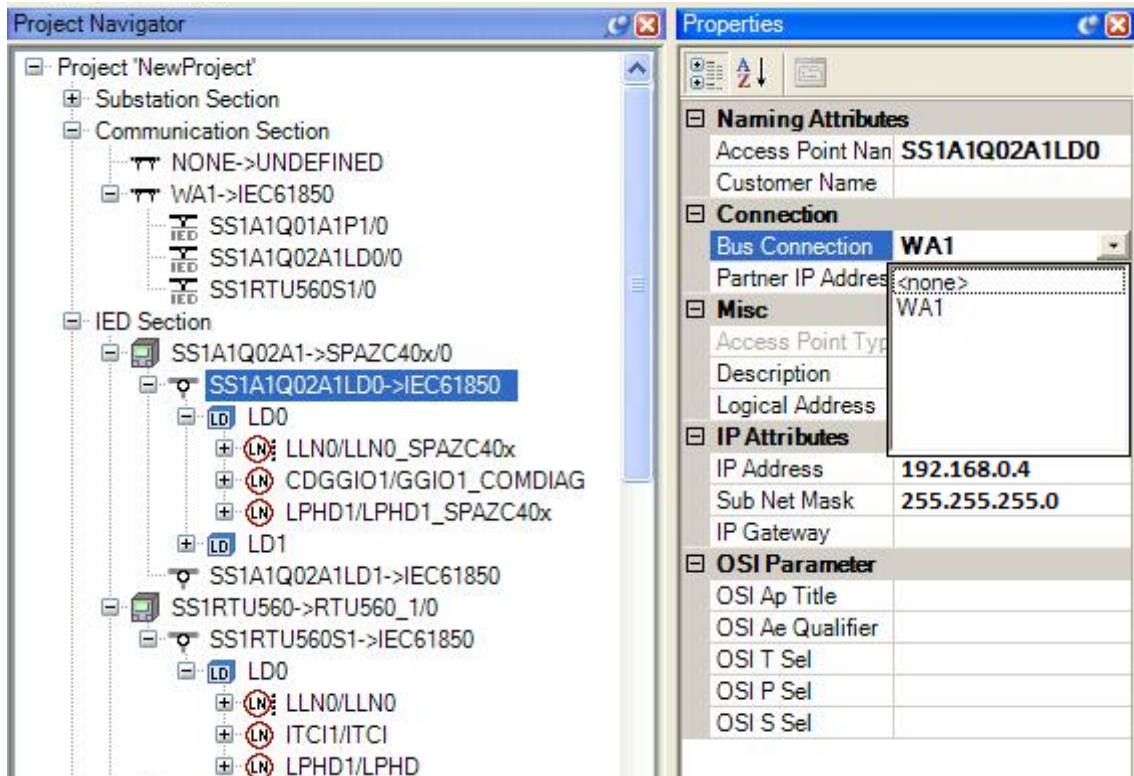


Figure 17. Selecting the bus connection for SPA-ZC in CCT.

With the same bus connection selected it was necessary to select *update IEC 61850 data flow* from the tools menu and that will automatically apply the RTU as a client logical node to the report control blocks sent by the relays.

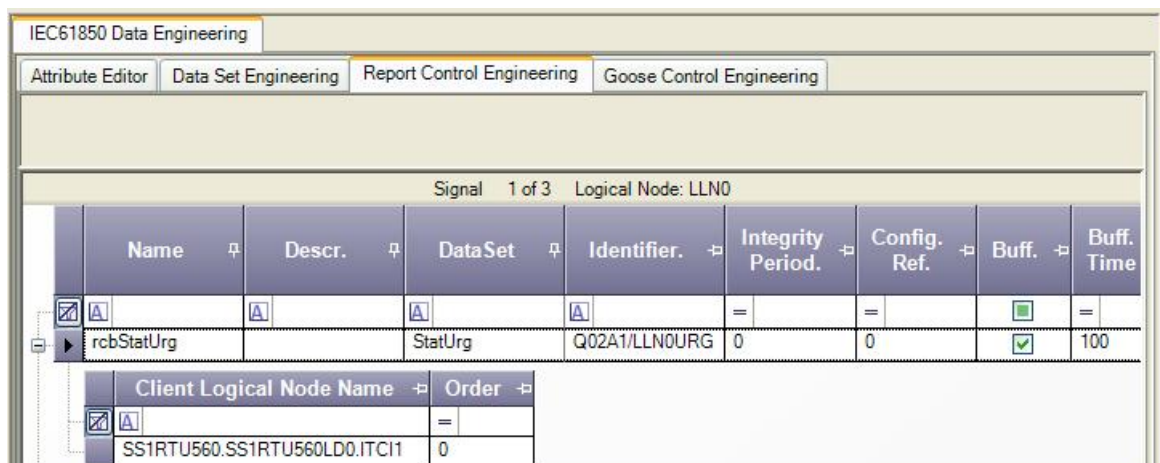


Figure 18. RTU 560G seen as a client logical node for the SPA-ZC RCB rcbStatUrg.

Once all settings were made a new .scd file was extracted.

5.6.1 Problems with CCT

One major problem encountered during the testing stage was that after a new configuration upload to the RTU, the RTU would never get past the boot stage, meaning that it would reboot over and over making it inaccessible for any troubleshooting. The only way to reset the RTU was to pull out the compact flash card and insert it to the PC for file access which enabled me to manually remove the configuration files from the flash card. The insertion of the flash card back into the RTU would make the RTU boot with default settings and accessible for a retry. The problem originated from the CCT configuration and that the redundant RCBs that came with the VAMP import had to be removed or otherwise the RTU would go berserk. An exact reason or explanation for the behaviour could not be determined.

5.7 Back to RTUtil

Now it was time to get the process signals added into the RTUtil project, and to do so the .scd file generated by the CCT tool had to be imported and synchronized with the Excel file previously exported with RTUtil. The process signal attributes are then written into the Excel file. It is a good idea to create a backup copy of the Excel file before the synchronization, as the file will be overwritten.

By later opening the Excel file with Microsoft Excel it was possible to see all process objects available for use with the station. Due to the limitations of the license issued with the RTU 560G used in this thesis work not many process objects were allowed to be used simultaneously; it was limited to about 60 objects which would be easily filled with just two relays and a few PLC objects. This meant that a few objects found in the Excel file had to be removed, since the total amount exceeded the limitation. The objects were scattered and needed to be given a unique object identifier for use in RTUtil. For a new user of the IEC 61850 standard it can be hard to recognize the objects. See figure 19 for a view on how the objects are inserted into the Excel file, rows are horizontal and columns vertical.

IED name	Logical Device Instance Name	Logical Node Prefix	Logical Node Class	Logical Node Instance	Signal Data Object Name	Signal Common Data Class	Signal Data Attribute Name	Signal Data Type	Signal Function Code	Signal Support select before operate	Signal Support enhanced security
IEC61850 Address											
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	Y / N	Y / N
CNSS1IN	CNSS1LD	CNSS1LNP	CNSS1LNC	CNSS1LNI	CNSS1SDV	CNSS1CDC	CNSS1SAN	CNSS1SDT	CNSS1SFC	CNSS1SSO	CNSS1SES
SS1A1Q02A1	LD1	CB	CSWI	120	Pos	DPC	Oper.ctVal	BOOLEAN	CO	Y	N
SS1A1Q02A1	LD1	CB	CSWI	120	Pos	DPC	stVal	Dbpos	ST	N	N
SS1A1Q02A1	LD1	ESW	CSWI	127	Pos	DPC	stVal	Dbpos	ST	N	N
SS1A1Q02A1	LD1	ESW	CSWI	128	Pos	DPC	stVal	Dbpos	ST	N	N
SS1A1Q02A1	LD1	ESW	CSWI	129	Pos	DPC	stVal	Dbpos	ST	N	N
SS1A1Q02A1	LD1	CBEW	GGIO	187	Alm	SPS	stVal	BOOLEAN	ST	N	N
SS1A1Q02A1	LD1	DEF	PTOC	40	Op	ACT	general	BOOLEAN	ST	N	N
SS1A1Q02A1	LD1	I	MMXU	200	A.phsC	CMV	cVal.mag.f	FLOAT32	MX	N	N
SS1A1Q02A1	LD1	I	MMXU	200	A.neut	CMV	cVal.mag.f	FLOAT32	MX	N	N
SS1A1Q02A1	LD1	I	MMXU	200	A.phsA	CMV	cVal.mag.f	FLOAT32	MX	N	N

Figure 19. IEC 61850 object addresses in Microsoft Excel where each row is one object.

By cross referencing the objects logical nodes and classes with the ones found in Vampset and CET, their origin and objective could be determined. The unique object identifier may be written in Excel or made later in RTUtil, but since Microsoft Excel has superb copy paste functions it is much faster to write them along with the IEC 60870-5-101 address attributes in Excel. The signal type, SPI, DPI, MFI etc. was recognized during the synchronization so that the column could be left as it was. It was also important to mark all cells with Y (yes) at the column *RTUtil560 import* for the objects to be imported later into RTUtil. When the Excel data was imported to RTUtil the project got the same name as the Excel file. When the relays had already been defined, and given the same names as in the new Excel file, all process objects were added to the signal tree and automatically linked to their respective relays in RTUtil. Once a consistency check was done in RTUtil so that no objects would have the same addresses, the RTU files could be built.

Signal type	System data type (SEV, SSC)	RTUtil560 import	Modified	Station	Subnet	Bay	SCADA object	Intelligent electronic device (IED) name	IED name	Logical Device Instance Name	Logical Node Prefix
Signal				Process Object Identification				IED Name			
3 ASCII	4 ASCII	Y / N	1 ASCII	8 ASCII	8 ASCII	8 ASCII	8 ASCII	32 ASCII	ASCII	ASCII	ASCII
STTY	STDT	STIM	STMD	OI01	OI02	OI03	OI04	IEDN	CNSS1IN	CNSS1LD	CNSS1LNP
DCO		N		SS1	A2	Q02A1	Q0CONTR	REF541	SS1A1Q02A1	LD1	CB
DPI		N		SS1	A2	Q02A1	Q0STATUS	REF541	SS1A1Q02A1	LD1	CB
DPI		N		SS1	A2	Q02A1	Q0LOCLW	REF541	SS1A1Q02A1	LD1	ESV
DPI		N		SS1	A2	Q02A1	Q0LOCHIG	REF541	SS1A1Q02A1	LD1	ESV

Figure 20. Signal type and unique process object names shown in Microsoft Excel.

5.8 Multiprog WT

The RTU 560G has a PROCONOS programmable logic controller and to create programs for it the software MWT is used. From RTUtil it is possible to do an MWT export so that the “foundation” for the PLC program is built automatically. It is also possible to launch the MWT application from RTUtil.

Firstly the configuration files built with RTUtil needed to be re-imported to MWT so that variables with their I/O configuration could be accessed and applied to function block I/Os. The main function made for the program was a duplication of a breaker status command so that both relays would open or close with just one command from either RTU 560 HMI or MicroSCADA.

The four extra SPIs made under PLC in RTUtil were used as indicators in the RTU HMI for the states that the boolean signals would get when the duplicated command was given from either RTU HMI or MicroSCADA. Value 1 and 0 on the function blocks would stay the same after a command but SE (select) and EX (execute) states were only active for one PLC cycle, which is a few milliseconds. So for the HMI to briefly show the boolean state 1 it was necessary to add a TOF (time to off) delay set at 4 seconds at the input. A variable type USINT (unsigned short integer) called COT6 was created and given the permanent value of 6 and used so that it would give all commands the *cause of transmission* set to *activation*. The program section made for duplication of breaker commands and the boolean states can be seen in figure 21.

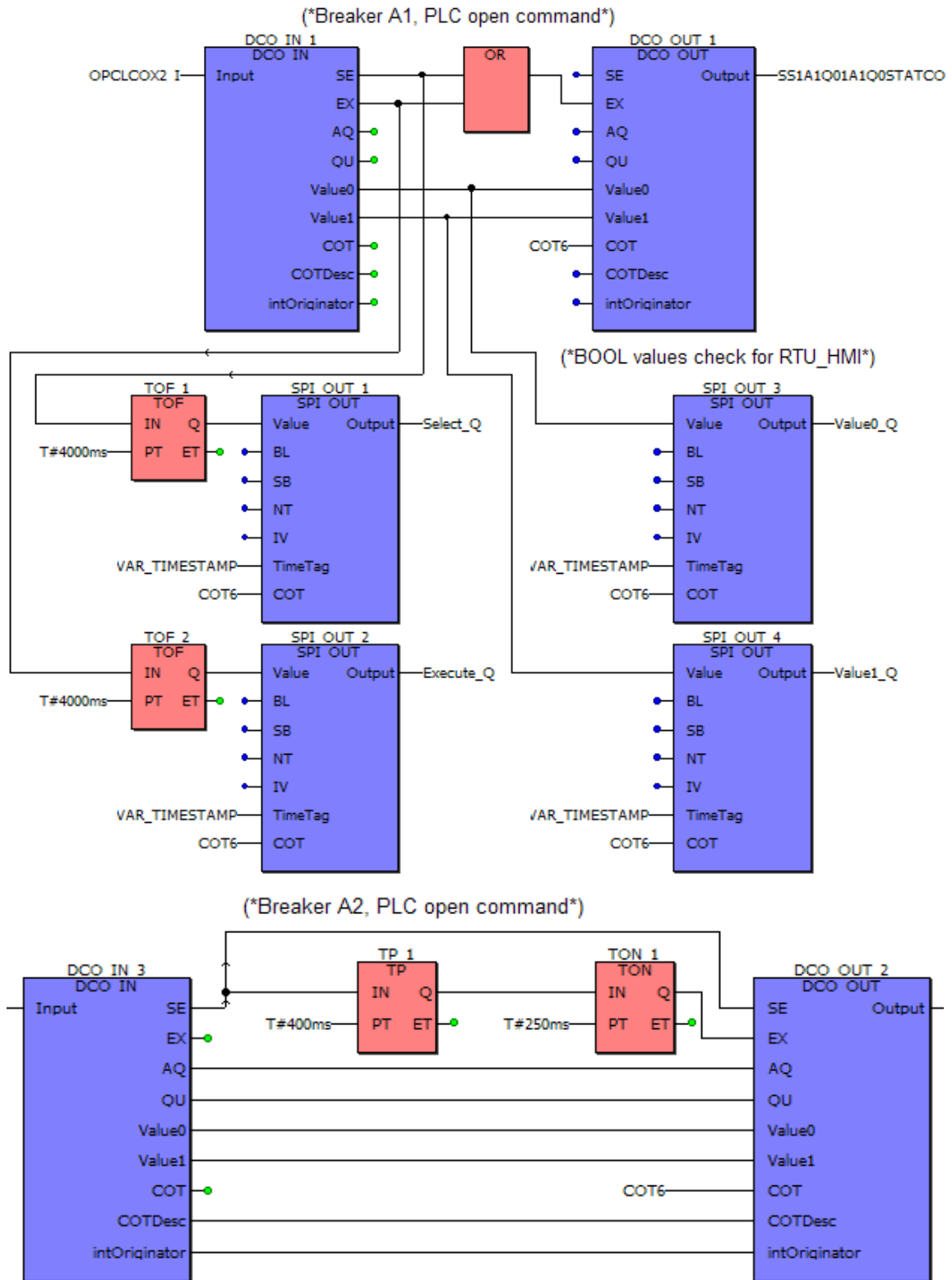


Figure 21. PLC program for duplicate commands and signal bit states.

Apparently the REF 541 relay demanded a signal from PLC objects to be active for at least 100ms, so by creating a 150ms delay from select to execute command the relay functioned as intended. Function blocks DCO_IN_1 and DCO_IN_3 have the same input DCO made under PLC in RTUutil.

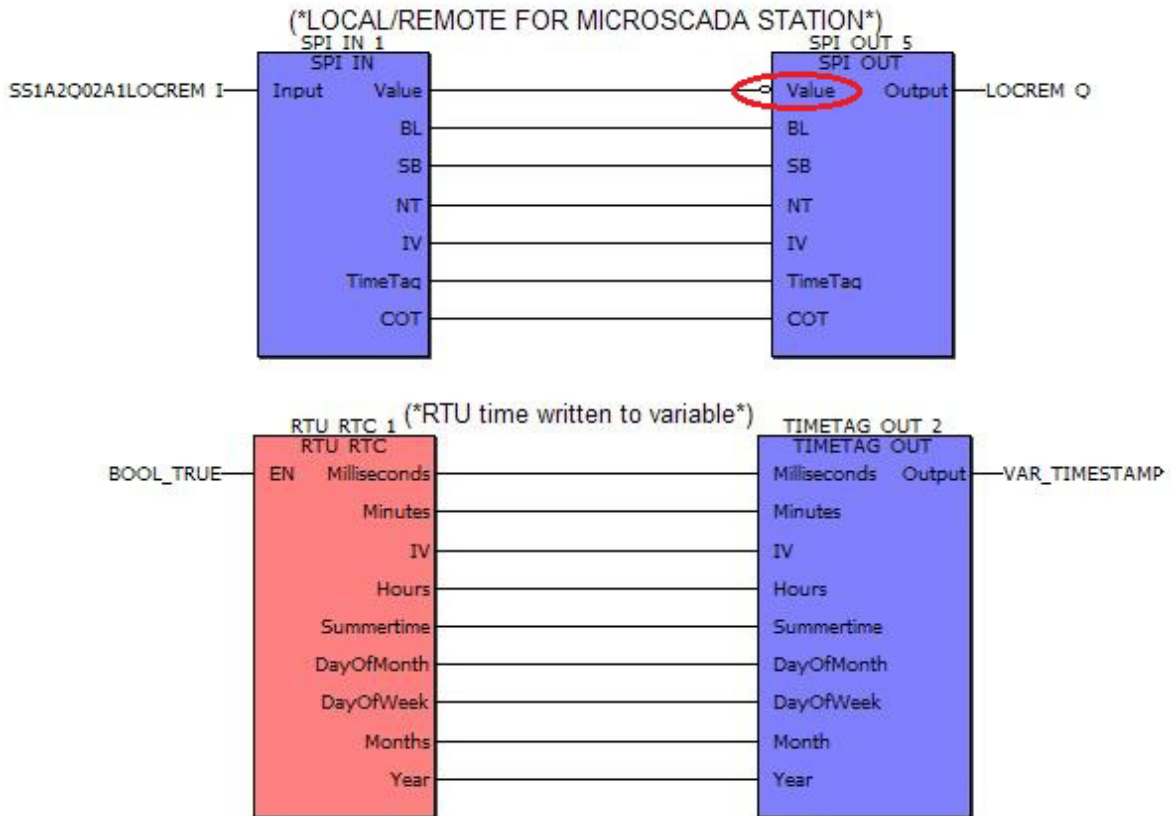


Figure 22. PLC program section for inverted L/R and RTU time written to variable VAR_TIMESTAMP.

When the program had been completed the PLC program files were built. By copying the program file directly to the flash drive using a PC the RTU was able to boot with the PLC program. The file to be copied was found under the PC catalogue address <PLC program root directory>/<program name>/c/<RTU name>/r/plc/bootfile.pro and was copied to <RTU flash drive root directory>/plc.

There is however another way to connect and upload the program over the TCP/IP connection in order to enable the user to debug the program in real-time. To do so a certain .dll file is needed and no such file was acquired or used during this thesis project.

5.9 HMI Editor

The HMI Editor is a simple program written by ABB in Java code and is easy to use. It is similar to Windows Paint. By simply drawing a single line diagram of the station bus and adding symbols such as breakers and measurements a main page was created. By selecting a RTU configuration file into the project, indication and command objects could easily be set to these symbols by just selecting symbol properties. Three more pages were made with

one containing event lists and the two others containing measurement diagrams for the two bays. Buttons for switching between the pages were also created from pictures drawn in Adobe Photoshop, since drawings in jpeg format could be imported and assigned to tasks. See figure 23 for the final HMI view.

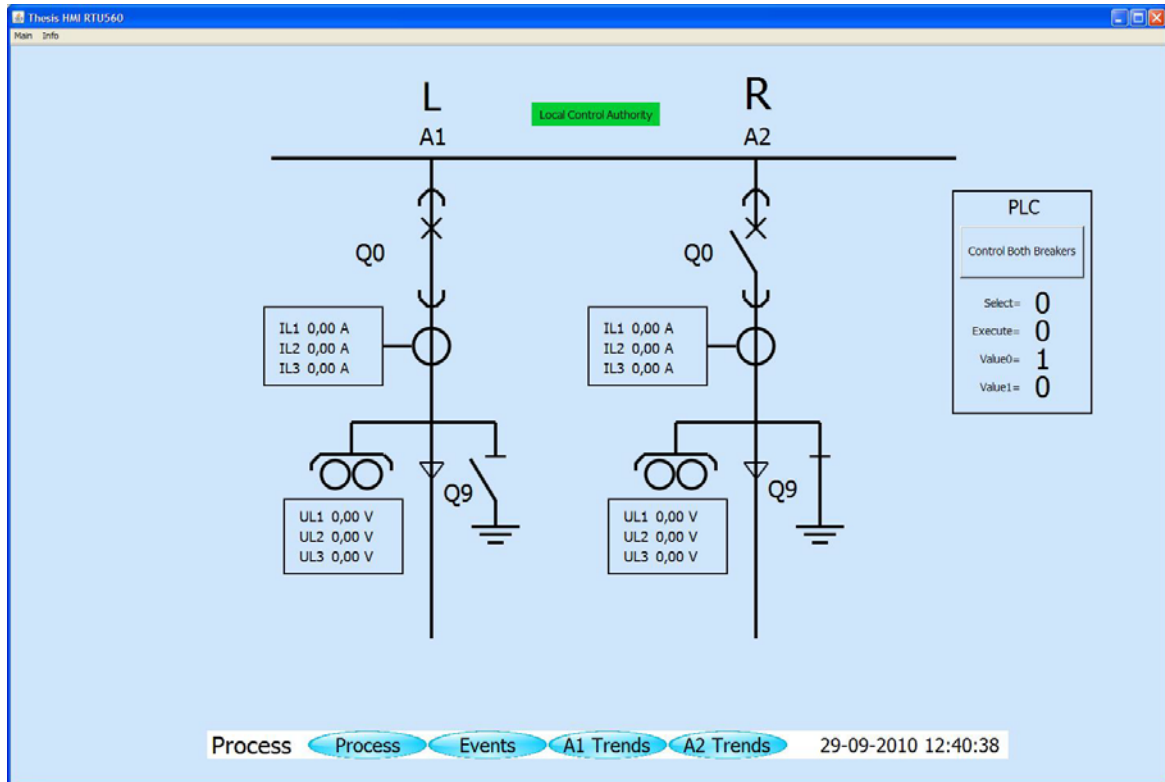


Figure 23. RTU 560G HMI

Custom symbols were made for the indication of breaker position and L/R (local/remote). This was done in the component view editor within HMI Editor.

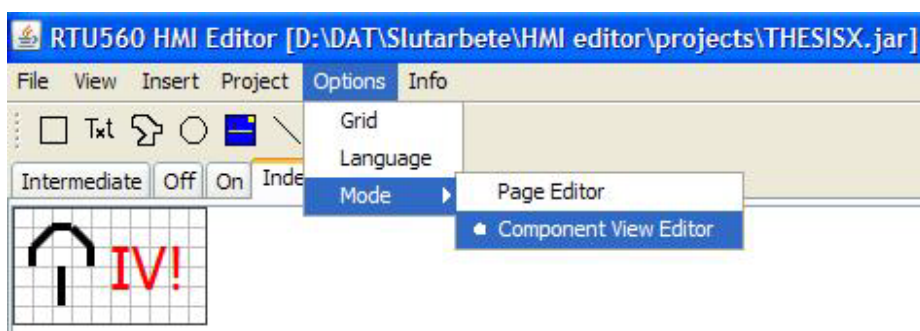


Figure 24. Switching between component view editor in HMI Editor

When all of the configuration files needed to make a working RTU controlled station were done it was time to upload the configuration files built from RTUutil (*.gcd, *.iod, *.oad and *.ptx) along with the HMI configuration file .jar.

The RTU web interface can be accessed by setting the computer in the same IP range as the RTU, in this case 192.168.0.X where the X can be any number ranging from 5 to 254. IP numbers ending with 1 to 4 are already in use by the station devices. As the RTU got the IP of 192.168.0.2 the web interface could be accessed by typing *http://192.168.0.2* in the web browser.

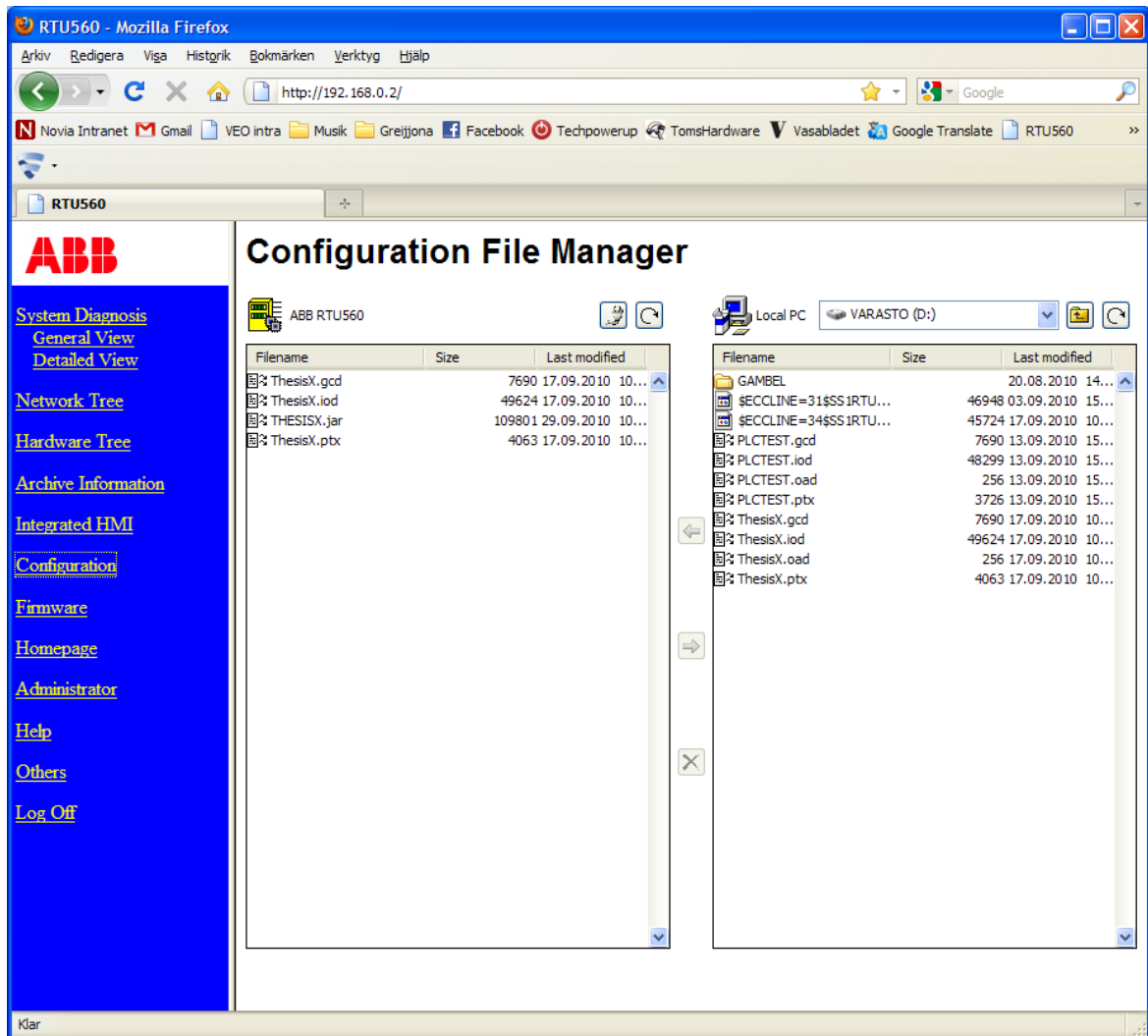


Figure 25. Web interface showing the configuration upload view of the RTU 560.

5.10 MicroSCADA

Lastly the control station was made in MicroSCADA. The communication from the RTU to MicroSCADA was made by using the IEC 60870-5-101 protocol over a RS 232 interface. For the communication to work a custom cable had to be made. The wire connections can be seen in figure 26.

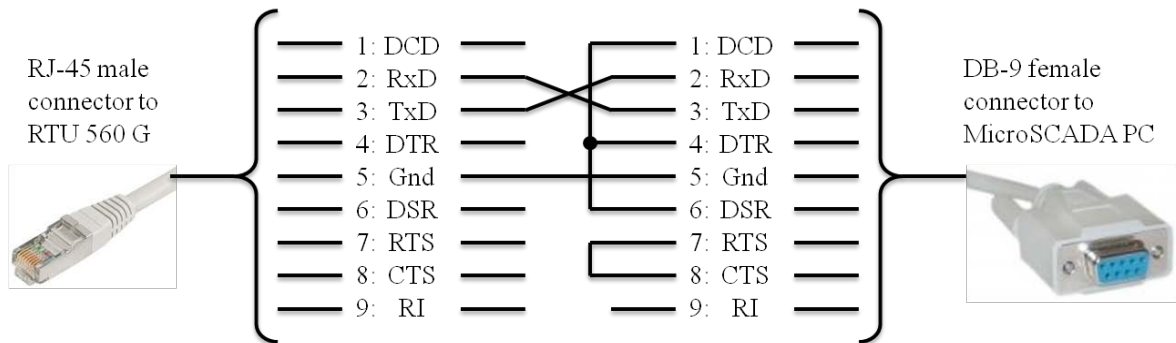


Figure 26. Custom cable made for IEC 60870-5-101 link

By applying the same link properties as in RTUtil to MicroSCADA the communication could be tested by sending a simple time synchronization command and see if the RTU would get the same time as the PC.

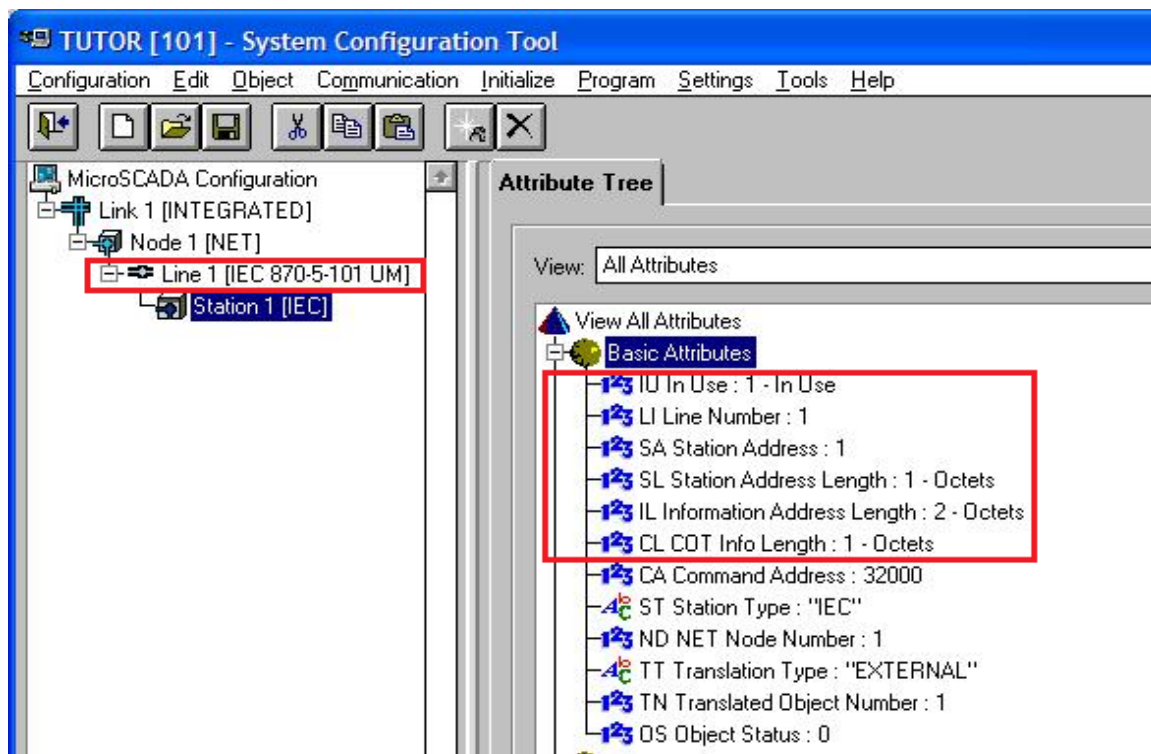


Figure 27. Important link properties set in MicroSCADA.

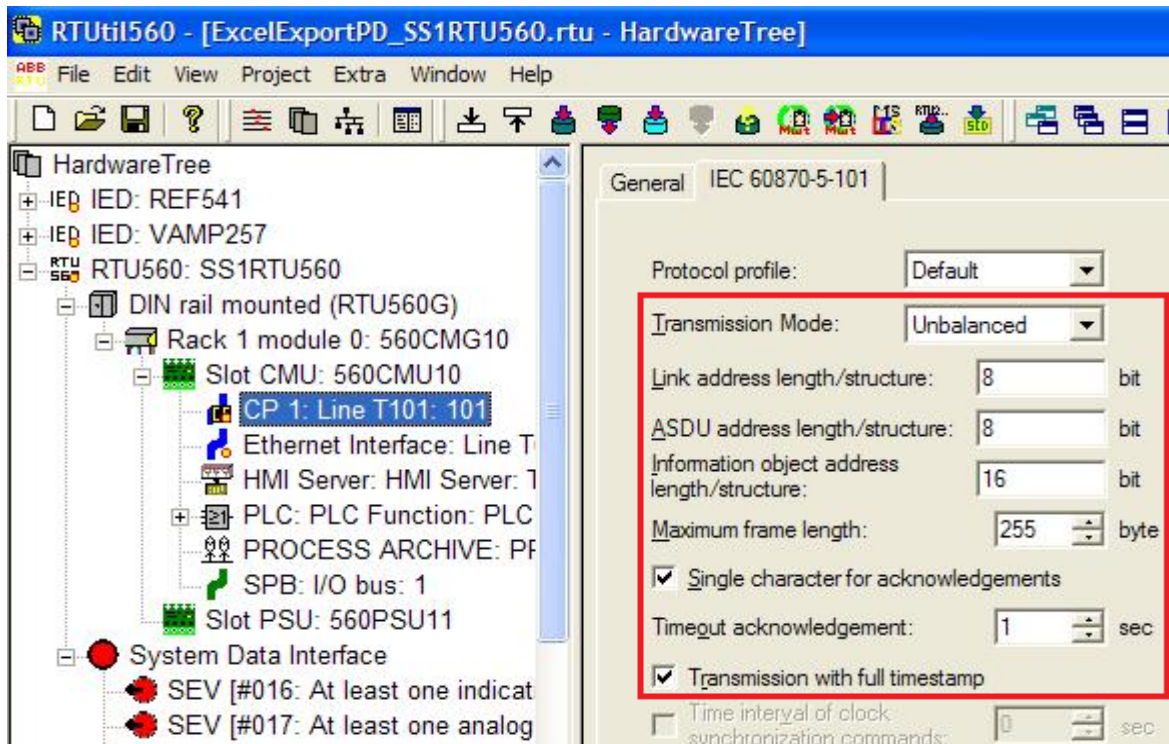


Figure 28. Link properties made in RTUtil.

Once the communication was established it was time to add the process objects, which was done by utilizing *Install standard functions* in *Object Navigator*. This navigator gives the user the ability to easily add and edit process objects relating to the same station device simultaneously. It will also greatly ease the insertion of process object symbols later in the display builder.

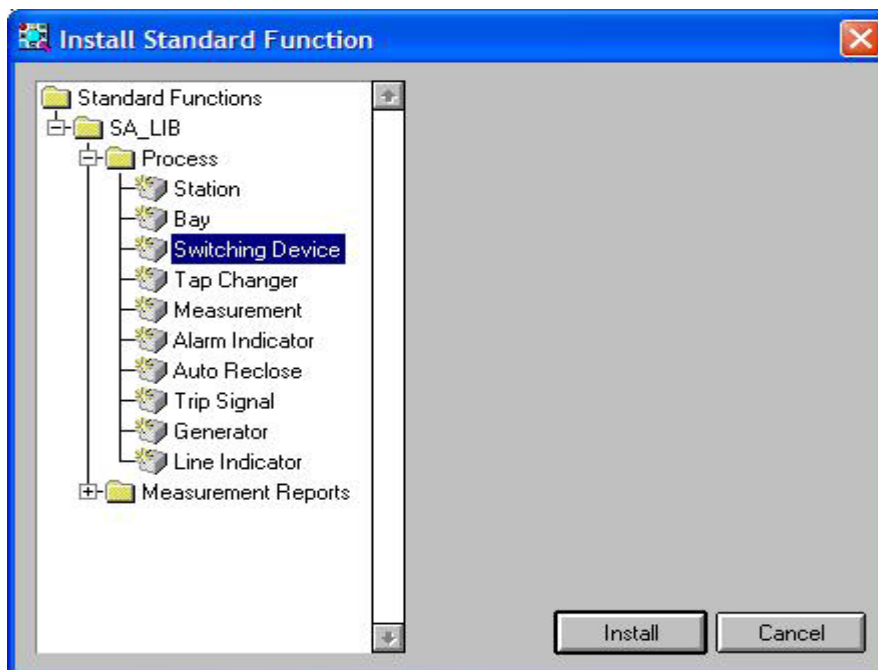


Figure 29. Install Standard Function interface.

SAGR_INST/SAI_SSWI_S - Standard Function

Attributes Programs Tools

STATION_NAME: SS1

BAY_NAME: A2

DEVICE_NAME: Q02A2

LIB_OBJECT_TYPE:

P_OBJECT_LN: SS1A2Q02A1

SWITCHING_DEVICE_TYPE: Circuit breaker

SWITCHING_DEVICE_PURPOSE: Circuit breaker

SWITCH_SECTION:

STATION_TYPE: IEC 870-5-101/104

DEFINE ITEM NAME

from Product: SYS 600

OK Cancel Apply Help...

Figure 30. Setting of attributes to the switching object.

The VAMP 257 switching command was made without the *select before operate* function. It was set in the REF 541 since only the REF 541 had been configured to use the function. When the attributes were made the objects could be created and addresses defined.

Process Object Tool

New Existing Other

Logical Name	Index	IU	SS	UN	[OA]	[OB]	Object Text
SS1A2Q02A1	10	1	2	1	4120		Breaker position indication
SS1A2Q02A1	13	1	2	1	4118		Breaker command
SS1A2Q02A1	15	1	1				Breaker device control block
SS1A2Q02A1	16	1	1				Breaker open interlocked
SS1A2Q02A1	17	1	1				Breaker close interlocked
SS1A2Q02A1	18	1	2				Cause of interlocking
SS1A2Q02A1	19	1	2				Breaker selection on monitor
SS1A2Q02A1	20	1	2				Breaker command event
SS1A2Q02A1	113	1	2				Breaker command

IU SS: 2 - Automatic UN: 1 [OA]: 4118 [OB]:

Create All Create Delete Edit Connect All Cancel

Figure 31. Creation of objects and editing of addresses and basic properties.

When the objects had been set the display builder could be launched and the process display was built by the drag-and-drop method from the object browser into the display.

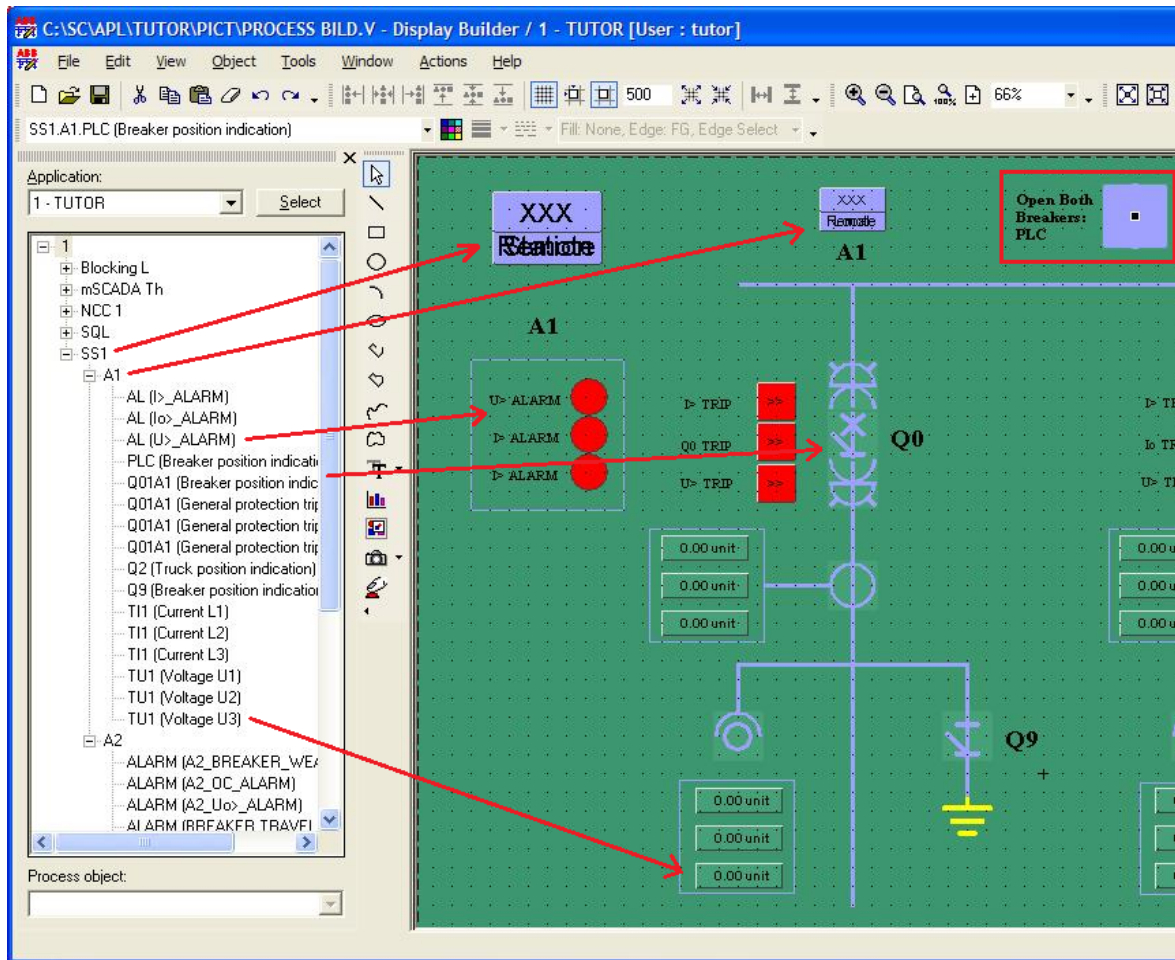


Figure 32. Dragging objects into the process display

A custom button was made for the PLC multiple open/close command by inserting a “virtual switch” seen in the upper right corner of figure 32. A SCIL code line was then added to the button for it to activate the plc open command; `#SET SS1PLC:PSE13=LIST(SE=1, OV=1, OG=1, TY=46, CT=1)`. For the PLC command to close, the code was changed with `OV=0` (object value = 0).

Finally a time synchronization loop for the RTU was made as a command procedure in *Object Navigator*. A new command procedure was made with a SCIL code for the time synchronization command.

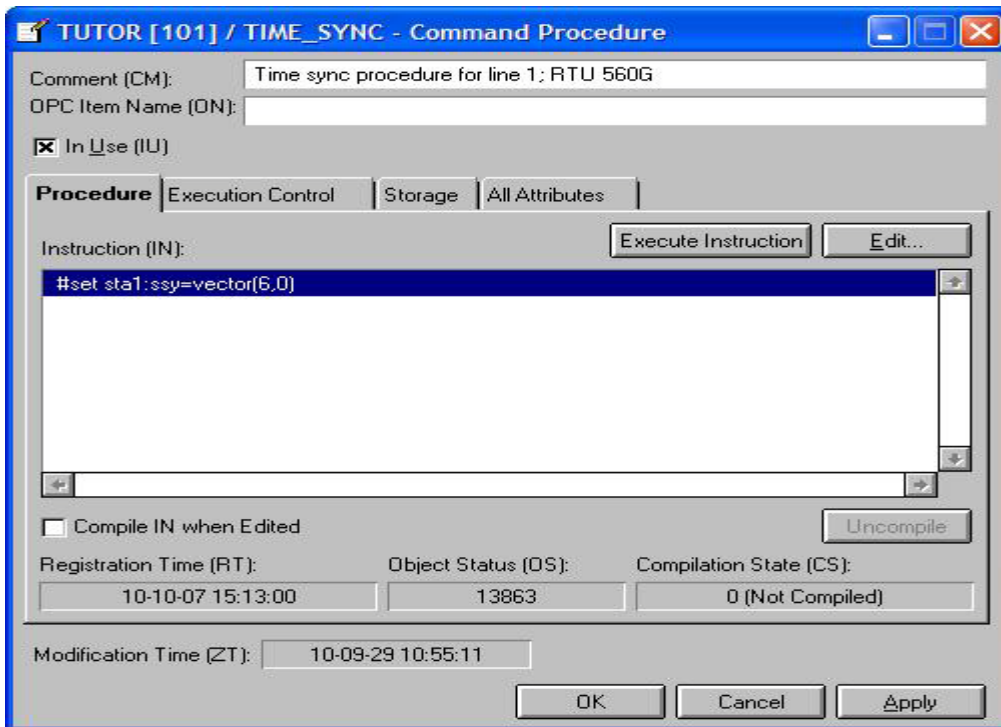


Figure 33. Command procedure for time synchronization.

The command procedure then had to be linked to a time channel that would execute the code over and over after a time interval. This time interval needs to be shorter than the timeout setting in RTUtil.

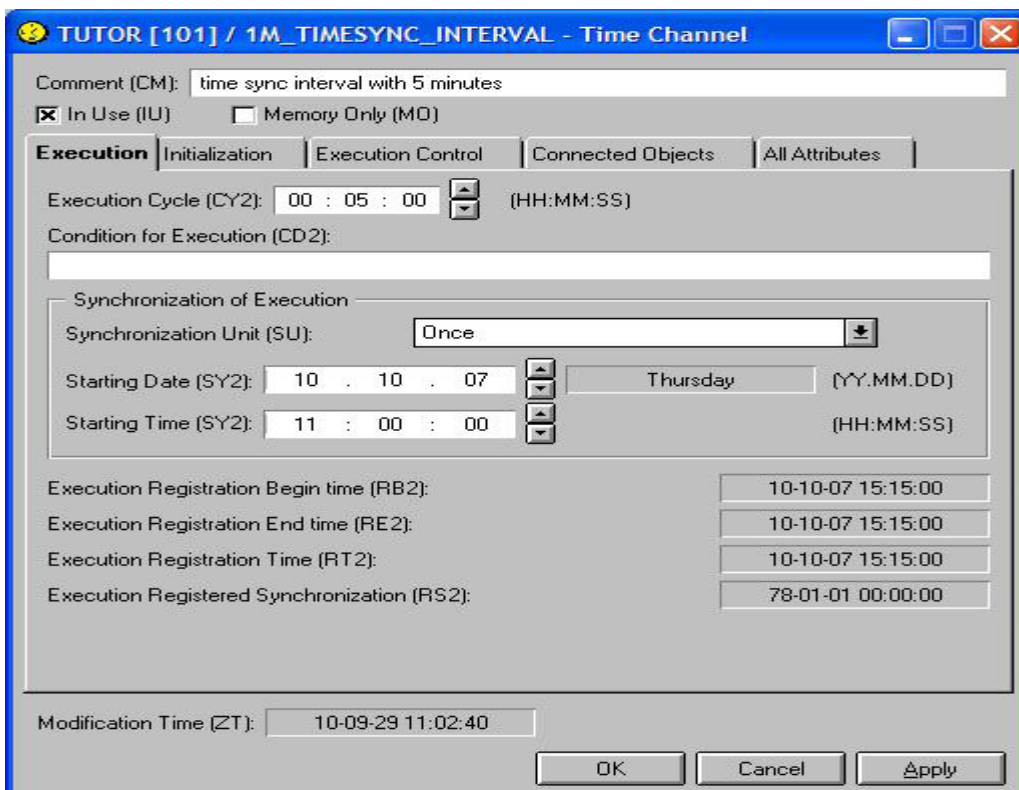


Figure 34. Time channel with 5 minute time interval for command execution.

The final station process display is seen below in figure 35.

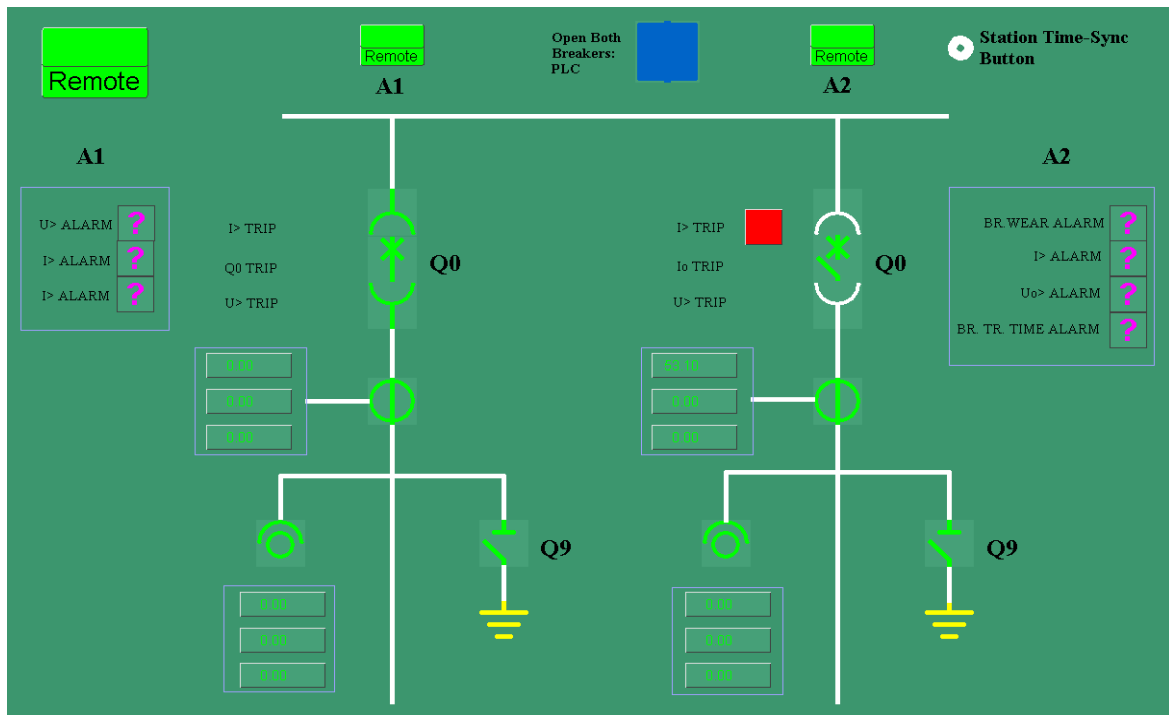


Figure 35. Process display made in MicroSCADA showing a simulated overcurrent trip on bay A2.

5.10.1 Problems with MicroSCADA

MicroSCADA won't allow the user to control the bay if the station or the bay shows *local* in the process. Due to the lack of external I/O ports on the RTU it was necessary to create a "simulated" L/R setting for the station, and that was done with the RTU PLC by copying the REF 541 relays L/R signal. After the station display had been made in MicroSCADA it showed *local* at station level and *remote* in the bay A2, even though it was exactly the same signal with no invert mode made in REF 541, PLC or MicroSCADA. It was also noticed that when the REF 541 was in remote setting the object value at MicroSCADA was 0, meaning *local*.

Firstly the L/R state read addresses were changed in CET to inverted. Afterwards it was changed in RTU PLC as a value inversion for the stations L/R and then everything was OK.

A similar problem occurred with the VAMP relay and is explained in section 5.3.1.

5.11 Results

The final result was a functioning station with devices made by two vendors in both *local* and *remote* control. Even if the remote simulation was done on the same PC as the local control, it was still over a different communication link as if it had been done from a separate distant control station.

The PLC program with both breakers opening simultaneously did not function quite as well as initially planned, as it somehow made the VAMP execute two times per command. And the REF needed a 10 second delay between commands for it to execute, otherwise a negative acknowledgement would appear.

Some initially planned functions were left out, such as VAMP latch release command and trip counters, due to process object amount limitations and lack of time for testing. But all the indications, alarms, trips, measurements, command signals and clock synchronization included in the final configuration functioned as intended.

6 Discussion and conclusion

At first when this thesis work was assigned to me I did not know what to expect or how to start, as most engineering concerning station automation and data communication was new to me. After digging through the Internet and a few books, the functionality required at substations became clearer and after that the smaller details also started to fall into place. After acquiring basic information and procedures for substation engineering the practical work could begin.

This thesis has taken quite some time to complete since I had to learn and get familiar with all the different software programs used. Almost every step had to be figured out one by one and as a new configuration was built it had to be tested for errors. A solution to one problem could present a new problem made with an earlier software, which resulted in many hours of troubleshooting. But I believe troubleshooting belongs to the nature of IEC 61850 engineering.

Once the majority of problems had been solved it was nice to see that it all started falling into place as the station began to acknowledge signals generated by process events. While the RTU 560 products family cover a whole lot of other untested functionalities in this project, it was still very satisfying to be able to prove that communication can be established with relays from different vendors to the control station via RTU 560G.

Even though IEC 61850 has the advantage of new features and wider compatibility with the latest hardware, those advancements may also be the protocol's weakness, as the advancements makes the protocol very complex and hard to fully utilize. Many of the older protocols are simpler and easier to configure, making them favorable in some situations.

If I had to go through this configuration again I would save a lot of time by knowing the way around simple errors that may occur during the test phase. I would also put more time into planning in the beginning. It would pay off well when you know exactly what to do and what to include in the different stages of software configuration.

No functions have been created or improved as communication could be established with existing methods. I hope the primary goal has been achieved meaning that this document can be used as a guide to help others that are working with similar RTU 560 setups.

I have not had the opportunity to read the specific demands of the IEC 61850 standard, but the interoperability can still be improved between products from different vendors by ensuring a better compliance with the IEC 61850 standard. The standard could also be extended to cover all the basic electrical IED functions, so that a third party software manufacturer could compete and concentrate on making the configuration process as easy and reliable as possible. An advancement in the configuration process would be to lower the amount of softwares needed, as they could be merged into one software that can handle all the functions needed.

7 List of sources

1. **ABB Substation Automation Oy.** ABB. [Online] July 31, 2001. [Cited: March 1, 2010.]
[http://library.abb.com/global/scot/scot229.nsf/veritydisplay/811733b652456305c2256db40046851e/\\$File/SPAcommprot_EN_C.pdf](http://library.abb.com/global/scot/scot229.nsf/veritydisplay/811733b652456305c2256db40046851e/$File/SPAcommprot_EN_C.pdf).
2. **ABB Substation Automation.** ABB Substaion Automation. [Online] [Cited: October 9, 2010.]
[http://www05.abb.com/global/scot/scot258.nsf/veritydisplay/c52cd496dbc40747c1257705005cd6d8/\\$File/Xf536DEABB%201596%2010%20en%20RTU560%20for%20DIN%20rail%20Flyer.pdf](http://www05.abb.com/global/scot/scot258.nsf/veritydisplay/c52cd496dbc40747c1257705005cd6d8/$File/Xf536DEABB%201596%2010%20en%20RTU560%20for%20DIN%20rail%20Flyer.pdf).
3. **Cegrell, Torsten and Sandberg, Ulf.** *Industriella Styrssystem*. Borås : Responstryck, 1994. ISBN: 91-88330-00-1.
4. **Curtis, Ken.** DNP Users Group. [Online] Mrach 20, 2005. [Cited: February 26, 2010.] <http://www.dnp.org/About/DNP3%20Primer%20Rev%20A.pdf>.
5. **Netcontrol.** Netcontrol. [Online] [Cited: October 9, 2010.]
<http://www.netcontrol.com>.
6. **Proudfoot, Douglas.** Nettedautomation. [Online] March 21, 2002. [Cited: February 2, 2010.]
<http://www.nettedautomation.com/download/UCA%20and%2061850%20for%20dummies%20V12.pdf>.
7. **Siemens Energy Automation GmbH.** SICAM 1703. [Online] 2009. [Cited: October 9, 2010.]
https://w3.energy.siemens.com/cms/00000020/en/products/substation_technology/sicam1703/Pages/ak_1703_acp.aspx.
8. Simply Modbus. [Online] [Cited: February 26, 2010.]
<http://www.simplymodbus.ca/FAQ.htm#Modbus>.
9. **Vaasa Engineering Oy.** www.veo.fi. [Online] [Cited: Mars 21, 2010.]
http://www.veo.fi/In_English/Company_presentation.
10. Wikipedia. [Online] [Cited: February 25, 2010.] <http://en.wikipedia.org>.

VEO

Responsibility for Energy

RTU 560 with IEC 61850 configuration guide

Guide

Revision 1

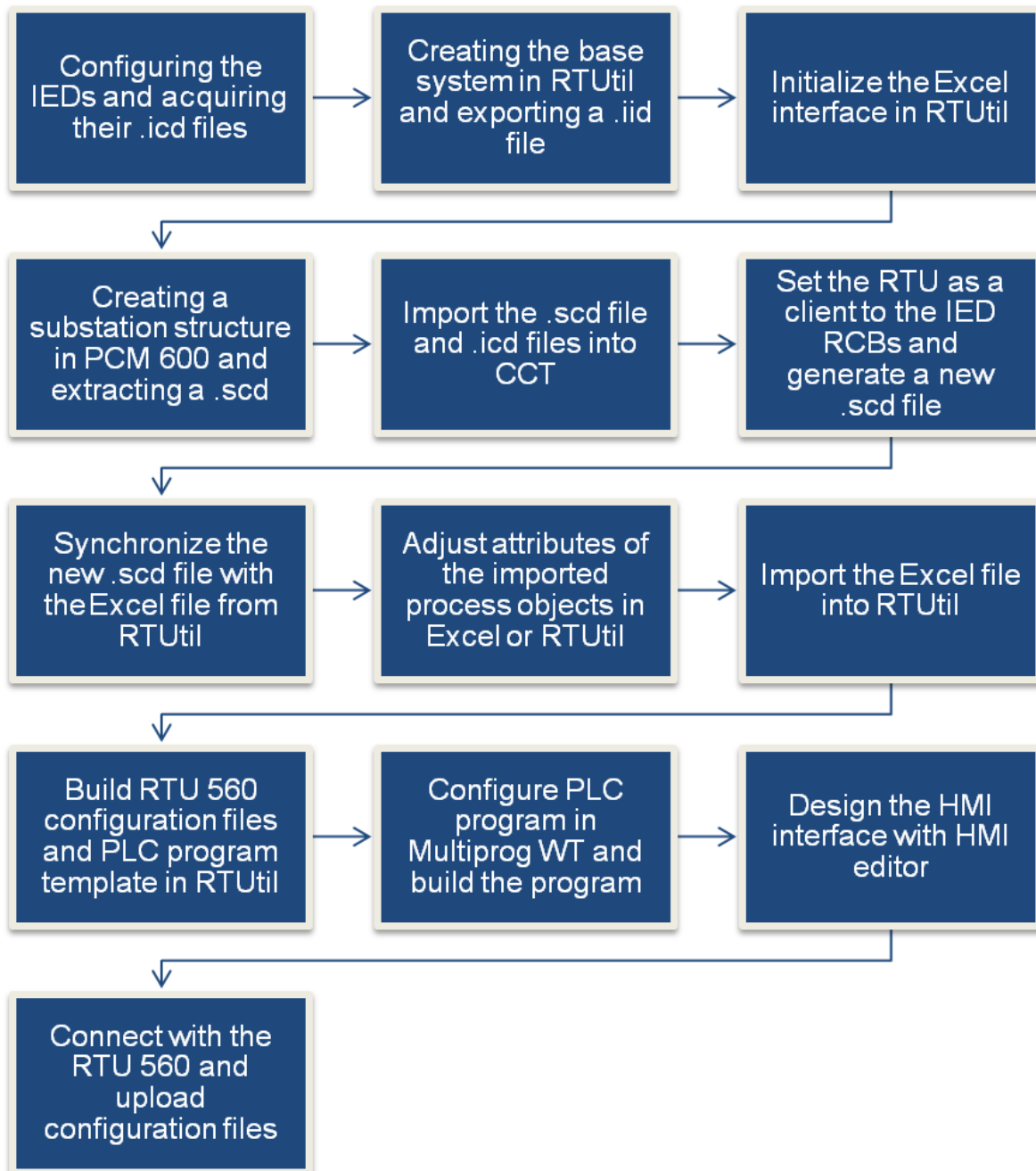
Made by: Joakim Ainasoja
1.12.2010

Contents

RTU 560 with IEC 61850 configuration guide.....	2
1 General steps of the RTU 560 configuration for IEC 61850.....	3
2 Introduction.....	4
3 VAMP IEC 61850 settings.....	4
4 SPA-ZC 400 IEC 61850 settings.....	6
5 Software needed for RTU 560 configuration (latest versions preferred):.....	7
5.1 Additional software for IEC 61850 engineering:.....	7
5.2 Additional useful software and accessories:.....	7
6 Configuration of RTU 560 in RTUtil.	8
7 PCM 600 configuration.....	15
8 CCT configuration.....	16
9 Importing IEC 61850 process objects into RTUtil.....	18
10 Multiprog WT configuration for PLC functions.....	24
11 HMI editor.....	29
12 Web interface and upload of configuration files.....	31
13 Final words.....	33

RTU 560 with IEC 61850 configuration guide

1 General steps of the RTU 560 configuration for IEC 61850



2 Introduction

This guide has been made in co-development with a thesis project made in 2010 at Vaasa Engineering Oy concerning the RTU 560 as a station RTU and gateway between IEC 61850 and IEC 60870-5-101. It covers all the configuration steps used to complete the thesis.

This document will go through all the necessary software needed to create and establish IEC 61850 communication with the RTU 560G, while also showing the important settings required in the VAMP and SPA-ZC 400 units for IEC 61850 communication with the RTU.

3 VAMP IEC 61850 settings

Firstly it is necessary to set the basics for Ethernet communication and it is done by setting the IP address and NTP server for clock synchronization.

ETHERNET PORT	
Ethernet port protocol	IEC-61850
IP port for protocol	101
IP Address	192.168.0.3
NetMask	255.255.252.0
Gateway	0.0.0.0
NTP server	192.168.0.2
IP port for setting tool	23
TCP keepalive interval	10 s
MAC address	001AD3000033
Message counter	0
Error counter	0
Timeout counter	0

Figure 1. Ethernet settings.

IEC 61850 main config	
Port	101
Check upper addresses	No
AP ID	1.1.1.999.1
AE Qualifier	12
P Selector	1
S Selector	1
T Selector	0
IED Name	SS1A1001A1
Delete dynamic datasets	-
RsrvTms included in BRCBs	Yes

Figure 2. Setting the device name.

To choose the objects to be included with the datasets, simply select Yes under the preferred dataset and also change *In use* to Yes as shown in figure 3.

IEC 61850 data map						
Index	LN	Description	Dataset 1	Dataset 2	Dataset 3	In use
90	DI32GGIO76	Digital input 32	No	No	No	No
91	DOC1PTOC12	IDir>	No	No	No	No
92	DOC2PTOC13	IDir>>	No	No	No	No
93	DOC3PTOC14	IDir>>>	No	No	No	No
94	DOC4PTOC15	IDir>>>>	No	No	No	No
95	EF1PTOC4	Io>	Yes	No	No	Yes
96	EF2PTOC5	Io>>	No	No	No	No
97	EF3PTOC6	Io>>>	No	No	No	No
98	EF4PTOC7	Io>>>>	No	No	No	No
99	EnergyMMTR1	Energy exported imported	No	No	No	No
100	fdaMMXU9	frequency demand	No	No	No	No
101	fMMXU8	frequency	No	No	No	No
102	Har2PTOC11	If2>	No	No	No	No
103	I3pdaMMXU3	IL1,IL2,IL3 demand	No	No	No	No
104	I3pMMXU1	IL1,IL2,IL3	Yes	No	No	Yes
105	I3prMMXU2	IL1,IL2,IL3 RMS	No	No	No	No
106	IArcPIOC1	I Arc	No	No	No	No
107	Io1ArcPIOC2	Io1 Arc	No	No	No	No
108	Io1MMXU11	Io1	No	No	No	No
109	Io2ArcPIOC3	Io2 Arc	No	No	No	No
110	Io2MMXU12	Io2	No	No	No	No
111	IOC1GGIO142	Fault current of Io>	No	No	No	No

Figure 3. Selecting objects into datasets.

Set the dataset(s) to be used into report control block(s) and set the report control block name. The report ID can be left at default as it is unique by default. It is recommended to use the BRCB (*Buffered Report Control Block*) for communication with RTU stations.

BRCB 1	
Dataset	DS1
Name of selected Dataset	LLN0.DS1
Report ID	Q01A1.LLN0DS1
Integrity Period	0 ms
Buffering Time	0 ms
Triggering Options	
- Data Change	Yes
- Quality Change	Yes
- Data Update	Yes
- Integrity	Yes
- General Interrogation	Yes
Optional Fields	
- Sequence Number	Yes
- Report Time Stamp	Yes
- Reason For Inclusion	Yes
- Dataset Name	Yes
- Data Reference	Yes
- Buffer Overflow	Yes
- Entry ID	Yes
- Configuration Revision	Yes
Lost reports count	0

BRCB 2	
Dataset	None
Name of selected Dataset	None
Report ID	VAMP_257.LLN0\$brcbEV201

Figure 4. Report control block settings.

And that concludes the necessary VAMP settings.

4 SPA-ZC 400 IEC 61850 settings

As with VAMP it is necessary to set name and IP addresses for the device, as well as set the IP address for the time synchronization server.

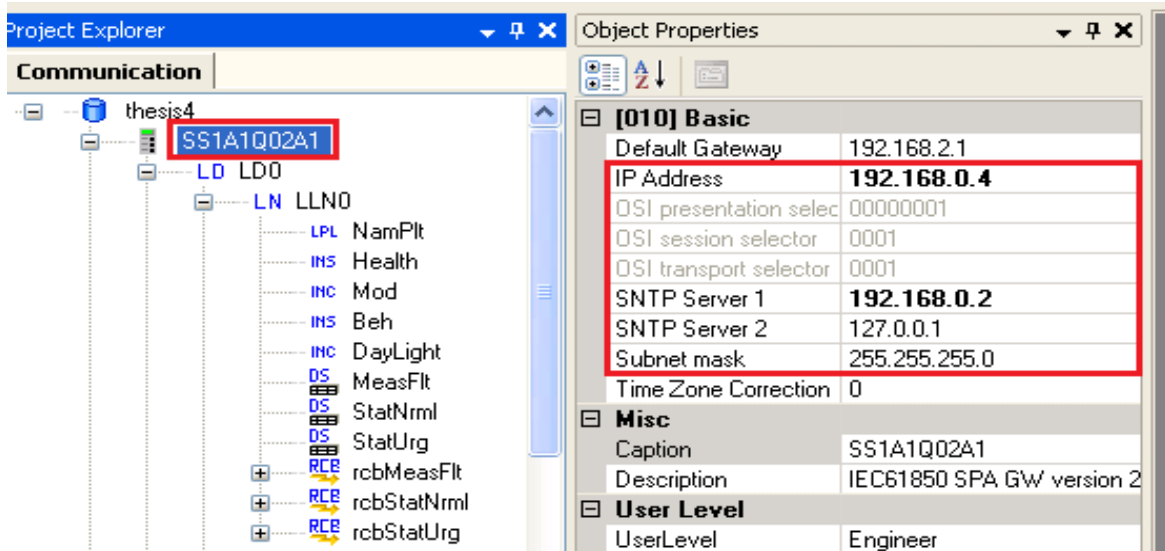


Figure 5. SPA-ZC name and IP addresses.

Make sure that the RCB attributes correspond to the desired functions. Data set should be set to true.

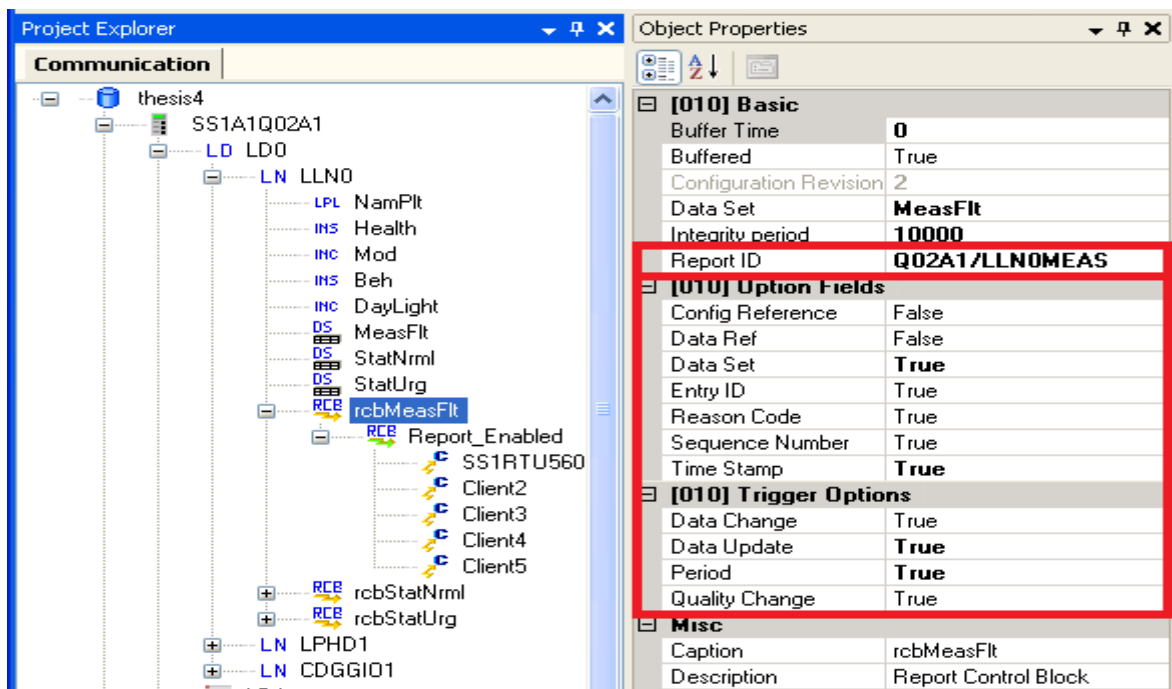


Figure 6. RCB attributes in CET tool.

If the local remote setting later on appear to be inverted in the RTU event list, it is possible to revert the state by changing *SPA Off Event Code* to 2 and *SPA On Event Code* to 1 in CET.

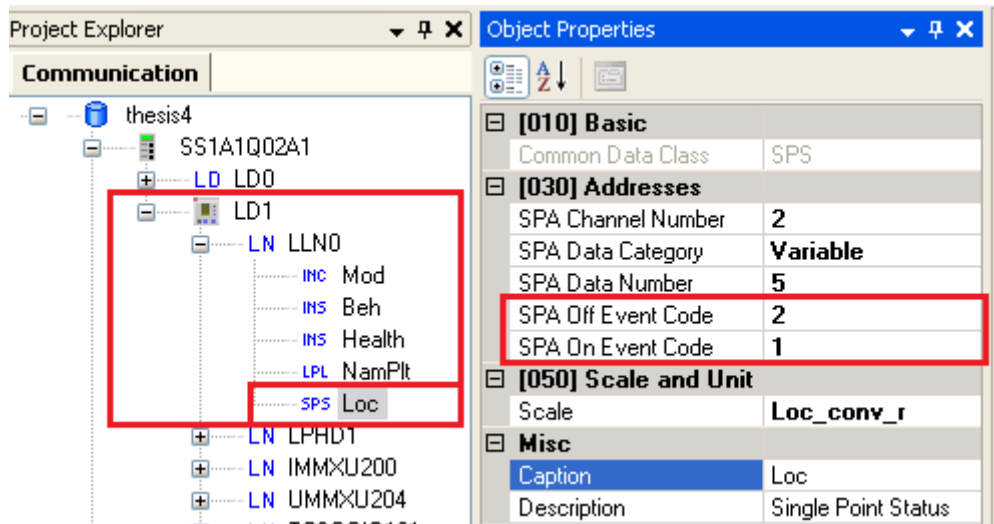


Figure 7. Inverted local/remote setting.

5 Software needed for RTU 560 configuration (latest versions preferred):

- a) RTUtil
- b) Web browser with Java plugin (e.g. Internet Explorer)
- c) Multiprog wt (for PLC functions)¹
- d) HMI editor (for integrated HMI functions)

5.1 Additional software for IEC 61850 engineering:

- e) PCM600 with necessary connectivity packages
- f) IET/CCT (comes bundled with PCM600)¹

5.2 Additional useful software and accessories:

- g) Windows Notepad (for accessing .ICD and .SCD files)
- h) Compact flash card reader for PC
- i) Microsoft Excel¹ (not required but helpful)

¹ Requires license

6 Configuration of RTU 560 in RTUtil.

Firstly it is necessary to build the project environment data.

Initialize the signal tree: Set number of levels by giving them names, if three levels are desired, leave level four empty. If the plan is to give level one the name *Runsor* which is six characters, level two the name *22kV* and three *AA3* it is then recommended that level one has 7 bytes reserved for characters (name + space), level 2 should have 5 bytes reserved and level 3 should have 4 bytes.

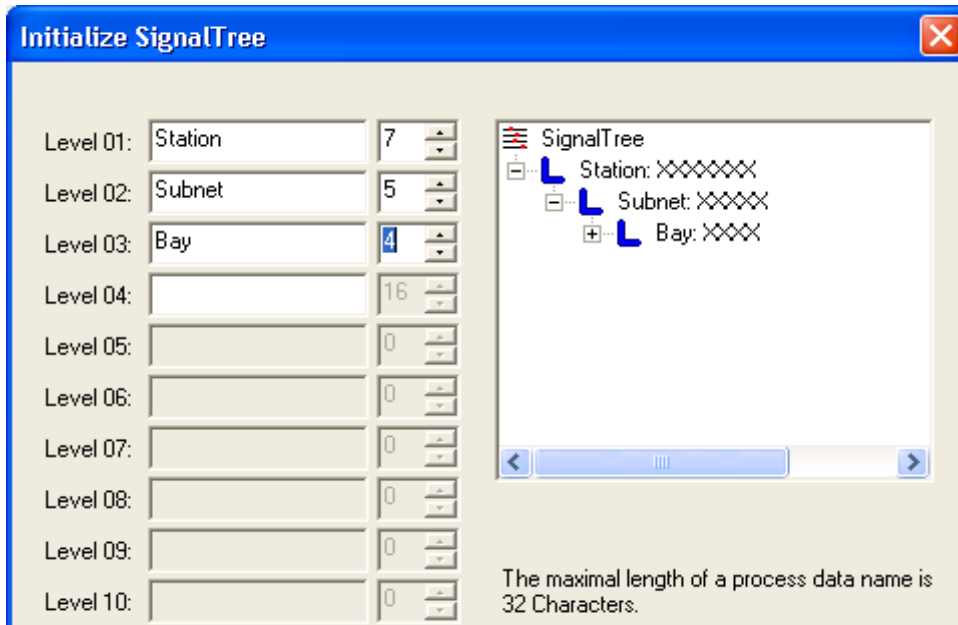


Figure 8. Signal tree initialization.

Level four in the example above has 16 characters which is left for the unique part of the object name, e.g. Q0 Status. Q0_status could then have the full object name of: *Runsor 22kV AA3 Q0_status*.

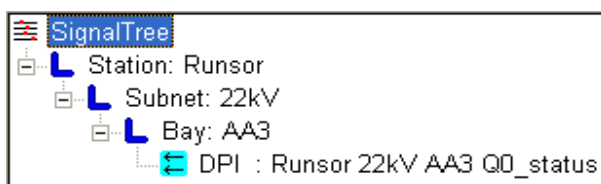


Figure 9. Example of object in signal tree.

Then build up the tree structures (Network, Hardware and Signal tree).

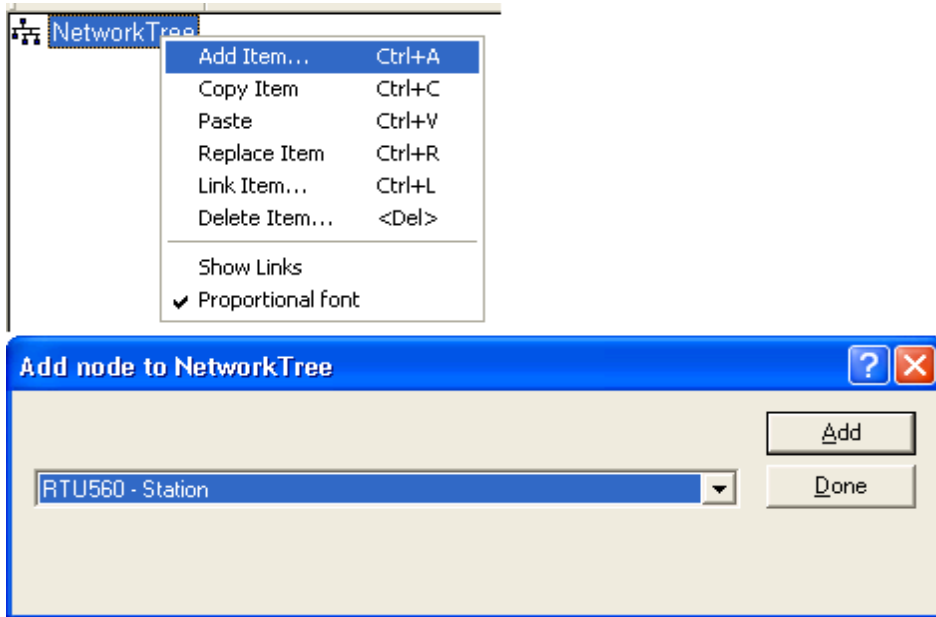


Figure 10. Adding items to the network tree.

Then the final network tree might look something like figure 11.

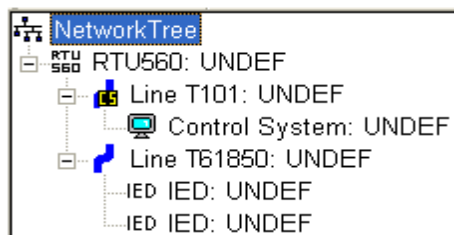


Figure 11. Network tree

Afterwards the Station RTU and IEDs need to be linked to the hardware tree.

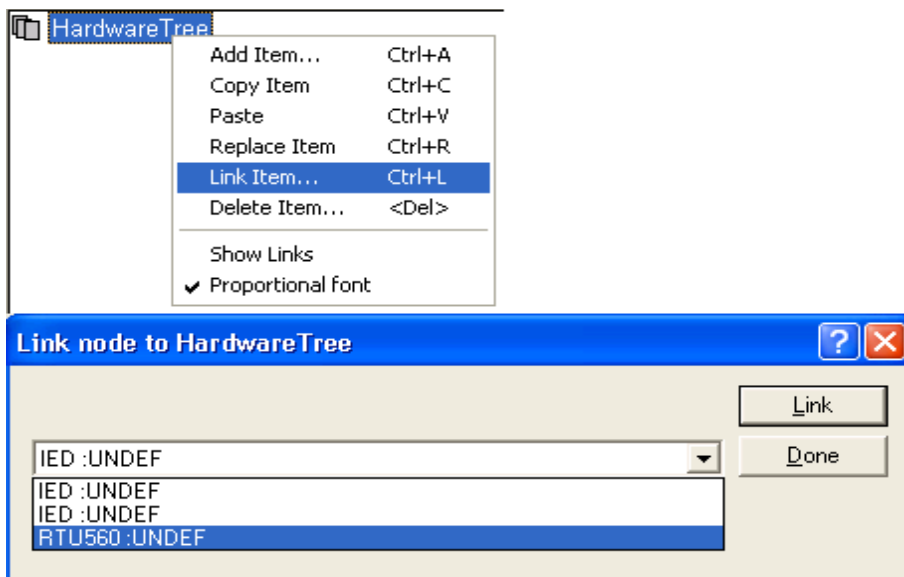


Figure 12. Linking items into the hardware tree.

Notice that all items that are linked to another tree get a small red circle at the item icon indicating that it has been linked. This will also happen to the icon in the tree that the item is linked from. Items to be linked between the trees are signal objects, IEDs, Hardware objects and communication lines.

Build up the hardware tree by selecting items that are meant to be used in the station like type of RTU, I/O boards, PLC, HMI, archive etc. Another important thing to add is an I/O bus to the RTU main CMU, as this handles the internal communication. Signals may be added to the signal tree and linked to their corresponding hardware. IEC 61850 objects should not be added at this point as they will be imported later.

Signals added to the PLC can be used as “simulated” signals that are not triggered/changed by process events, instead they can be “manipulated” and triggered by the PLC program. They can be set to communication lines so that sub devices or control stations recognize them as process events.

Once the hardware tree is built, names and attributes can be set for all items. The result may look like in figure 13.

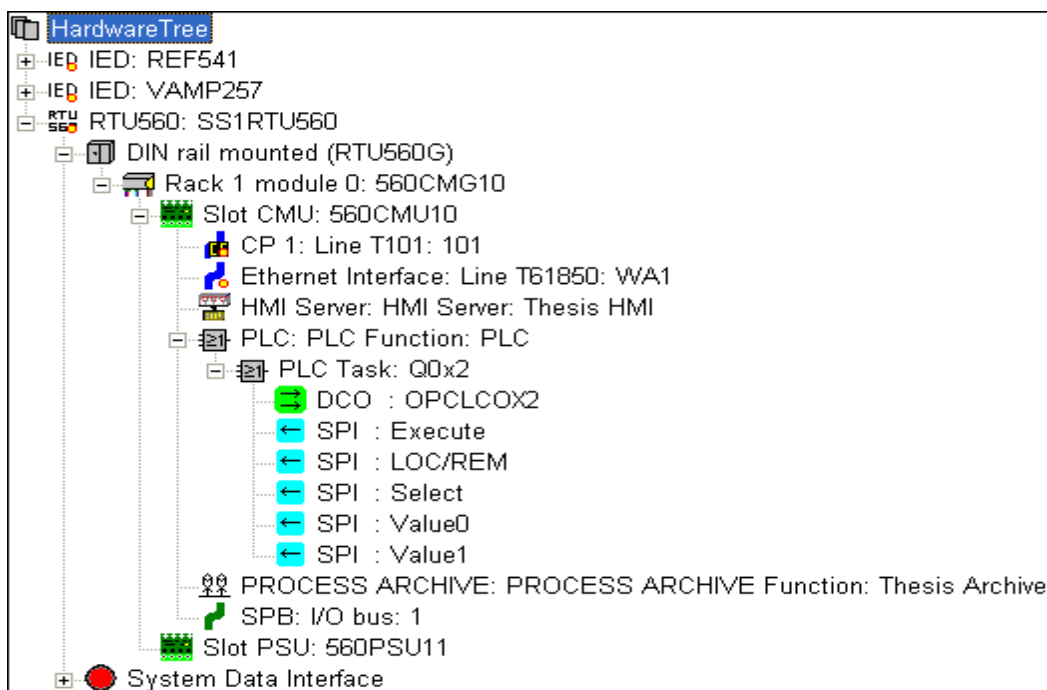


Figure 13. Hardware tree made in the thesis project.

Certain settings worth checking are the time master for the RTU, which can be found under network tree -> RTU -> parameter tab, and the time synchronization lost option which should also be set (found under the same tab).

In the hardware tree select RTU -> main CMU board -> Ethernet tab and set IP address, also set the RTU as SNTP time server if sub devices should be time synced by the RTU over Ethernet. If broadcast is selected and a time interval is set, the RTU will make a cyclic synchronization with selected interval of the sub devices. The RTU can also be set as SNTP client under this same settings tab, but it will then be necessary to select time master to sntp_ under the RTU parameter tab, as mentioned above.

When all required settings are made it is time to do a consistency check. The consistency check will reveal any direct error or if some items have the same name, address etc. It is recommended to do a consistency check regularly as the project is being built so that mistakes can be detected early.

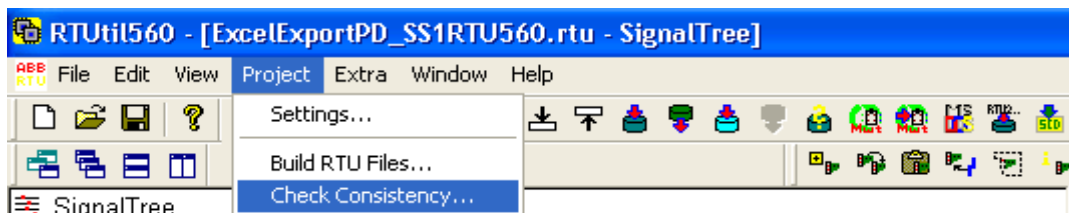


Figure 14. Consistency check menu.

If no errors appear it is time to export the Excel pattern files. Two Excel files are exported, one with the process data (often contains the letters PD in the filename) and the other for pattern data. These Excel files will contain all project data and can be modified manually by simply typing or selecting the required settings. The IEC 61850 process objects are also later imported to the PD file. If IEC 61850 is to be used it is recommended to leave the Excel file unmodified until the objects are imported. Make a backup copy of the exported Excel files.

When the Excel files are exported the Excel initialization should be made. This is done by selecting *Settings* in the *Project* menu.

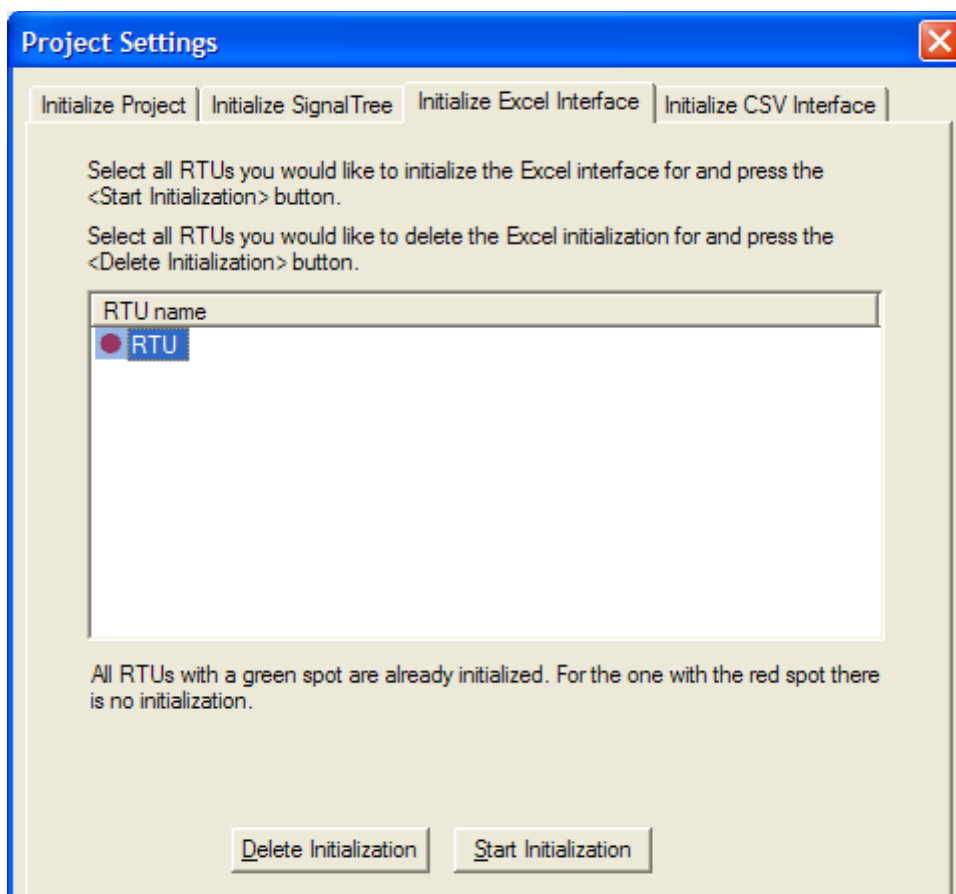


Figure 15. Initialize Excel interface tab in project settings.

Then select the exported PD Excel file by browsing it. When the Excel file has been selected it should open in the Excel software (if installed) for the user to view the default names. Then select the correct unique column identifiers for all the process data fields (on line 5 unless the Excel file has been modified).

RTU:

Give the Excel sheets with the datapoints for the listed sublines and the RTU560.

Local I/O and sublines	Excel sheet	Column ID row
SS1RTU560	Local IO	5
WA1	WA1	5

Figure 16. Selecting the Excel file sheets for the corresponding hardware.

	R	S	T	U	V	W	X	Y	Z	AA	AB
1											
2		IED name	Logical Device Instance Name	Logical Node Prefix	Logical Node Class	Logical Node Instance	Signal Data Object Name	Signal Common Data Class	Signal Data Attribute Name	Signal Data Type	Signal Function Code
3		IEC61850 Address									
4		ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
5		CNSS1IN	CNSS1LD	CNSS1LNP	CNSS1LNC	CNSS1LNI	CNSS1SDN	CNSS1CDC	CNSS1SAN	CNSS1SDT	CNSS1SFC
6											
7											
8											

Figure 17. Excel view of the IEC 61850 address columns with unique column identifier names on row 5.

Then select the Excel columns representing the corresponding parameters for all existing communication lines.

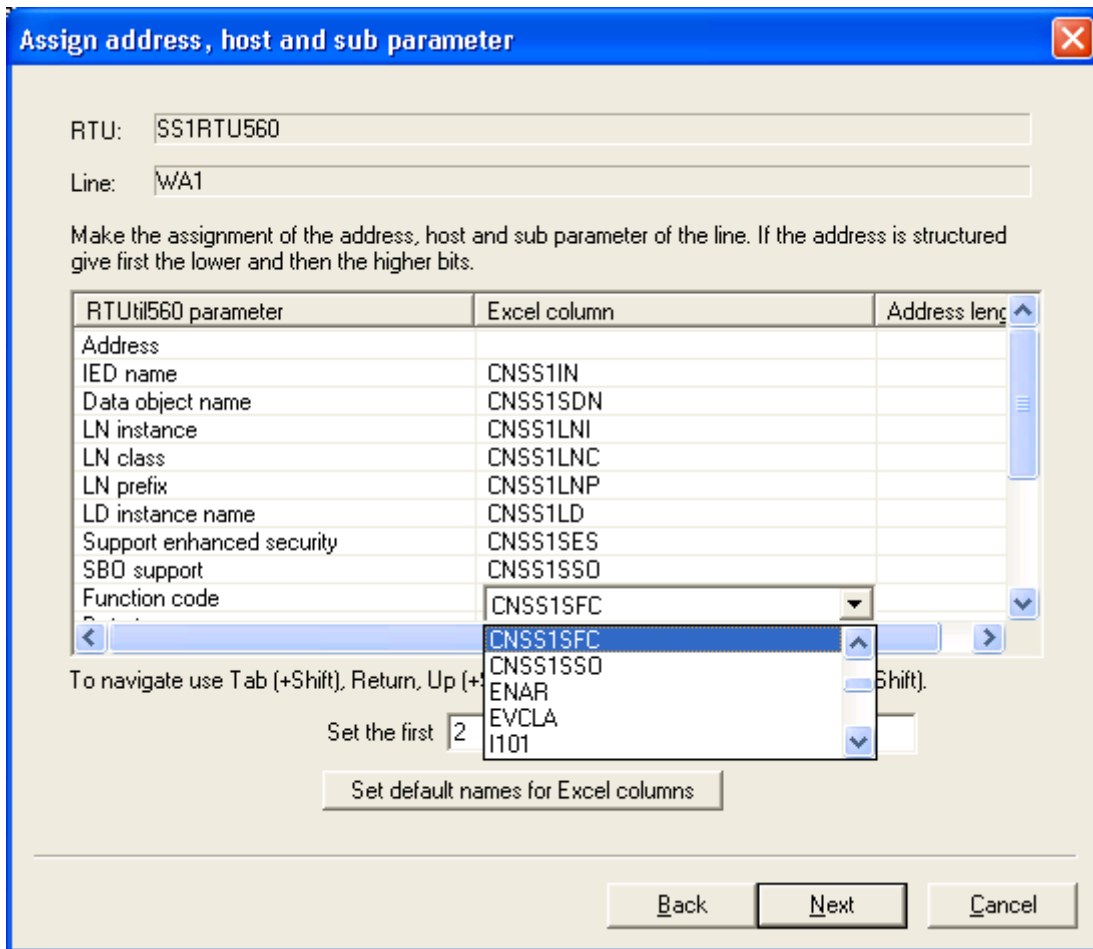


Figure 18. Selection of columns for the corresponding parameters.

A secondary faster method of selecting these columns is to use the “set the first ___ characters of the default name to: _____” by checking first how many letters are equal in the columns. Then the amount of equal letters is set to the first “box” and then writing the string of letters in the other “box”. Afterwards click the *set default names for Excel columns* so that all selectable columns receive their unique identifier names automatically. Example in figure 19.

For some of the lines it is not necessary to type anything in the “boxes” as the correct column names appear by default.

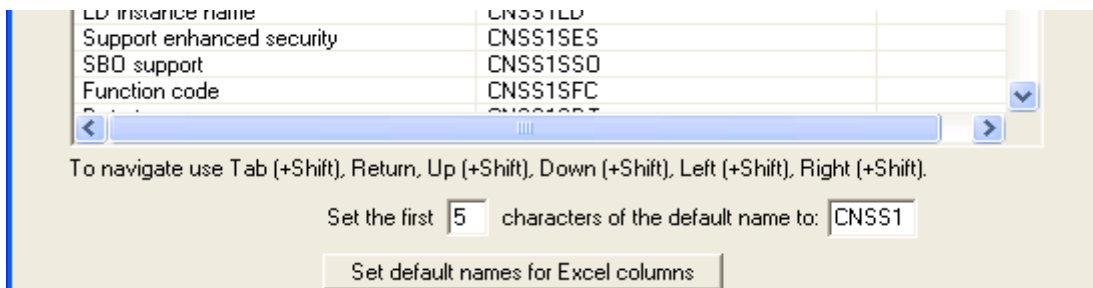
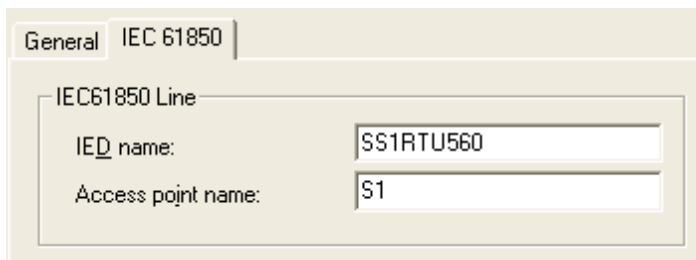


Figure 19. Automatic insertion of names for the columns.

This method should be used for all the lines and items where applicable. Once the initialization is done a green ball should appear before the RTU initially selected in figure 15.

Export a .iid file from RTUtil and rename it to .icd. This file is then needed in the CCT tool. At IEC 61850 sub line, IED name and access point name should be set before the .iid export.



The image shows a software dialog box with a tabbed interface. The 'General' tab is active, and the 'IEC 61850' sub-line is selected. Within this sub-line, the 'IEC61850 Line' section is expanded, revealing two input fields. The first field, labeled 'IED name:', contains the text 'SS1RTU560'. The second field, labeled 'Access point name:', contains the text 'S1'.

Figure 20. Example of IED name and access point name for the RTU on a IEC 61850 line.

7 PCM 600 configuration

Not many steps are needed with the PCM 600 tool for IEC 61850 engineering with RTU 560. Basically it can be used just for setting up a substation structure with generic IEC 61850 IEDs. One extra IED may be added under a bay to act as the RTU later in CCT. IP addresses may be added and technical keys need to be set for all the IEDs. The technical key of the RTU should be the same as set in RTUtil. There is no need to add the same technical keys to the substation and the voltage level as in the RTUtil signal tree.

The screenshot displays the PCM 600 tool interface. On the left, the 'Plant Structure' tree shows a project named 'THEISISX' containing a 'Substation' with a 'Voltage Level' and two 'Bay' objects. Each bay contains an 'IEC61850 IED'. On the right, the 'Object Properties' window is open, showing the configuration for one of these IEDs. The properties are organized into several sections:

- [020] Addresses:** IP Address (192.168.0.2), IP-GATEWAY (192.168.0.2), IP-SUBNET (255.255.255.0), OSI ACSE AE Qualifier (23), OSI ACSE AP Title Value (1,3,9999,23), OSI Presentation Selector (00000001), OSI Session Selector (0001), OSI Transport Selector (0001).
- [030] Communication Control:** Configuration Revision Check Local (True), Dynamically Create Data Sets (False), Enable EntryID Check (False), MMS Request Timeout (5000), Report Control Block Initialize (False), Use 32 Bit Entry ID (False), Use Sequence Number Check (True).
- [040] Polling:** Polling Timeout (120).
- [060] Control Authorization:** (This section is highlighted with a dashed border in the image).
- [070] OPC Alarm and Event:**
- [080] Authentication:**
- Misc:** Always query DR directory from IED (False), Auto. delete recordings in IED after (False), Caption (IEC61850 IED), Description (Description of IEC61850 IED), Manufacturer (empty), Path for disturbance records in IED (empty), Re-upload already uploaded disturbance (False), Technical Key (SS1RTU560), Use LD name as prefix to DR file name (False).
- SCL Information:** Confirmation Version (empty).

The 'Object Properties' window also shows a scrollable list of properties at the bottom, with '[060] Control Authorization' currently selected.

Figure 21. RTU 560 settings in the PCM 600 tool.

Then simply export a .scd file that should later be imported into CCT.

8 CCT configuration

Build a new project and import the .scl file generated from PCM 600 so that a substation structure will be built. The IEDs appear empty, therefore the IEDs .icd files should be imported by right-clicking them and selecting import.

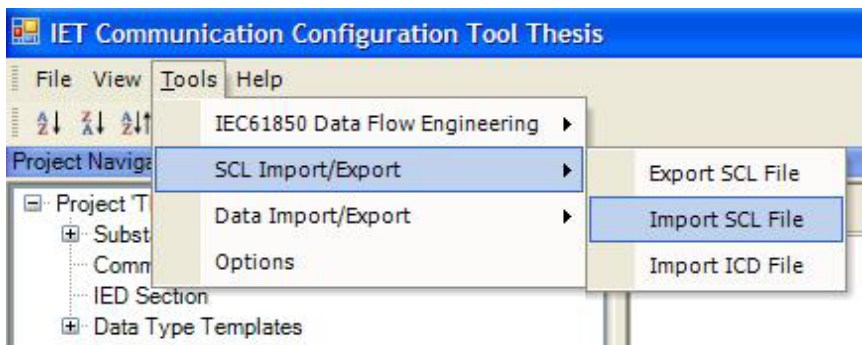


Figure 22. Menu for importing the .scl file.

During the thesis project it was noticed that regularly generated .icd files would not import properly into the used version of CCT. However by generating .iid files from the relays and then renaming those files to .icd (as done with the RTU .iid) it was possible to import data to the IEDs in CCT.

When the IED data has been imported the RTU should be set as a client to the report control blocks (RCBs). That is done by selecting the same bus connection for the IEDs. See figure 23 for an example of the procedure. Also set the IP addresses for the IEDs.

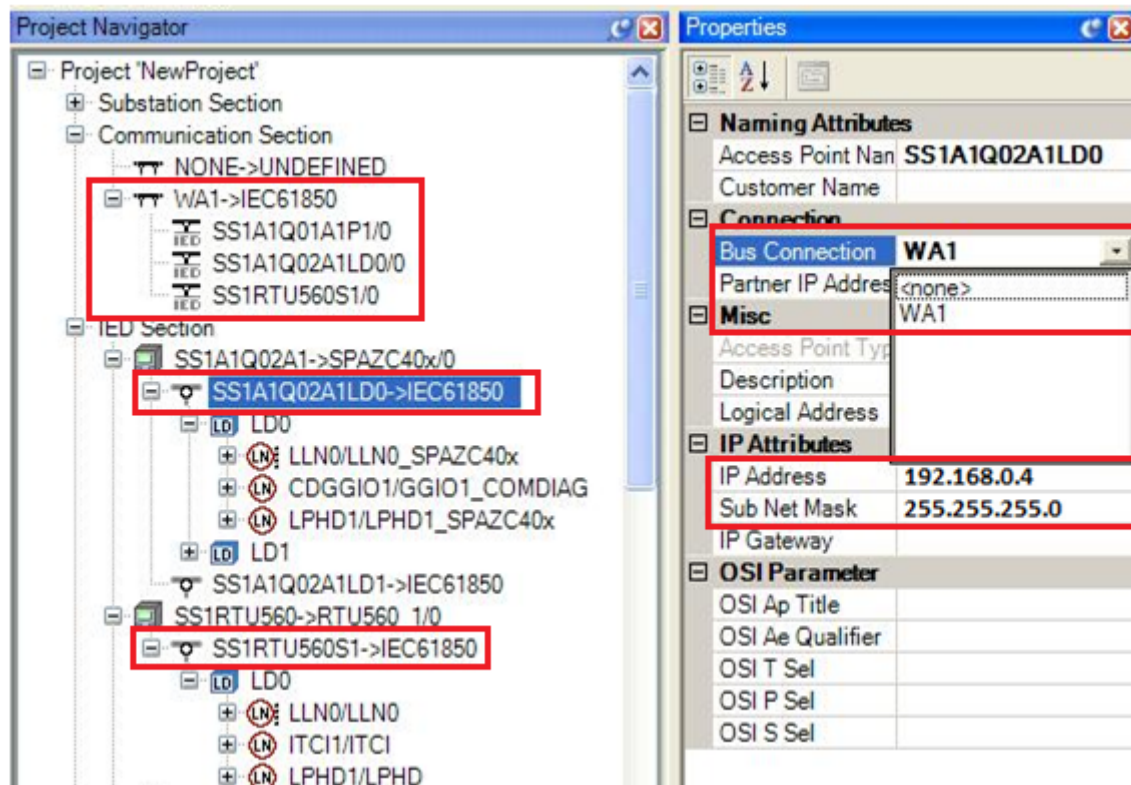


Figure 23. Selecting the bus connection for a relay in CCT.

When the bus connection has been selected it is important to select *Update DataFlow* from the Tools menu as illustrated in figure 24.

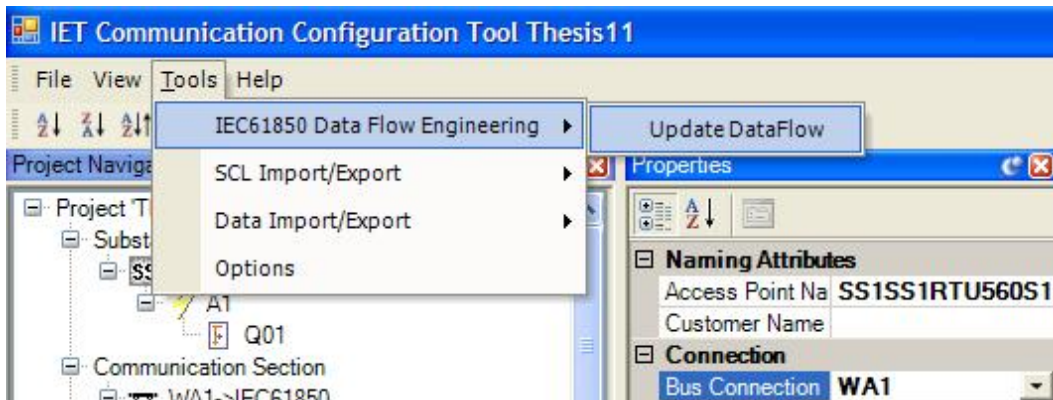


Figure 24. Selecting update dataflow in CCT.

Now the RTU should appear as a client to the RCBs from the relays.

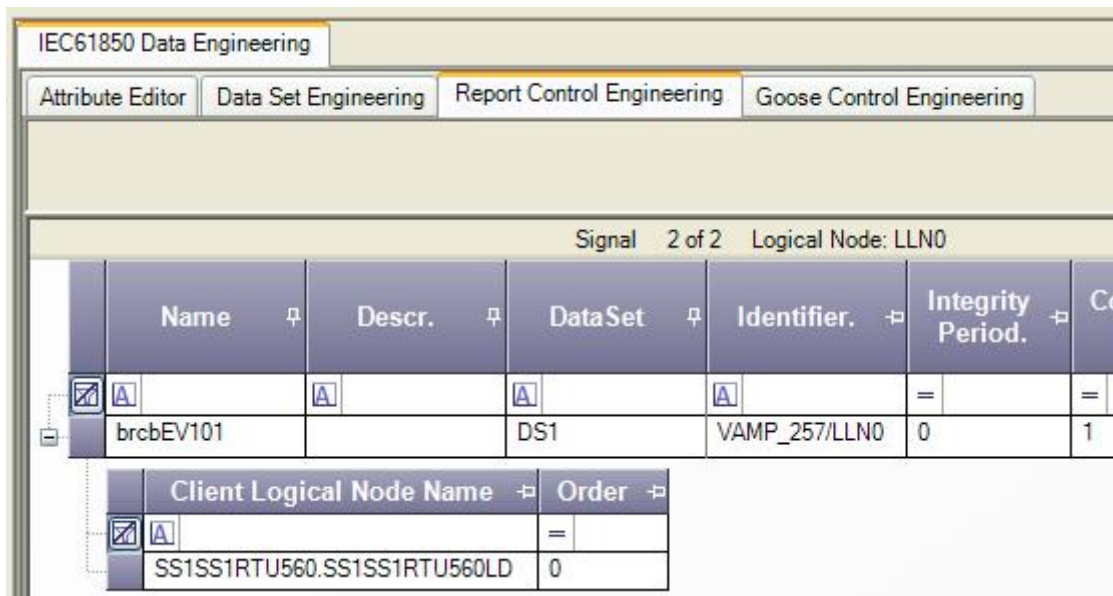


Figure 25. RTU 560 as client to a VAMP RCB.

Another crucial error noticed in the thesis project was that the unused RCBs from the VAMP relay was included at the CCT report control engineering tab, and if the unused RCB was not removed, the RTU 560G would end up in an endless booting loop making it inaccessible for configuration file removal. The only way to reset it to default was to take out the compact flash card and then directly remove the configuration files with a PC. The removal of unnecessary RCBs in CCT can be seen in figure 26.

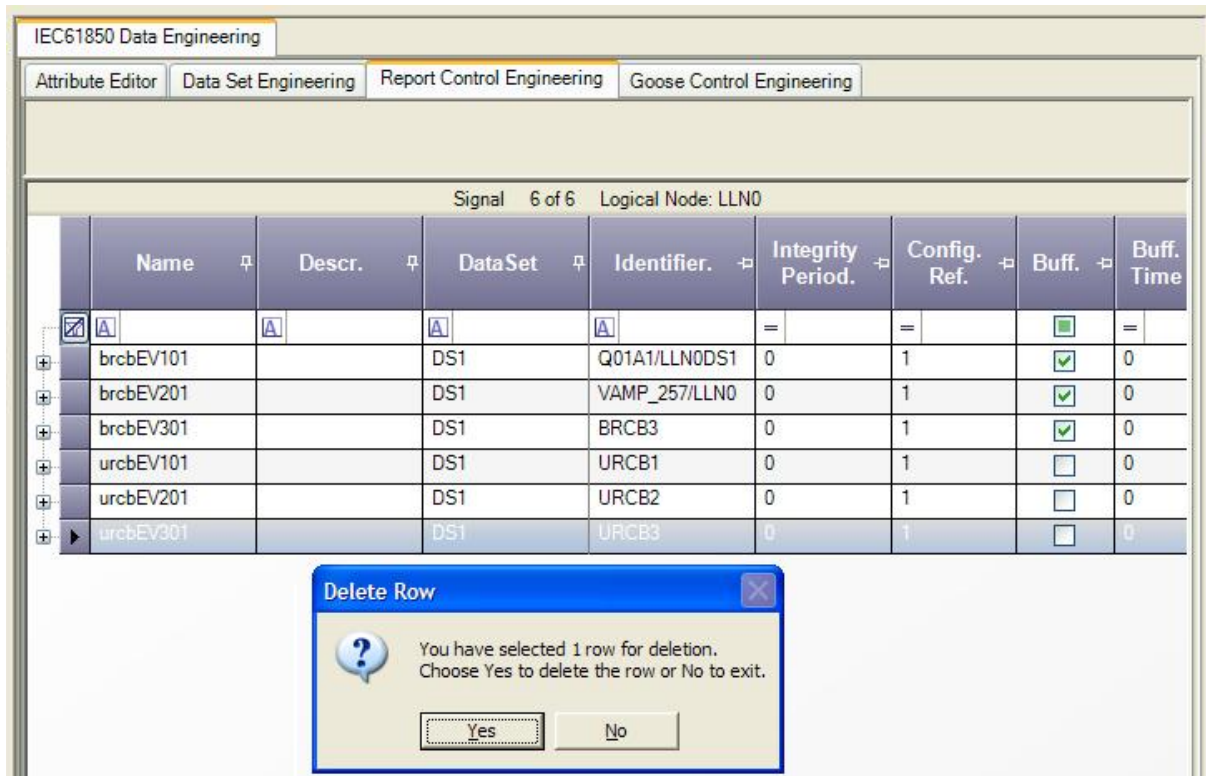


Figure 26. Removal of unnecessary RCBs in CCT.

Now when the settings are made a new .scd file should be exported.

9 Importing IEC 61850 process objects into RTUtil

The new .scd file from CCT should then be synchronized with the previously exported Excel file. **Make sure that a backup copy has been made of the Excel file** (in case any changes need to be made or errors occur) as the Excel file will be overwritten containing new data.

The synchronization is done by selecting SCD import in RTUtil.

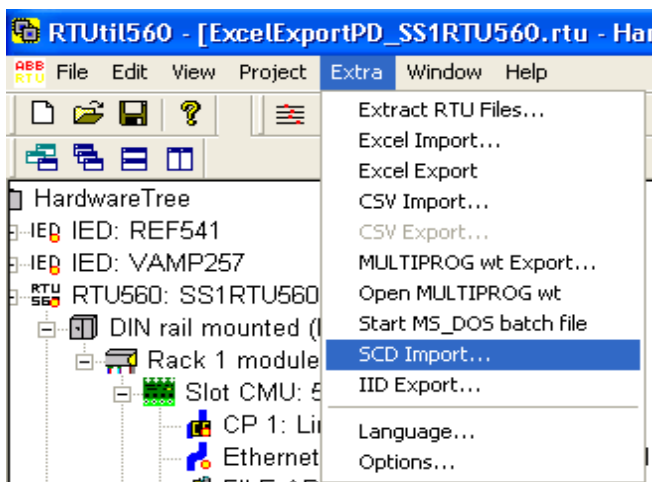


Figure 27. SCD import in RTUtil.

Select the .scd file for import and then select the Excel file containing the process data (usually named *ExcelExportPD_<RTU name>.xls*)

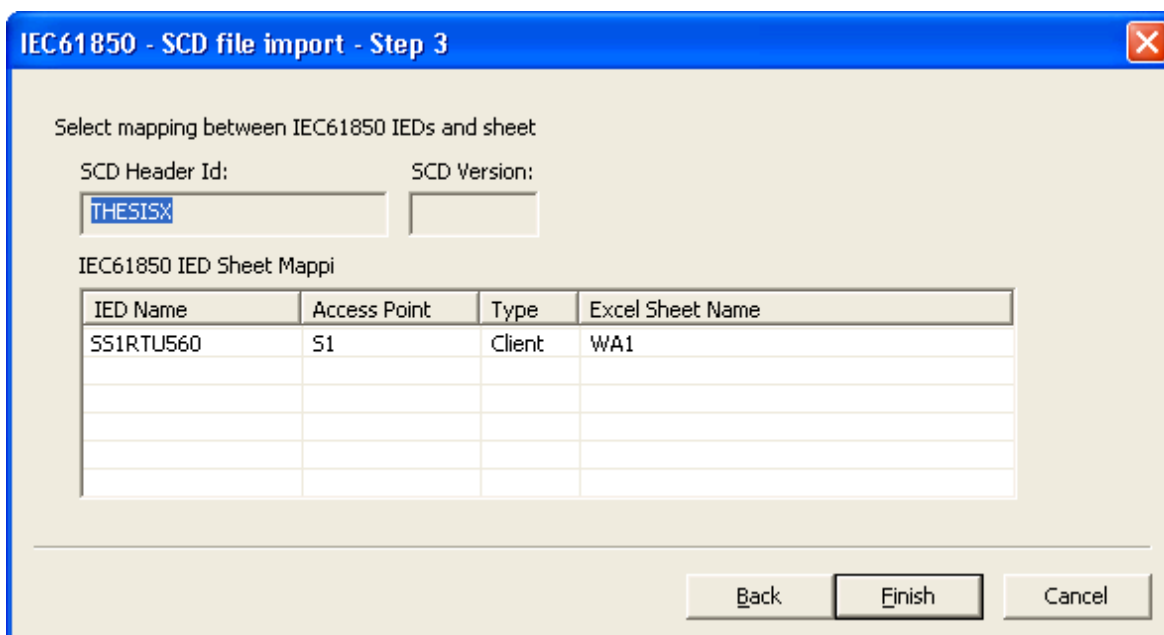


Figure 28. Step 3 of SCD import in RTUtil.

When the import is done the IEC 61850 process object appear in the Excel file as one signal per row.

IED name	Logical Device Instance Name	Logical Node Prefix	Logical Node Class	Logical Node Instance	Signal Data Object Name	Signal Common Data Class	Signal Data Attribute Name	Signal Data Type	Signal Function Code
IEC61850 Address									
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII
CNSS1IN	CNSS1LD	CNSS1LNP	CNSS1LNC	CNSS1LNI	CNSS1SDN	CNSS1CDC	CNSS1SAN	CNSS1SDT	CNSS1SFC
SS1A1Q02	LD1	RES	PTOV	44	Op	ACT	general	BOOLEAN	ST
SS1A1Q02	LD1	ESW	XSWI	129	BlkCls	SPC	stVal	BOOLEAN	ST
SS1A1Q02	LD1	ESW	XSWI	128	BlkCls	SPC	stVal	BOOLEAN	ST
SS1A1Q01	Relay	Obj1	CSWI	1	Pos	DPC	Oper.ctfVal	BOOLEAN	CO
SS1A1Q02	LD1	SPRC	GGIO	190	Alm1	SPS	stVal	BOOLEAN	ST
SS1A1Q02	LD1	PH	PTOC	31	Op	ACT	general	BOOLEAN	ST
SS1A1Q02	LD1	DEF	PTOC	40	Str	ACD	general	BOOLEAN	ST
SS1A1Q02	LD1	ESW	CSWI	129	Pos	DPC	stVal	Dbpos	ST
SS1A1Q02	LD1	CB	XCBR	120	BlkOpn	SPC	stVal	BOOLEAN	ST
SS1A1Q01	Relay	EF1	PTOC	4	Str	ACD	dirGeneral	Enum	ST
SS1A1Q02	LD1	I	MMXU	200	A.phsC	CMV	cVal.mag.f	FLOAT32	MIX
SS1A1Q02	LD1	SPRC	GGIO	190	Alm	SPS	stVal	BOOLEAN	ST
SS1A1Q02	LD1	PH	PTOC	32	Str	ACD	general	BOOLEAN	ST
SS1A1Q02	LD1	ESW	XSWI	130	BlkCls	SPC	stVal	BOOLEAN	ST
SS1A1Q01	Relay	Obj1	CSWI	1	Pos	DPC	stVal	Dbpos	ST
SS1A1Q01	Relay	V01	GRIO	97	Ind	SPS	stVal	BOOLEAN	ST

Figure 29. Example of imported process objects in an Excel file.

Excel has great functions for sorting data and that can be used to sort the process objects and make them easier to find within the Excel file. To do so simply select all the rows containing IEC 61850 objects and select *Sort..* from the Data menu.



Figure 30. Sorting of rows by columns in Excel.

The object unique identifiers can then be written into Excel as the objects can be identified by cross referencing them with the objects in the relay configuration tool. This task is time consuming but needs to be done. The object identifiers can also later be added in RTUtil if not in Excel.

17	WT1	SS1A1901	relay	130	SS1A1901	1	ApriSL	DMY	CYALING1	FLUR102	BA
18	DCO	SS1A1901	Relay	Obj1	CSM	1	Pos	DPC	Oper.ctlVal	BOOLEAN	CO
19	rev	SS1A1901	Relay	Obj1	CSM	1	Pos	DPC	Oper.ctlVal	BOOLEAN	CO
142	MotStGGI027	Motor starting				No	No	No	No	No	No
143	Obj1CSW1	Object 1				Yes	No	No	No	Yes	
144	Obj1RSVM1	Synchrocheck object 1				No	No	No	No	No	

Figure 31. Object for VAMP breaker control (Excel above and Vampset below).

Station	Subnet	Bay	SCADA object
Process Object Identification			
8 ASCII	8 ASCII	8 ASCII	8 ASCII
O/I01	O/I02	O/I03	O/I04
SS1	A2	Q02A1	Q0CONTR
SS1	A2	Q02A1	Q0STATUS
SS1	A2	Q02A1	Q0LOCLOW

Figure 32. Example of process object identifiers, the last column is the unique name.

Then the Excel file should be imported to RTUtil (note that a new RTUtil project will be made and get the same name as the Excel file once the import is done). Note that the third column (RTUtil560 import) in Excel must be set to Y (=yes) for all the objects to be imported.

An error that occurred in the thesis project was when the access point name were different in RTUtil compared to the name inside the .scd file.

The access point name can also be viewed in the synchronized Excel file by opening the file in Excel -> right-clicking one of the sheet tabs at the bottom -> select *View Code* -> select *(IEC61850files)* -> set *xlSheetVisible* -> go back to Excel and view the new tab at the bottom of the Excel window.

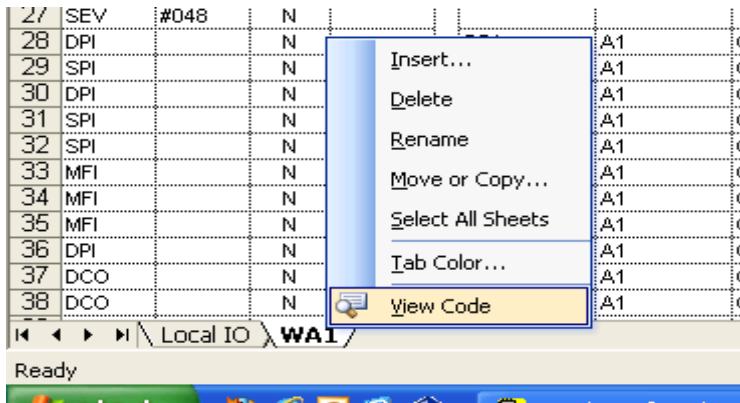


Figure 33. Selecting view code at sheet tabs in Excel.

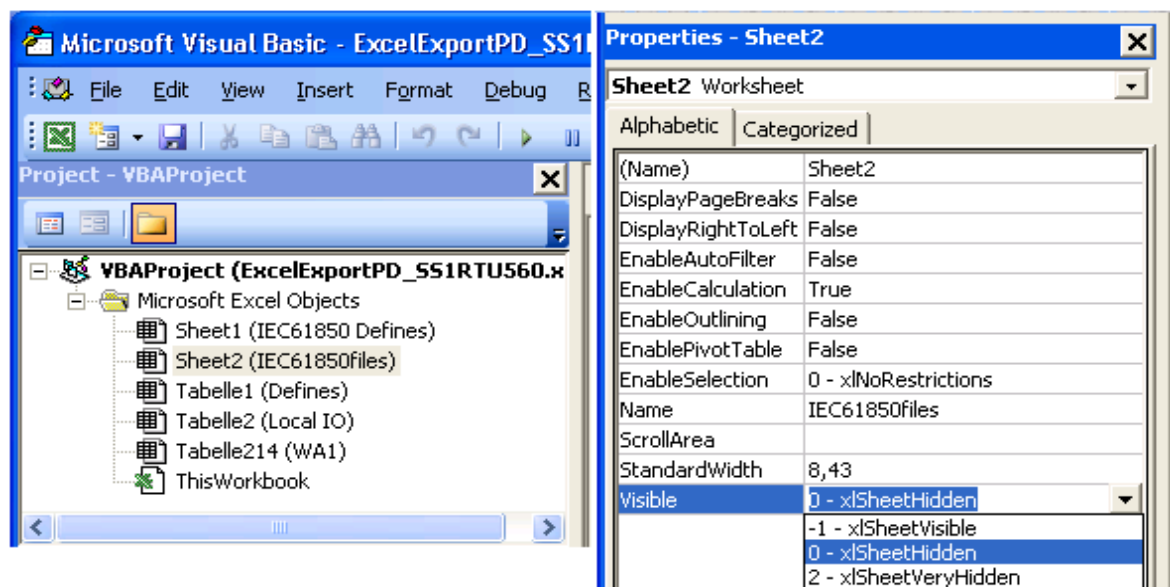


Figure 34. Selecting xlSheetVisible.

	A	B	C	D	E	F	G	H
1	Sheet Name	TNO Line Id	ITCI IED Name	ITCI Access Point	SCD File Header	SCD File Name	SCD File Time	SCD File Size
2	WA1	6	SS1RTU560	S1	THESISX	SS1X.scd	D2010-08-20 07:11	127184 Bytes

Figure 35. Locating the access point name in Excel.

When the Excel file has been imported the process objects should appear in the signal tree and if the IEDs have already been defined they should automatically be linked. Otherwise they should be linked manually to the corresponding IED.

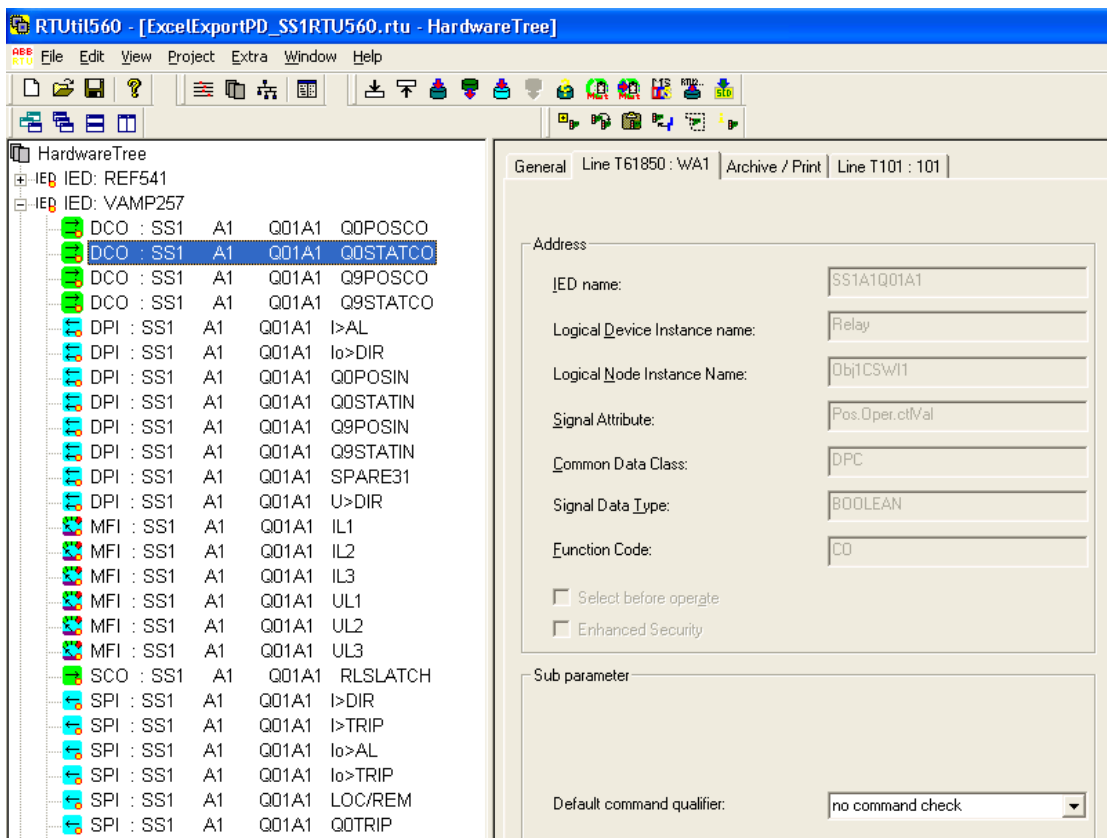


Figure 36. Process objects in RTUtil.

Now objects meant to be used in the PLC function can be assigned to the desired PLC task. See figure 38 for an example.

Archive function can be set for all objects to show up in the RTU event list, both in Web interface and HMI.

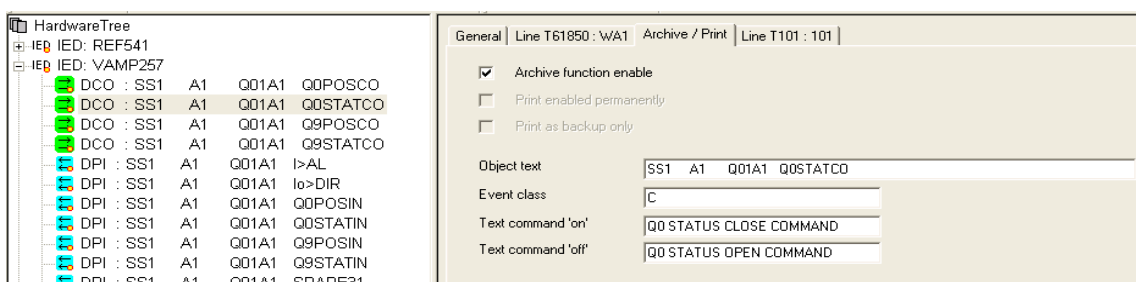


Figure 37. Example of archive function for Q0_status object.

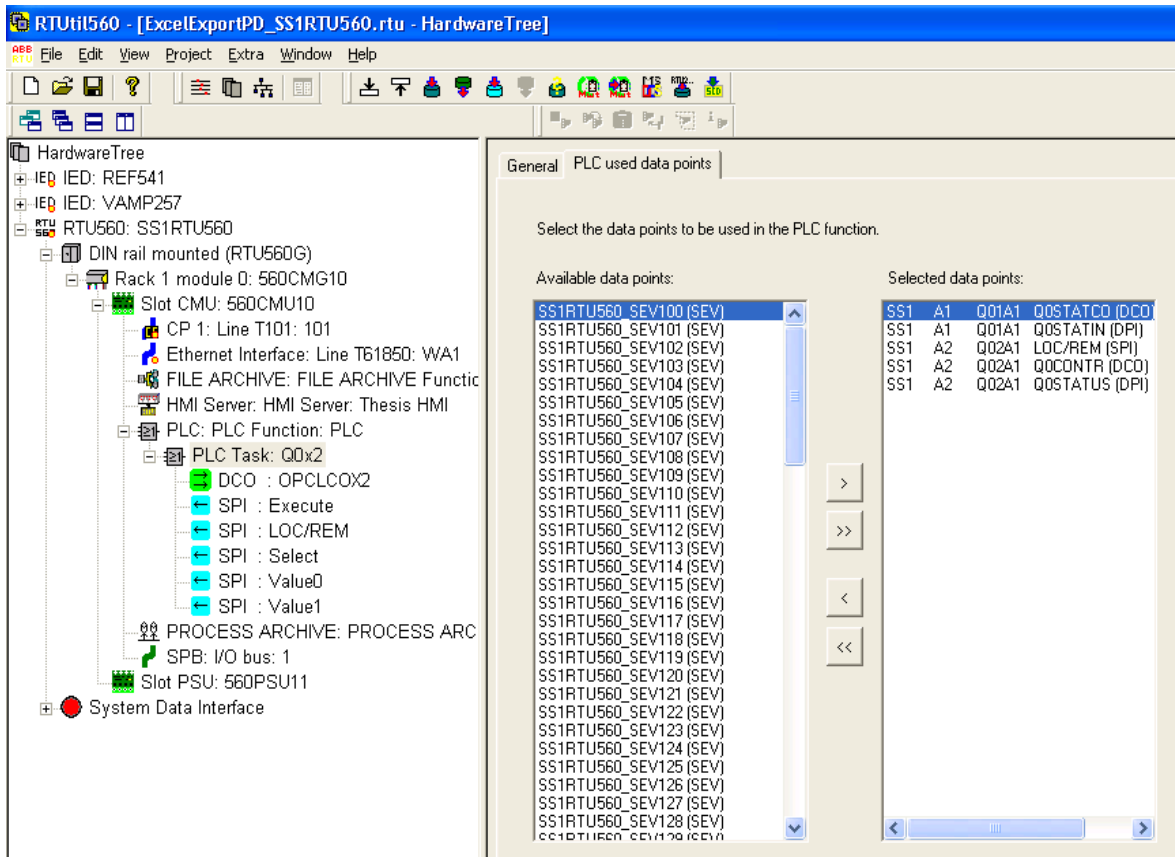


Figure 37. Selected data points for PLC task Q0x2.

Now that all objects are inserted and the attributes are set, the RTU configuration files can be built. Four files will be built (.iod, .gcd, .ptx and .oad file formats) and all these files can later be uploaded to the RTU.

A Multiprog WT export should be made so that the PLC program can be made with MWT (Multiprog WT).

10 Multiprog WT configuration for PLC functions

Multiprog WT can be launched from RTUtil along with an exported RTU560 template project. Firstly it is important to do an import of process data addresses, see figure 38.

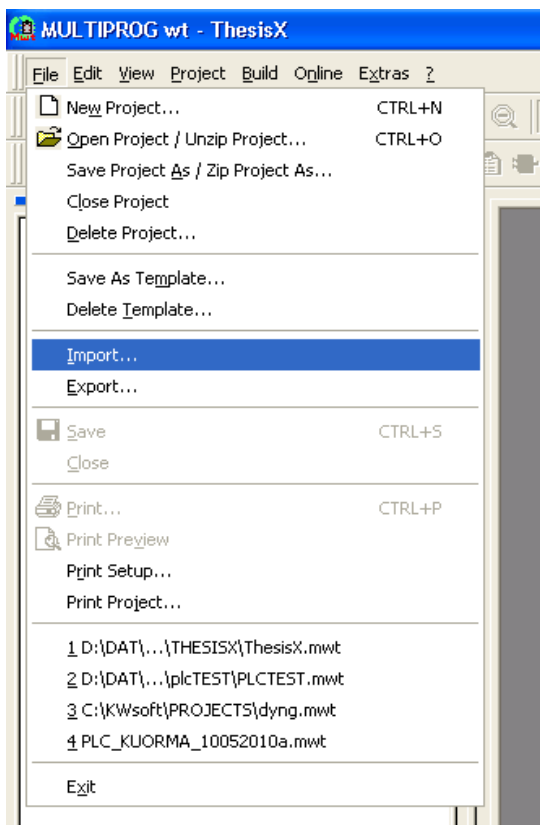


Figure 38. Import of RTU objects in MWT.

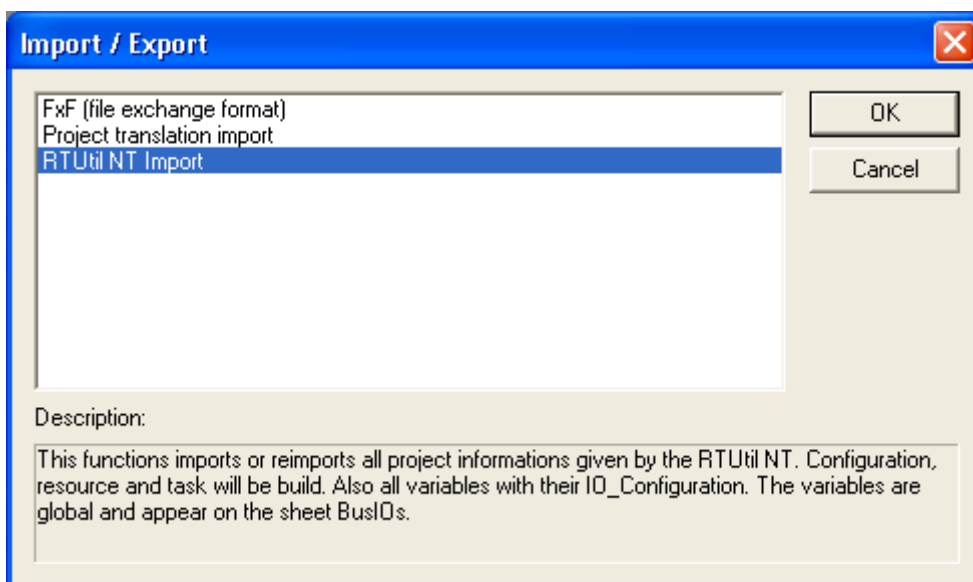


Figure 39. Select RTUtil NT Import.

Then build the POU by right clicking the folder named Logical POUs and the Insert. A new window will appear where you can select the program name etc., see figure 40 for an example of a new program.

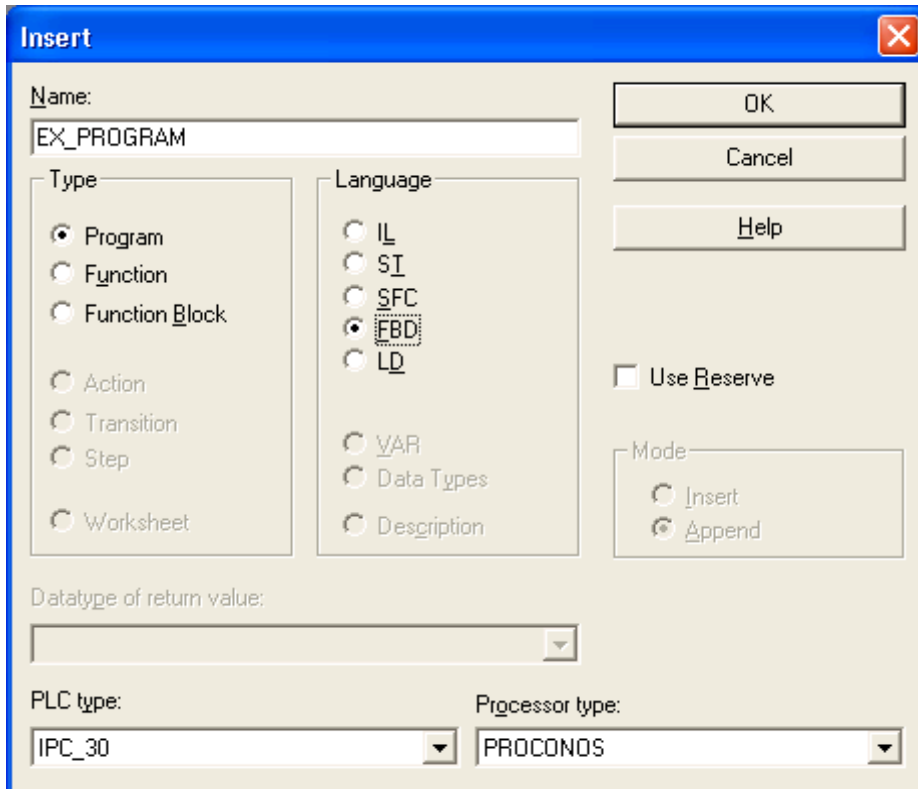


Figure 40. Example of a new program in MWT.

When the worksheets for the program appear, simply right-click the worksheet with a “function-block-looking” icon and select *Open Worksheet*.

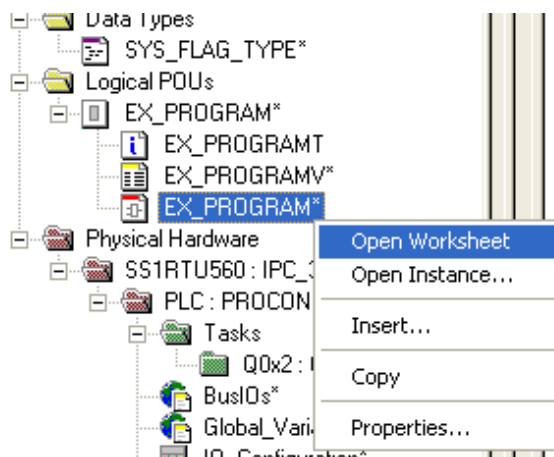


Figure 41. How to open worksheets in MWT.

Now a blank new window will appear in the center of MWT and here it is possible to add the function blocks and connect outputs to inputs, create variables and use basic function blocks like AND, OR, TON (on delay) etc.

Creation or selection of RTU variables can be done by just right-clicking any input or output on the function blocks and select *Variable*.

Create variable: At *Scope*, *Global* should be selected. If *Global Variables Worksheets* is selected at *Global_Variables* it is possible to type a name in the variable list -> select *Properties* -> select variable type and set the *initial value* for a new static variable.

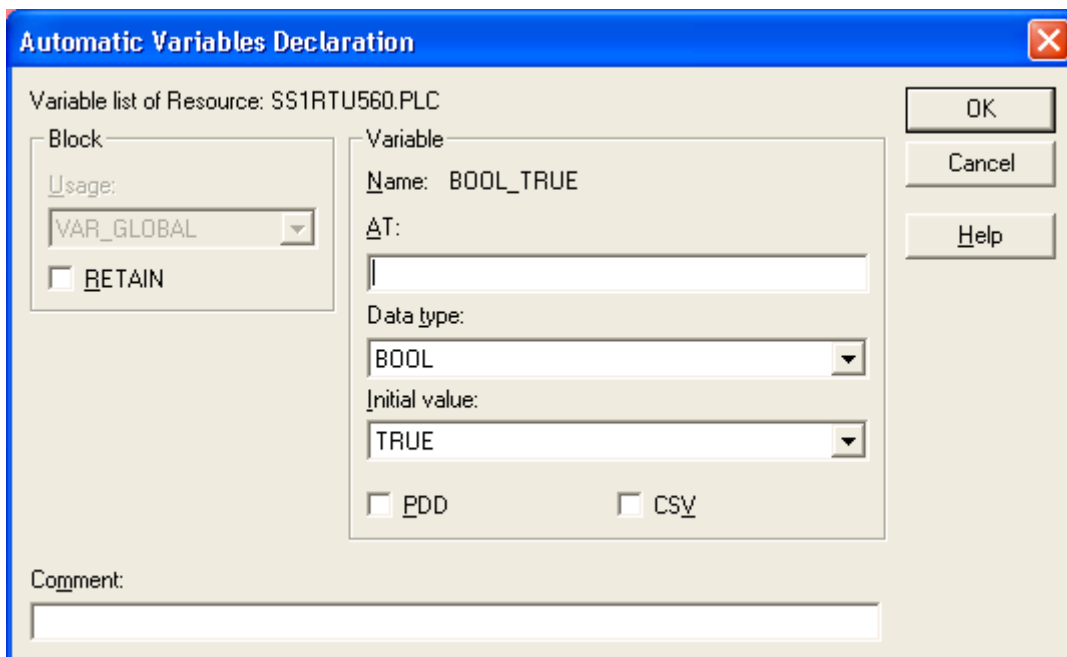
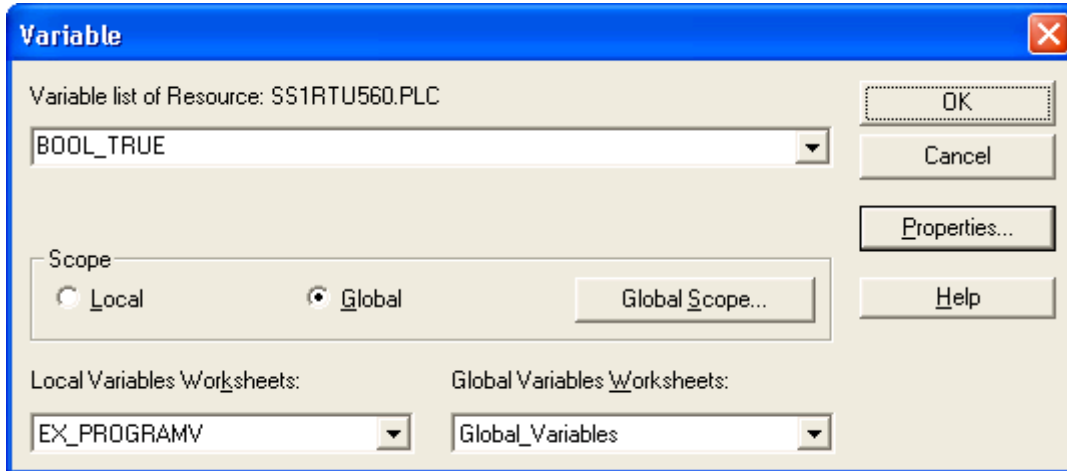


Figure 42. Creation of a new variable.

Use RTU signal object: At scope, select global and at *Global Variables Worksheets* select *BusIOs*. Now the RTU objects can be found under variable list. Notice that all RTU variables appear twice but one is ending with an I (input) and the other ending with a Q (output). I should be used for inputs and Q for outputs, the program will report an error when trying to do otherwise.

See figure 43 for an example of how a program could be built.

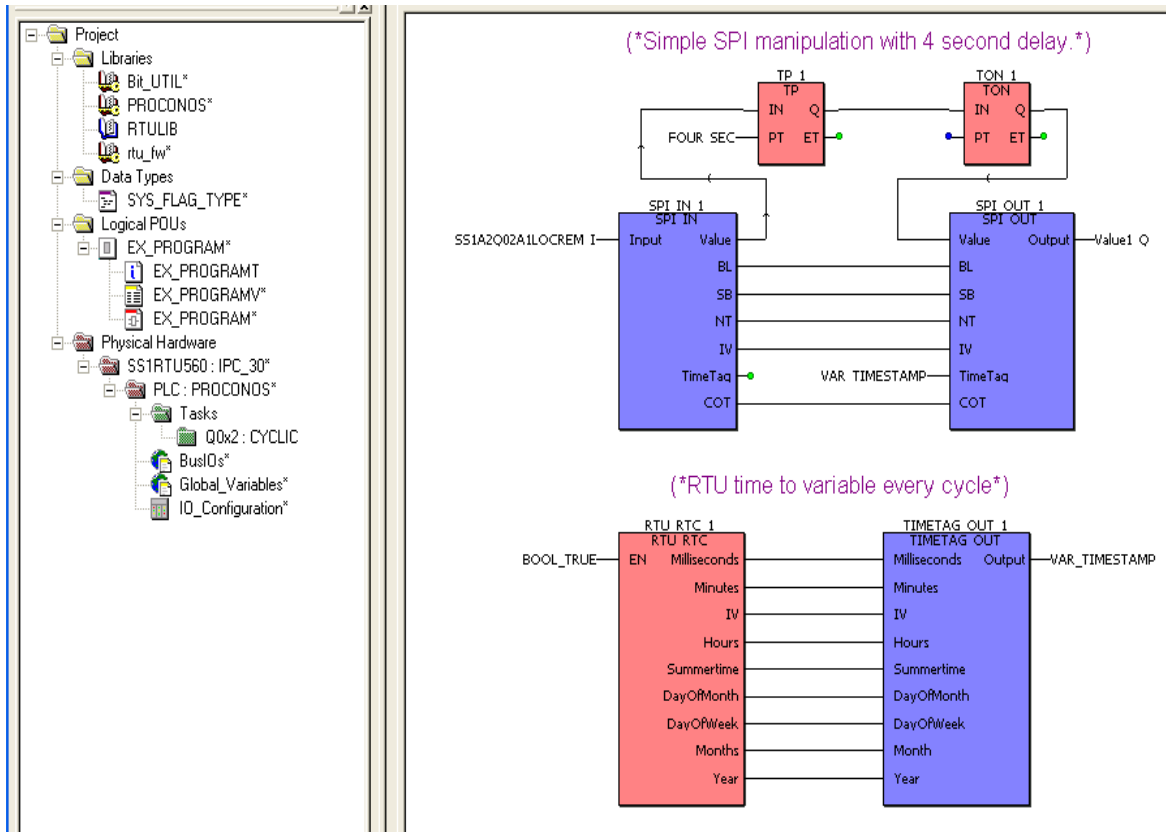


Figure 43. Example program worksheet in MWT.

Remember to compile the worksheets and build cross references when you are done. The program should then be assigned to the desired task.

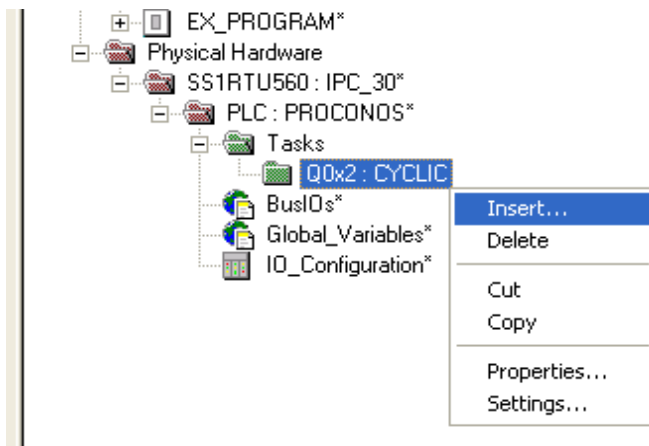


Figure 44. Insert program to task.

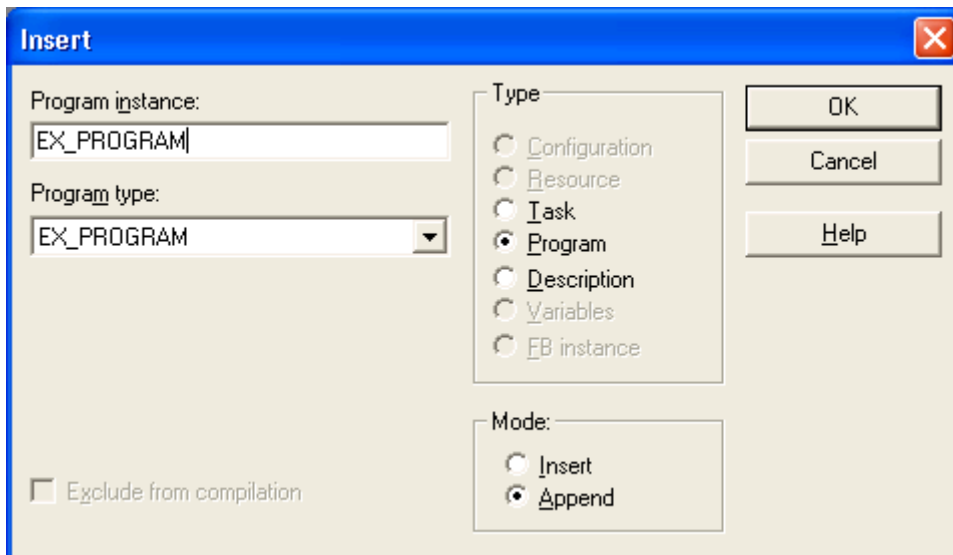


Figure 45. Selection of program to task.

Now the PLC program can be built from the Build menu.

One way to get the program into the RTU is to manually copy the file into the compact flash card. For the bootfile to be generated during the build process, right-click the *PLC:PROCONOS* folder and choose settings, then mark the checkbox *Generate boot project during compile*.

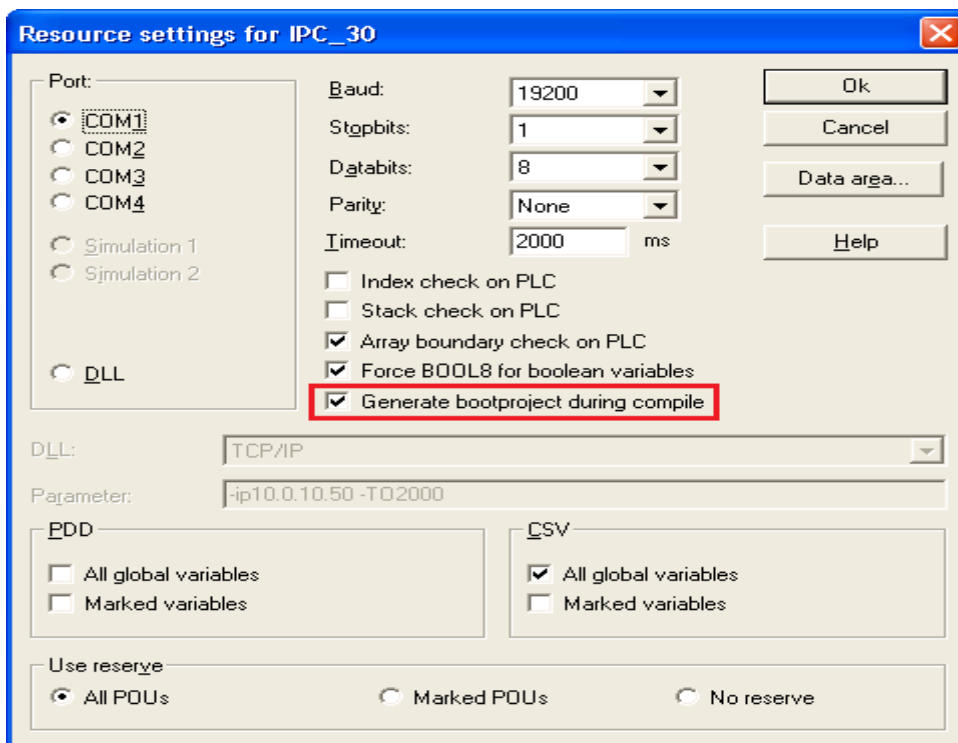


Figure 46. Selection for generating a boot file during compile.

The bootfile to be copied may be found under the project folders; <PLC program root directory>/<program name>/<RTU name>/r/plc/**bootfile.pro** and should be copied to <RTU flash drive root directory>/plc/.

It is also possible to directly connect with the RTU unit for uploading and debugging of the PLC program in real-time. That was not tested during the thesis project so no step-by-step guide for that method can be given here. At least for a TCP/IP connection a certain .dll file is needed. The connection settings should most likely be made in the PROCONOS settings window shown in figure 46.

11 HMI editor

The HMI editor is a Java code based software which is very easy to use.

Before using the HMI editor the RTU 560 configuration files should have been built as the .iod file will be imported into HMI editor.

Begin by creating a new project and a new page. Multiple pages can be made in case it is known how many are needed. A recommended way to build the HMI interface is to have one or more pages for the process display while event lists, alarms and measurement graphs can be made in separate pages.

For the process objects to be available for selection the .iod should be imported by selecting *Configuration file* from the *project* menu. Then simply browse for the .iod file and apply it.

The background size for the pages can be set by right-clicking the background and choosing properties. The page size can be defined in pixels vertically and horizontally. A good way to choose size is to find out the specifications of the monitor to be used as the local control monitor and then set 100 pixels less than the native resolution both vertically and horizontally. Otherwise objects near the bottom or far to the right might end up behind scrolling bars.

.gif and .jpg image files can be imported as background or used as indications in the HMI.

It is possible to switch between the regular *page editor* and *component view editor*. In the component view editor custom indicator blocks can be made. Four tabs representing the different states can be edited and their size can be modified in the same manner as the page background.

If the object is set to a DPI object then all four states are in use, but if a SPI is used for indication only the middle states (on and off) are in use.

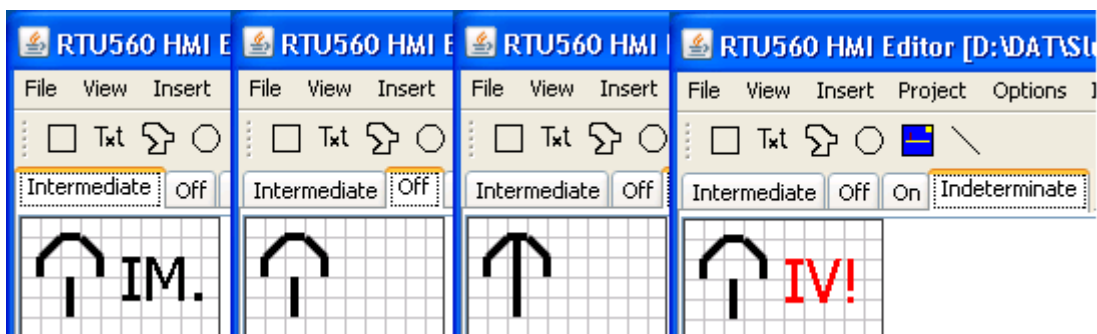


Figure 47. Example of a breaker location indicator in HMI editor

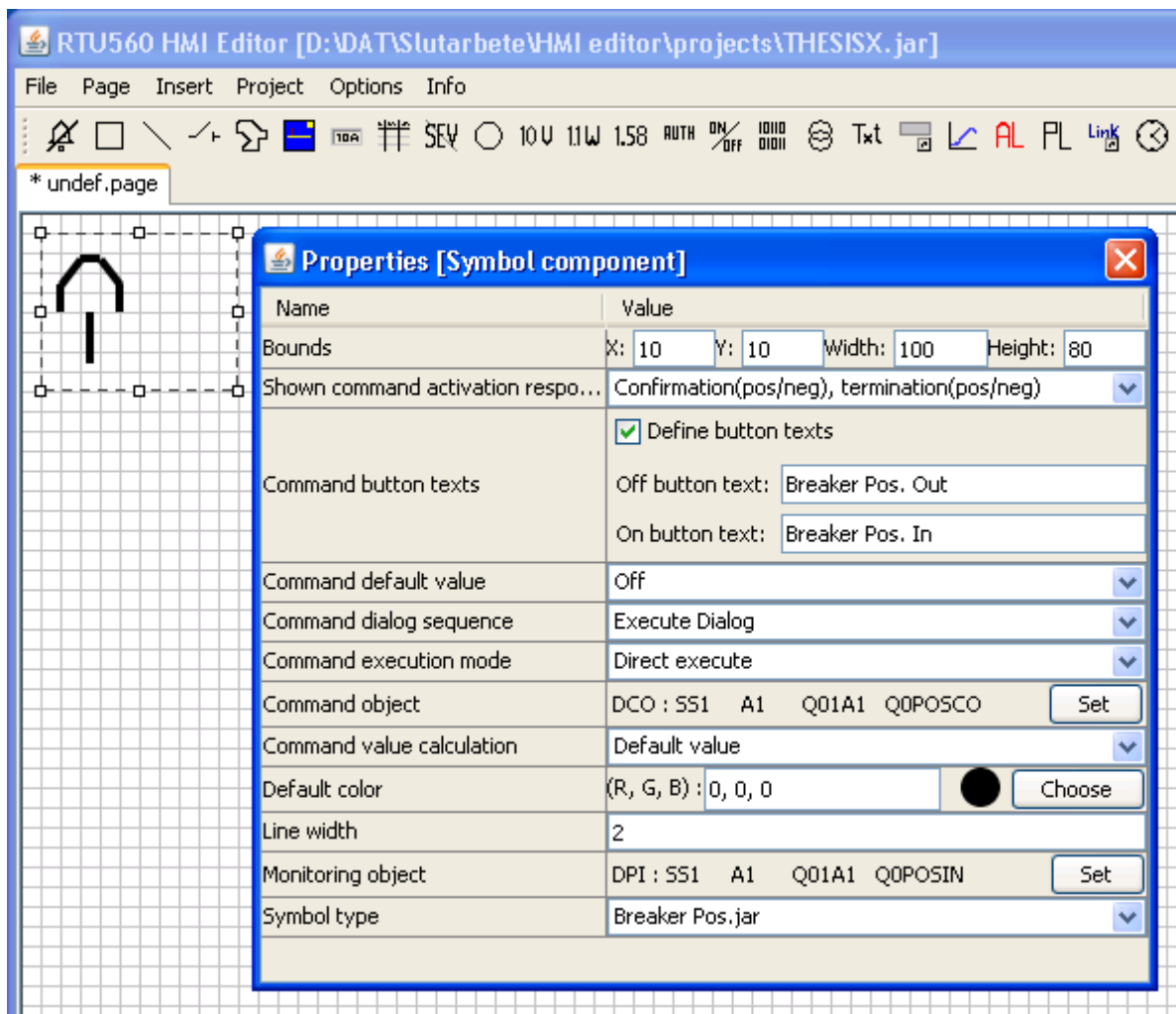


Figure 48. Example on how the breaker position indicator and command are set.

If more than the main page is made, remember to make buttons for navigating between the pages.

When the process display is made, simply save the project and upload the project file (.jar file format) to the RTU.

12 Web interface and upload of configuration files

To connect to the RTU web interface you can simply connect through a web browser such as Internet Explorer or Mozilla Firefox by entering the IP address of the RTU; *http://<RTU_IP_address>*. If the IP is unknown the RTU 560 CMU board has a jumper that can be moved so that it will get the default IP address of 192.168.0.1.

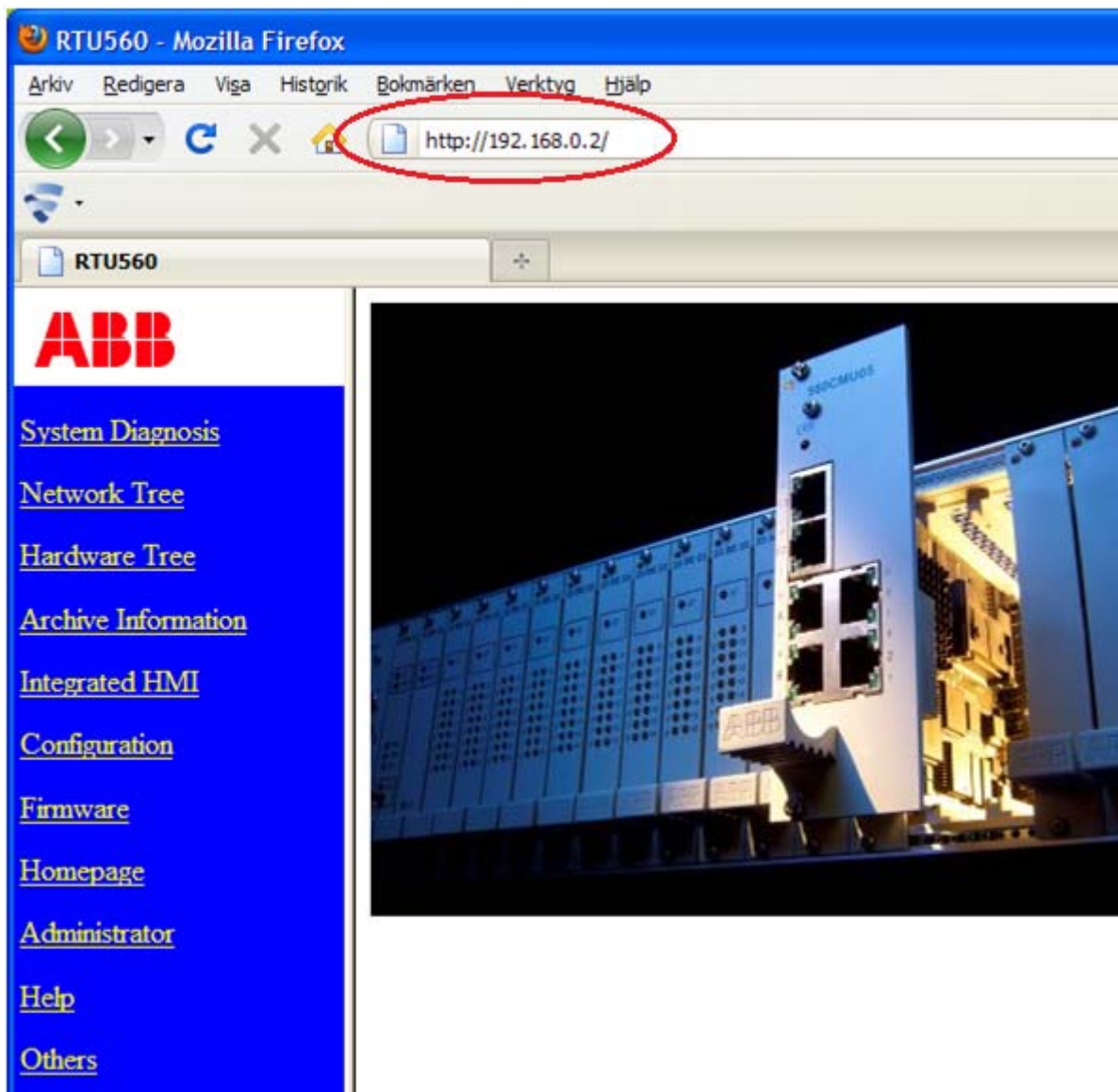


Figure 49. RTU 560 web interface welcome screen.

Log in as user *Load* with the password *Load* to be able to upload configuration files.

Different default login users:

Username	Password (by default)
Load	Load
Admin	Admin
Control	Control
Show	Show
Operator	Operator

Configuration File Manager

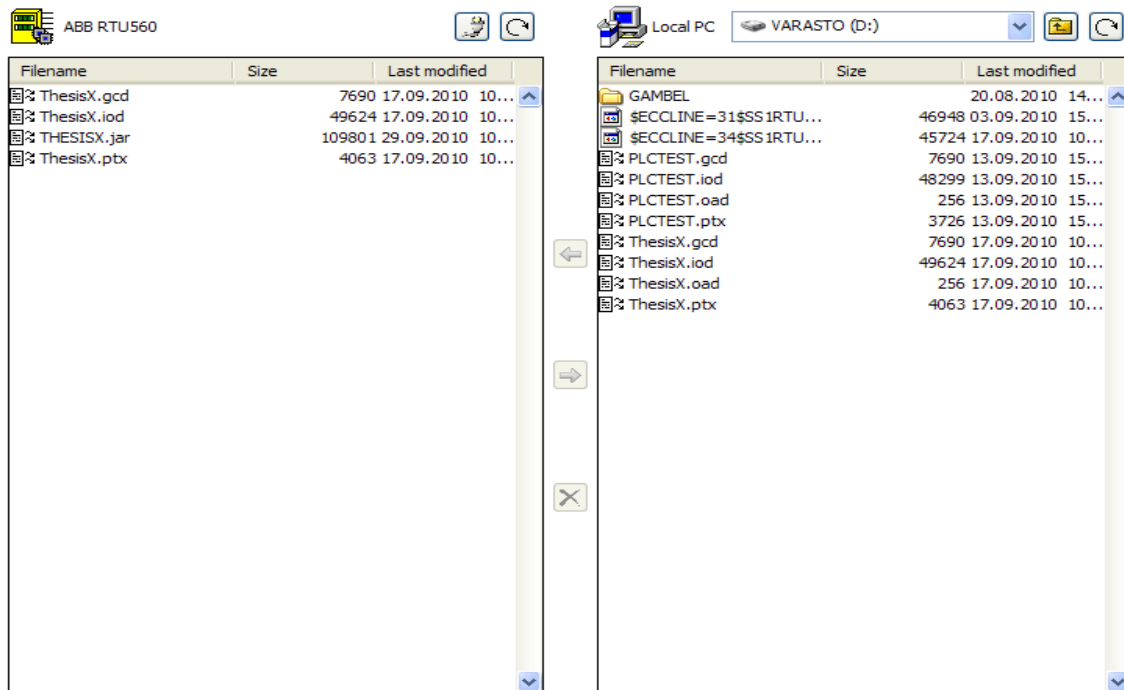



Figure 50. Configuration file manager in RTU 560 web interface

When logged in as Admin the user has the rights to change passwords and “roles” (user rights) for the other users. The Admin user can also enable the test mode on the RTU web controls.



- [System Diagnosis](#)
- [General View](#)
- [Detailed View](#)
- [Network Tree](#)
- [Hardware Tree](#)
- [Archive Information](#)
- [Integrated HMI](#)
- [Configuration](#)
- [Firmware](#)
- [Homepage](#)
- [Administrator](#)
- [Help](#)
- [Others](#)

Debugging Control

User logging/debug interfaces

Control test mode	<input type="text" value="disabled"/>	<input type="button" value="Enable"/>
PLC online debugging	<input type="text" value="disabled"/>	<input type="button" value="Enable"/>
RIO protocol logging	<input type="text" value="disabled"/>	<input type="button" value="Enable"/>

Developer debug interfaces

IEC61850 trace log	<input type="text" value="not active"/>	<input type="button" value="Activate"/>
OS debugging agent	<input type="text" value="not active"/>	<input type="button" value="Activate"/>

Figure 51. Debugging control options

The HMI application can be launched from the web interface, but a quicker method is to right click the HMI launch link, copy the link address and create a shortcut icon on the desktop for easier access to HMI.

A problem noticed in the thesis work was that even if the web browser Firefox had Internet access and a connection with the RTU, the HMI application would not launch if Internet Explorer was set to offline mode.

13 Final words

Most likely this guide does not cover all situations that might cause problems, but hopefully it can help users avoid a few. If mistakes are found in this guide or if you have suggestions for improvements, don't hesitate to contact me by e-mail to: joakim.ainasoja@veo.fi so that I can update this document accordingly.

