Cape Peninsula
University of Technology

DEVELOPMENT OF AN EMBEDDED SYSTEM ACTUATOR NODE FOR INTEGRATION INTO AN IEC 61850 BASED SUBSTATION AUTOMATION APPLICATION.

By

John-Charly Retonda-Modiya

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: Electrical Engineering

in the Faculty of Engineering

at the Cape Peninsula University of Technology

Supervisor: Prof. P. Petev
Co-Supervisor: Mr. C. Kriger

Bellville campus
November 2012

Declaration

I, John-Charly RETONDA-MODIYA, declare that the contents of this thesis represent my own unaided work, and that the thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed   Date

Abstract

The introduction of the IEC 61850 standard in substations for communication networks and systems by the International Electrotechnical Commission (IEC) in 2003 provided the possibility for communication between devices of different manufacturers. However, the advent of this standard also brought about many challenges associated with it.

The challenges introduced by this fairly recent standard of communications in Substation Automation Systems (SAS), and the need for the development of cost effective IEC 61850-compliant devices, motivated the decision of the Centre for Substation and Energy Management Systems within the Electrical Engineering Department of the Cape Peninsula University of Technology to focus on the implementation of the IEC 61850 standard using an embedded hardware platform.

The development of an IEC 61850 embedded application requires substantial knowledge in multiple domains such as data networking, software modelling and development of Intelligent Electronic Devices (IEDs), protection of the electrical system, system simulation and testing methods, etc. Currently knowledge about the implementation of the IEC 61850 standard usually resides with vendors and is not in the public domain.

The IEC 61850 standard allows for two groups of communication services between entities within the substation automation system. One group utilizes a client-server model accommodating services such as Reporting and Remote Switching. The second group utilizes a peer-to-peer model for Generic Substation Event (GSE) services associated with time-critical activities such as fast and reliable communication between Intelligent Electronic Devices (IEDs) used for protection of the power network.

The messages associated with the GSE services are the Generic Object Oriented Substation Event (GOOSE) messages. The use of GOOSE messages for protection of the electrical system is very important in modern substations. Detailed knowledge of the structure of these messages is important in instances requiring fault diagnosis to determine the cause of mal–operation or to address interoperability concerns or when developing custom IEC 61850-compliant devices with limited functionality.

A practical protection application (overcurrent) case study is presented where GOOSE messages are exchanged between a commercial IED and an IEC 61850-compliant controller based on an embedded platform. The basic data model and software development of an actuator node for a circuit breaker is proposed using an IEC 61850 communication stack on an embedded platform. The performance of the GOOSE messages is confirmed to be as per the functional behaviour specified, and per the IEC 68150 standard in terms of the temporal behaviour required.

This thesis document tables the methods, software programs, hardware interfacing and system integration techniques that allow for the development and implementation of a low cost IEC 61850-compliant controller unit on an embedded systems platform for the substation automation system.

The overcurrent case study distributed between a commercial IED (SIEMENS Siprotec device) and the actuator application developed on an embedded platform for this project (DK60 board) is in compliance with the IEC 61850 standard and utilizing GOOSE messaging is successfully completed both in terms of functional and temporal behaviour.

This novel research work contributes not only to the academic community, but to the international Power Systems community as a whole.

Keywords: IEC 61850 standard, IEDs, GOOSE message, software modelling, software development, substation automation systems, communication stack, embedded systems, actuator.

Acknowledgements

> Jeremiah 10: 23: Well know, O Jehovah, that to earthling man his way does not belong.
> It does not belong to man who is walking even to direct his step.

I would like to express my sincere appreciation to Professor R. Tzoneva, (Head of the Centre for Substation Automation and Energy Management System- CSAEMS), Professor P. Petev (my supervisor), M. Carl Kriger (Co-supervisor) and M. Shaheen Behardien for their constant supervision, support, motivation and encouragement throughout this thesis. I have really found in you a family for everything I needed. May the blessing of the Most High remain upon you.

Special thanks to Dr. Alexander Apostolov (PAC World magazine), M. Karlheinz *Schwarz* (NettedAutomation), both support teams of Beck IPC (Germany) and SystemCorp (Australia). Your close cooperation as field experts has made this work possible in spite of the distance between us. The equipment support from SIEMENS, ALECTRIX, SEL, RuggedCom, ABB, RTDS and others is greatly appreciated.

I must also express my gratitude to my colleagues and the staff at the CSAEMS, namely Fessor Mbango (former student at CPUT), Phaphama Panda (Secretary), M. Xolile Bekwa (Technician) and many others. You have shown a true friendship and demonstrated to me the Spirit of Africa “Ubuntu”. I will keep you in my heart forever.

I would like to further extend my deepest gratitude to my sister Liliane Anguilet (former student at CPUT) for her advice and humour, my fiancé Laureith Abika for her affection and patience during all these years, my brothers and sisters Stevy, Rudy, Tatiana, Flavia, Grace (…) for taking care of my grandmother while I was away from home, my fellow friends of the Gabonese Student Association in Cape Town (ADEGAC) for their encouragement, all my friends of the Association of International Students (AIS) of CPUT for their friendship.

The financial assistance received from CPUT towards this research is acknowledged. The financial assistance of the Gabonese government is also acknowledged.

May God bless you!
John-Charly RETONDA- MODIYA

Dedication

This thesis is dedicated to my parents M. **Roger Retonda** and **Manga Jeanne-pierette**. Further dedication goes to my grand-father **Eugène Capito** (Akewa mpolo) and to the entire family for all their committed support and patience.

May God continue to bless us!

Table of Contents

List of Figures

List of Tables

Glossary of Terms

Bay	A substation consists of closely connected sub parts with some common functionality. The bay level represents an additional control level below the overall station level.
Bus:	Communication system connection between Intelligent Electronics Devices (IED) with communication facilities.
Class:	Description of a set of objects that share the same attributes, services, relationships and semantics.
CT	Current Transformer. A device that transforms current from one magnitude to another magnitude.
data object	Part of a logical node object representing specific information for example status or measurement. From an object-oriented point of view, a data object is an instance of a data class.
FPGA	Field-Programmable Gate Array(s) is an integrated circuit designed to be configured by the customer or designer after manufacturing.
Function(s)	Task(s) performed by the substation automation system i.e. by application functions. Generally, functions exchange data with other functions. Details are dependent on the functions involved. Functions are performed by IEDs (physical devices). A function may be split into parts residing in different IEDs but communicating with each other (distributed function) and with parts of other functions. These communicating parts are called logical nodes.
GOOSE	Generic Object Oriented Substation Event is a high performance multi-cast messaging service for inter-IED communications, and is used for fast transmission of substation event.
HMI	Human Machine Interface
IED	Intelligent Electronic Device is any device incorporating one or more processors, with the capability to receive or send data/control from, or to, an external source. Device capable of executing the behavior of one or more, specified logical nodes in a particular context and delimited by its interfaces
Interoperabil -ity	Ability of two or more IEDs from the same vendor, or different vendors, to exchange information and use that information for correct execution of specified functions.
Line diagram	Schematic representation of a power system line or bus connection near an electrical node (mostly electrical substation).
Logical node	Smallest part of a function that exchanges data. A logical node is an object defined by its data and methods.
Logical node data	Information contained within a logical node. The term encompasses ACSI data, control blocks.
Merging unit (MU)	Interface unit that accepts multiple analogue Current transformer (CT) or Voltage Transformer (VT) and binary inputs and produces multiple time synchronized serial unidirectional multi-drop digital point to point outputs to provide data communication via the logical interfaces
MMS	Message Manufacturing Specification protocol
Model	Representation of some aspects of reality. The purpose of creating a model is to help understand, describe, or predict how things work in the real world by exploring a simplified representation of a particular entity or phenomenon.
Physical node	Point of connection on a physical device to a communication network. A physical node is a multi-functional unit providing both the communication server and the mapping to the real substation IED.
Router	Networking device used to interconnect multiple network
SAS	Substation Automation System. Provides automation within a substation and includes the

	IEDs and communication network infrastructure. An automated substation is a substation, where all secondary equipment within a substation is interlinked with communication buses.
SCADA	Supervisory Control and Data Acquisition
SCL	Substation Configuration description Language is a description language for communication in electrical substations related to the IEDs. This implies the use of configuration file.
State machine	The functional behaviour of any IED, logical node or object, can be defined and delineated by means of a state machine. This describes, normally by means of a state diagram, the functionality, responses, actions and re-actions, as a series of discrete, linked states, together with the criteria governing the transition from one state to another specific state.
Switch	Networking device used to interconnect Ethernet cable from different communicator in order for them to exchange information.
Unified Modelling language	Standardized constructs and semantics for diagrams, including state machines, which are used to describe/specify the functionality of an IED, object model or a process.
VHDL	*VHSIC Hardware Description Language* (VHSIC: *very-high-speed integrated circuit*) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits.
VT	Voltage Transformer. A device that transforms voltage level from one magnitude to another magnitude.

CHAPTER ONE:

INTRODUCTION

1.1 Motivation for research and problem definition

1.1.1 Motivation for research

Substations have always played a pivotal role traditionally in the power network for the protection of costly equipment and for the switching, routing, and distribution of power.

Internationally the power networks have expanded dramatically over the decades: the interlinked North American Power Network is now commonly regarded as the largest man-made machine in the world. Outside the traditional roles of protection and distribution, the automated monitoring, control and knowledge-based decision making within these expanded networks in conjunction with the intense interest in the “Smarter Grid” has as a consequence also driven a requirement for increased Substation Automation Systems (SAS).

Technologies to accommodate the increasing requirements of substation automation systems have evolved from the earliest days of hard-wired electromechanical systems (with no numerical computational capability) through non-bus-based numerical relays with limited computational capability, to standardized multi-functional bus-based systems with significant communications, memory, and computational capabilities. The consequence of non-standardization has often led to costly single-vendor proprietary solutions with very limited interoperability prospects. The imperatives of cost, maintainability, flexibility, and interoperability largely drove the early initiatives in the 1990’s for the promulgation of a standard for Substation Automation Systems. The nett result of this was the release of the IEC 61850 standard in 2003.

The technology needs mentioned above have driven a manufacturing industry to cater for it. Due to the fact that not only is costly investment required in manufacturing, but also for research, development, and especially the guaranteed performance in line with stated specifications to avoid aspects of liability. Due to these constraints only a few vendors internationally have become prominent in this market (e.g. such as SEL, ABB, Alstom, Siemens, etc.).

The expertise of developing equipment in compliance with the IEC 61850 standard lies almost solely within the vendor companies mentioned above. Academic institutions generally have not yet gained a deeper understanding of the IEC 61850 standard. Very little has been done in the academic sphere for teaching and familiarisation with the new technologies, or research into, or development and the adaptation thereof. Currently the situation is that detailed knowledge of implementation of the standard is not available in the public domain as industrial vendors ensure that their methods and implementation techniques remain confidential for economic leverage and also as justification for the high costs of the field devices.

The broad motivation for the project is familiarisation with the IEC 61850 standard to create a foundation to the degree that implementation of the IEC 61850 standard can be realized to some extent through experimentation. This foundational knowledge-base is essential for a) the development of further projects within The Centre for Substation Automation and Energy Management Systems (CSAEMS) under the auspices of which the project is being conducted, but also b) to be able to appreciate and respond to the changes made to the standard as it evolves.

The current research project is developed within the CSAEMS at the Electrical Engineering Department at the Cape Peninsula University of Technology. The CSAEMS is still in an embryonic phase and this project is supportive of the objectives of the centre.

1.1.2 Problem definition

The international standard - IEC 61850 for communication networks and systems in electrical substations is important within power utilities for the purpose of data acquisition for operational and management issues, easy reconfiguration, scalability and integration within the substation. The push for standardization has been assisted by the technological advances in the protection and monitoring devices within substations, and the ability for these devices to communicate and distribute field data amongst each other. Challenges of interoperability emerged as devices from different vendors were unable to share critical information for monitoring, protection and control.

The IEC-61850 standard is rapidly gaining popularity around the world for its flexible data communication across the IEDs in electrical substations. The model architecture for data commutation, which is based on an object-oriented approach, enables addressability and interoperability among equipment from different vendors. This introduces new challenges for software and hardware developers in the interpretation of the IEC 61850 standard to develop software for IEDs to ensure interoperability is achieved.

Many Intelligent Electronic Devices (IEDs) currently used in substations are multifunction devices that are able to perform protection, metering, control and monitoring functions. These devices are directly connected to sensors and actuators with high voltage copper cables. Since the inception of the IEC 61850 standard the “process bus” was proposed in addition to the more common “station bus”. Sensing units connected to the “process bus” communicate with other parts of the SAS supplying raw sampled value data to the system. This means that the development of a new generation of addressable sensing units (merging units) and actuator units is required. This provides the opportunity for research and development.

The problem can be sub-divided into the following:

- Detailed understanding and implementation of the IEC 61850 standard

  As stated previously, academic institutions do not have a detailed understanding of the IEC 61850 standard for an embedded system implementation. This knowledge of implementation methods usually reside with vendors and are generally not available in the public domain.
- Difficulty to source devices with limited or specific function nodes with IEC 61850 capabilities on the market.

Currently, the control of primary equipment such as circuit breakers within the Substation Automation Systems (SAS) is achieved using long hardwired cables that connect circuit breaker controllers (conventional actuator) to multifunctional and costly IEDs. Figure 1.1 shows the current implementation (limited) of the IEC 61850 standard (a) versus the extended or full implementation of it (b).

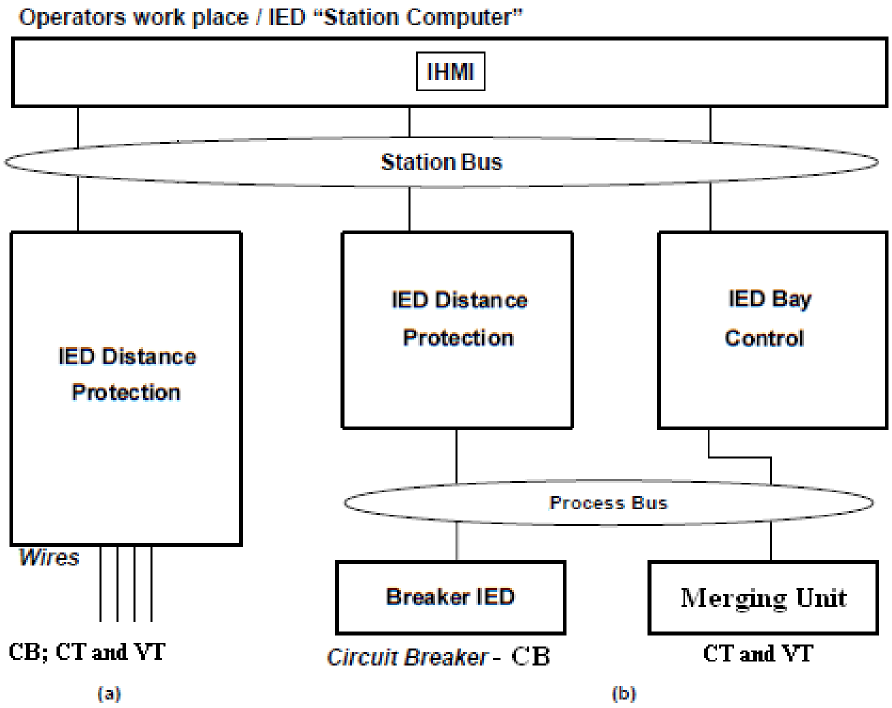


Figure 1.1: Limited IEC 61850 implementation (a) vs extended implementation (b)
Adapted from (Brand and Wimmer, 2003)

These IEDs are able to perform multi-functional (protection, control, metering, data acquisition…) roles. As a result, the installation cost of the SAS is relatively high. Development of IEC 61850 SAS or refurbishing of conventional SAS to be IEC 61850 compliant ensures the reduction of installation and maintenance costs. This justifies the need to develop new cost effective and reconfigurable IEDs such as merging units and actuators.

Case study: Figure 1.2 presents the proposed case study for this project. An IEC61850-compliant test injection set is used as a fault generator into an IEC 61850-compliant IED. The intelligent circuit breaker controller acts as an actuator[1] and is used to trip an emulated circuit breaker in response to an IEC 61850 based GOOSE message.

[1] The word "actuator" is used here in the sense of the desire functionality and do not refer to the process bus device as per IEC 61850.

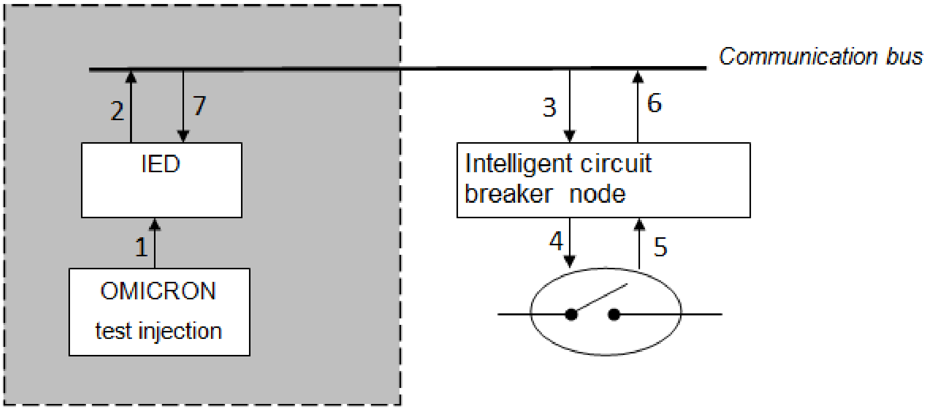


Figure 1.2: Project case study definition

The numbers in Figure 1.2 represent the communication flow of data between IEDs:

1: Simulate a fault condition to an IED.

2: Transmit a GOOSE message in the case of fault detection (over-current).

3: Receive the GOOSE message and take relevant action if required.

4: Open circuit breaker.

5: Read the current state of the circuit breaker.

6: Send a GOOSE message indicating the status position of the circuit breaker.

7: Read the incoming GOOSE message to clear the flag for a detected fault (Optional).

This research project focuses on the development of a cost effective controller with limited capabilities. This controller should be able to generate and respond to an IEC 61850 message and be integrated into a case study on a protection application distributed between a commercial vendor IED and the embedded systems-based controller. The project requires the development or adaptation of software, and interfacing of hardware to achieve the goal of system integration.

1.2 Research aim and objectives

1.2.1 Research aim

The project aim is to develop methods, software algorithms, hardware interfacing, a system integration technique, and a testing methodology that will allow for the development of a low cost IEC 61850-compliant actuator with a performance methodology to determine its functional and temporal behaviour, on an embedded platform interfaced to a circuit breaker emulator.

1.2.2 Research objectives

In order to achieve the aforementioned aim, the following project objectives are proposed:

- Literature review: history of IEDs and review of design theory for IED hardware and software to be conducted.
- Literature review: analysis and comparison of method for IEC 61850 modelling - development and implementation aspects to be undertaken.
- The frame format of the GOOSE message and its successful transmission over the network (as an indicator of functional behaviour) to be analysed utilizing data networking software utilities.
- A method for IEC 61850 embedded project development and system integration to be investigated and developed.
- Identification of a suitable hardware platform and software tools for the implementation of the software algorithms for the process application which has to be developed.
- A method for GOOSE performance evaluation in order to confirm that the communication stack used as well as the developed process application meet the temporal behaviour requirements specified by the IEC 61850 standard is to be investigated and developed.
- A circuit breaker emulator to be designed, implemented and interfaced to the actuator (embedded system platform).
- A test scenario for the aspect of cyber security to be developed.

1.3 Project assumptions and delimitation

1.3.1 Project assumptions

The following assumptions with respect to the scope of the research project are made:

- The IEC 61850 standard is sufficiently well documented to support any relevant data modelling or development processes and references to other related standard are clearly stated;
- There are software and hardware tools on the market that allow for the development of IEDs with limited IEC 61850 capability;
- The hardware target that can accommodate the design is available on the market and it can be reconfigured according to the implementation method and application requirements;

- A Substation Automation System environment can be created using a test-bench in a lab to simulate and implement the necessary monitoring and control functionalities in conformance with the IEC 61850 standard; and
- Relevant test equipment is available for the research study.

1.3.2 Project delimitation

The research project is limited to the development of an IEC 61850-compliant intelligent circuit breaker controller with limited functionality.

The following tasks form part of the project:

- Development of methods for IEC 61580 standard implementation;
- Software modelling as well as software development for the intelligent circuit breaker controller;
- Hardware integration; and
- Testing of the Intelligent Circuit Breaker Controller logical operation.

The following activities fall outside of the scope of the research project:

- Development and implementation of a complete IEC 61850 control model that uses the Select-Before-Operate method;
- SCADA system development;
- Design of current transformer (CT) and voltage transformer (VT);
- Design and implementation of a merging unit;
- Tripping of a real substation circuit breaker;
- Development of high voltage and/or high current interface circuitry;
- Management /or control of network communications within the SAS;
- Electromagnetic compatibility tests (EMC);
- Development of an IEC 61850 communication stack; and
- Mathematical modelling of a circuit breaker.

1.4 Research methodologies and techniques

The research focuses on the development of an IEC61850 application based on a hardware platform that supports IEC61850 functionality. The application requirement is that identification of the problem is accomplished first. Secondly, the investigation of a hardware platform that can accommodate the software application has to be performed by studying the internal structure of an IED. Next a logical device model of the intelligent circuit breaker controller must be developed together with the corresponding process application using appropriate software tools. In order to address the issues tabled in the brief discussion above the following methodologies and techniques are proposed:

a) Literature review – A detailed literature review of the IEC 61850 standard and other relevant areas will be conducted. A clear understanding of the IEC61850 standard philosophy is required as well as other important related standards (UML, OSI model, etc.). Existing methods for implementation of the IEC 61850standard have to be reviewed and researched. A review of the literature for the evolution of IEDs and the global structure of the Substation Automation Systems will be conducted.

b) Software development methods – The software tools required for modelling of an IEC61850 application will be researched and identified. Understanding of UML diagrams, syntax and structure of XML files, and programming within “C” environment

c) Hardware interfacing methods – Empirical investigations will be conducted to determine how commercials IEDs function in an IEC61850 environment. It is imperative that an understanding of the interaction between the user application, the communication stack, the operating system and the hardware system interface be developed.

d) Testing and performance evaluation methods – Test methods or procedures will be used to verify the correct implementation or realisation of an IEC61850 application on the selected hardware target. The notion of interoperability defined in the IEC61850 standard will be applied here. The developed device will be subjected to performance evaluations to verify its functional and temporal behaviour under real-world conditions.

1.5 Originality of the Thesis

Most of the work in the area of power network protection is the responsibility of the municipalities and national power providers. These organisations use vendor equipment and the emphasis is not on developing their own equipment, but in many cases they use the vendors to implement the needed solutions.

Within the academic sphere, first and foremost there is no curriculum or even training for this technology at the moment. Secondly this research work focuses not on using vendor equipment, but on the development of an IEC 61850-compliant custom device with limited functionality.

This research will contribute to the establishment of detailed knowledge within the field of Substation Automation Systems. This will be accomplished by means of the foundational understanding and intimate knowledge of the IEC 61850 standard and the practical implementation thereof on an embedded systems platform.

1.6 Organisation of the Thesis

The thesis consists of seven chapters detailing the background information, problem definition, developed methods, software algorithms developed, integration techniques applied, challenges encountered and results of the research project.

Chapter 1 presents an overview of the research project highlighting the research aims, the research delimitation, the research methodologies, research aims and objectives, and a description of the originality of the thesis.

Chapter 2 presents the literature search and analysis methodology including a detailed literature review of among others the substation history, the evolution of IEDs, current methods of interfacing IEDs, and a review of current actuator solutions. A comparative review of structural design of microprocessor/microcontroller based relays and other areas are also tabled.

Chapter 3 discusses an overview of the IEC61850 standard with a particular focus on the structure of GOOSE messages. A list of hardware/software tools associated with IEC61850 development is tabled with a short description of each of them.

Chapter 4 provides detailed information about the target hardware platform that is used to implement an IED prototype with limited functionality. This chapter also describes the modelling of the selected case study based on the IEC61850 standard. The function of each module is specified together with the interactions between them using UML.

Chapter 5 details all the algorithms and the software development of the final application. A description of the main software tools used to develop the code is provided. The software algorithms of each module is explained and also the way the developed application interacts with the system interface (network card adapter and other input/ output modules).

Chapter 6 introduces the world of IEC61850 communication-based GOOSE messaging by means of experimentation. Various software tools for simulation, device configuration and network analysis are explored to gain a deeper understanding of the IEC61850 standard. This chapter presents the testing procedure to verify the correct implementation of the IEC61850 standard onto the target platform in detail. All results of this study are tabled and described.

Chapter 7 provides the conclusion to this research project. The benefits that this research project offers are described and future research prospects for expansion of this project and other related relevant projects in this field of study are identified.

1.7 Publications

1. Retonda, J., Petev, P., Kriger, C., and Behardien, S., 2012. Software development for integration of an embedded system platform into a minimal lab-scale IEC 61850 application. Submitted to SAIEE Research Journal (awaiting confirmation for publication).
2. Kriger, C., Retonda, J., Luwaca, E. and Behardien, S., 2011. Analysis of GOOSE and Sampled Value Message Structure for Educational Purposes. PAC World magazine.
3. Retonda, J. and Behardien, S., 2010. Simulation of an IEC 61850 based GOOSE message, and analysis of its structure. South African OMICRON User Conference 2010. Johannesburg. South Africa, 8-9 November 2010.

CHAPTER TWO:

Literature review

2.1 Introduction:

The Substation Automation System has been the centre of constant innovation over time with one of its main aims to deliver electricity continuously with adequate protection of field workers, substation equipment, the end user, and stability of the network grid. Over the years - as the protection, control, and communication technology evolved, this evolution was reflected within substations that are characterised by the equipment involved and the philosophy behind it.

Chapter 2 is organised as follows: Section 2.2.1 presents a brief history of protection technology in substations; Section 2.2.2 presents aspects of the body of literature under review, and Section 2.3 deals with the comparative analysis of the papers presented in Section 2.2.2.

2.2 Literature research methodology

The project focuses on the development of an actuator node with limited functionality for circuit breaker control. The project aim is to develop methods, software programs and hardware integration techniques that allow for the development of an actuator compliant with the International Electrotechnical Commission- IEC 61850 standard.

In order to successfully achieve the aims of the specified project it is therefore important to perform a thorough literature search based on some predefined objectives. The literature search done in this chapter is articulated around the following goals:

- Establish a clear chronological evolution of device technology in substations;
- Establish a clear chronological evolution of the main communication methods used in substations;
- Have a good understanding of the internal structure – both hardware and software of the modern IEC 61850 standard-based IED;
- Have a good understanding of the IEC 61850 and other related standards, and,

- Understand the technological impact and challenges introduced by the IEC 61850 standard for substation communications, distributed protection scheme deployment and internal hardware and software requirements.

The literature search has been conducted according to four perspectives which are:

- History of protection devices within the substation;
- Design of modern Intelligent Electronic Devices (IEDs) which are microprocessor-based;
- History of substation communications; and
- Development of specific Application Programme Interface (API) for IEC 61850 development.

Certain search strings or key phrases have been used during the literature search. These are presented below:

- "Substation communication protocol" and "Substation protection device". A selection of documents from the resulting search is shown in Table 2.1.
- "Design of microprocessor relays": A selection of documents from the resulting search is shown in Table 2.3.
- "IEC 61850 substation": A selection of documents from the resulting search for general review is shown in Table 2.5 and the modelling aspects are shown in Table 2.6.
- "IEC 61850 software": A selection of documents from the resulting search is shown in Table 2.7.
- "Protection GOOSE[2]": A selection of documents from the resulting search is shown in Table 2.8.

Figure 2.1 shows the evolution of the number of papers produced from 2002 to July 2012 and published by the IEEE (Institute for Electrical and Electronic Engineers) with regard to the field of embedded software related to IEC 61850 (Key search : "IEC 61850 Software").

[2] GOOSE is an acronym for Generic Object-Oriented Substation Event (See Chapter 3)

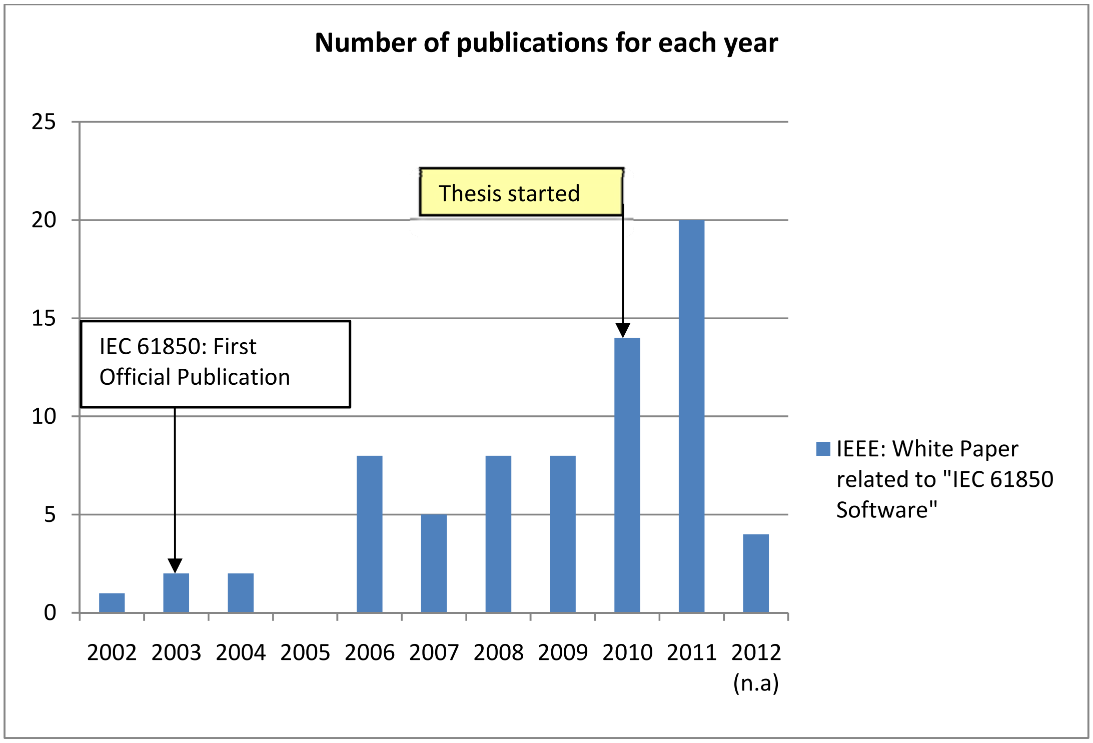


Figure 2.1: Publication rate for "IEC 61850 Software" topic

From the year 2002 to July 2012, seventy two papers have been published in the field of interest, namely IEC 61850 software. The graph clearly shows that there is a peak in the number of publications for the year 2011 with 20 papers being published. This peak happened eight years after the first part of the IEC 61850 standard was published, and one year after the current Master's thesis commenced. This serves to illustrate the importance of this current research work and also shows how relevant the current research is for the development of other IEC 61850 projects at the Cape Peninsula University of Technology.

2.2.1 Protection technology in substations

The developments within the field of electrical protection systems within substations are constantly evolving. Table 2.1 illustrates the work of Lundqvist (2003), Nordell (2003), Richards (2006) including Olovsson and Lejdeby (2008) on protection devices and communication technology evolution within the substation. These papers were sourced from the IEEE website using the search strings: "Substation communication protocol" and "Substation protection device".

Table 2.1: Comparison of Protection technology epochs in Substation Automation

Paper [3]/ Book	Emphasis and objectives of paper
Review papers/books published before the publication of the IEC 61850 standard	
(Dick, 1996) Review of communication standards for distribution automation applications	The paper is a review of communication standards with an emphasis on the Telecontrol communication protocol used in substations.
(IEEE Canadian Region, 2000)Electricity: The Magic Medium	This Book (available online at IEEE Canadian) is the most complete review on the history of Electricity (150 years of Electronics and Electrical history).
(Chaturvedi, 2002) Substation IED communications	The paper presents a review of the substation communication IED with respect to the UCA protocol and a brief comparison to the DNP protocol.
Review papers published on/or after the publication of the IEC 61850 standard	
(Nordell, 2003) Substation Communication History and Practice	The paper is a review that relates the history of most predominant protocols in the substation from the SCADA protocol to the IEC 61850 standard.
(Lundqvist, 2003)100 years of relay protection, the Swedish ABB relay history	The paper is a review of Protection device evolution over the last 100 years.
(Richards, 2006) Building Very High Reliability into the Design and Manufacture of Relays	The paper is a review of Protective devices in substations from electromechanical relays to modern numerical relays
(Walter, 2007)Protection history: the Start of Protection	This PAC World Article paints with astonishing clarity the early start of protection practices within power stations.
(Uzair, 2008)Communication methods (protocols, format and language) for the substation automation and control	The paper presents a detailed report on major protocols in use in the substation from Modbus to IEC 61850.
(Olovsson and Lejdeby, 2008) Substation evolution: Substation design in the 1900s and modern substations today	This paper is a review that analyses the architecture design of substation in the 1900's in comparison to modern substation design.
(Walter, 2009)Protection History: Generator Protection, The Beginning	The paper reviews the history and evolution of generator protection.

The information extracted from these review papers show the chronological evolution of the technologies within the electrical substation and are summarised below:

In 1881, Lucien Gaulard and John Gibbs obtained patents for a "series alternating-current system of distribution". The first transformer system to step up the voltage for transmission and step it down again for distribution is deployed (IEEE Canadian Region, 2000). This system used manual switches that were operated by personnel for intervention in emergencies – no significant substation automation.

[3] Table 2.1 (IEEE/ Key search): "Substation communication protocol" and "Substation protection device"

In 1882, Weston obtained patents for the first “Fusible Links” as part of primary protection. Fusible links are used to protect generators, lines and other equipment against short circuits and overload. The disadvantage of this type of protection is that when the fuse blows it has to be replaced (Walter, 2007). The combination of knife switches in series with fusible links produced the first generation of circuit breakers known as a tripping device (Walter, 2009).

Edison, T.A started operating the first central station and first public electric utility in New York on September4th1882. He used three 150 kW, 110 VDC, 1200 rpm “Jumbo Generators” (Walter, 2009).

In 1886, Hermann Meyer developed one of the first automatic circuit breakers combined with an electromagnetic undercurrent relay (Walter, 2009).

In 1898, the current transformer is invented by Benischke (Walter, 2009).

In 1905, the first standalone time-overcurrent relay is developed by ASEA (now ABB) (Walter, 2009). Later on, SIEMENS developed their first three-phase overcurrent protection "S" in 1936(Walter, 2009).

During the early 1960’s, the time delay within relays is achieved with the introduction of mechanical timers, an oil flask or a leather bantling (Walter, 2007). This is the beginning of Substation Automation Systems (SAS).

In 1965, the first major step in reducing the footprint of substations is ABB’s launch of gas-insulated switchgear (GIS). These smaller and compact switchgear enabled a reduction in the area needed to accommodate the substation by up to 70 percent - as compared to the conventional air-insulated switchgear (AIS), which in those days covered an area the size of a soccer pitch (ABB, 2011).

In 1968, ABB innovation in substation automation replaced conventional protection and control systems with numerical ones based on microprocessor technology and known as Intelligent Electronic Devices (IEDs). These offer advanced applications which enable operators to safely control and monitor the entire substation from a central control room (Lundqvist, 2003). At this point in time no communication standards were implemented and the issue of interoperability between different vendors becomes more and more prominent.

In 1970, the first static ultra-high-speed bus-differential relay is implemented (Lundqvist, 2003).

The beginning of the 1970's saw an increase in the use of integrated electronic circuitry based on transistor technology[4]. This gave an opportunity for extended capabilities to be added to SAS. Microcontrollers helped in the development of Supervisory Control and Data Acquisition (SCADA) systems, remote terminal units (RTUs) and high-speed communications systems. Industrial controllers have found their way into the system in the form of programmable logic controllers (PLCs) a few years later. PLC plus communications and appropriate transducers, has increased the SCADA system's abilities.

Between the 1980's and 2000's, desktop computers and appropriate gateways are used as the Human Machine Interface (HMI). Subsequently network connections within the substation are enhanced with web-based access and expert system programs (Wolf, 2009). The computational speed and memory capacity of microprocessor-based systems has constantly increased during this period. At the same time the size of the hardware platforms have decreased.

In 2004, ABB implements the very first IEC 61850 multi-vendor substation automation system that allows for interoperability (Lundqvist, 2003). Electromagnetic effects are reduced by the introduction of fibre optics within the substation automation system network. The new generation of standardised IEDs started to emerge.

The evolution of field devices within the substation can be divided into four categories or ages (AREVA, 2005). This can be condensed as described below:

- Electromechanical relays (1950's) use a moving armature to implement the protection element functionality. They are single-function devices and some of them could be interfaced to proprietary Supervisory Control and Data Acquisition (SCADA). Communications are based on telephone switching technology from a remote location to the substation or from the substation to the remote location. SCADA communication could not however allow for device-to-device communication within the substation system, and addressability based TCP/IP protocols are not supported.

[4] The progression trend has been from Small-Scale Integration (SSI) to Medium-Scale Integration (MSI) before the stage of Large-Scale Integration (LSI) onwards.

- Static relays (1960’s) used discrete elements based on transistor technology to implement their protection element functionality. Some of them have multiple functions integrated into one device. Communication is however still based on a proprietary SCADA system.

- Digital relays (1980’s) used microprocessors/ microcontrollers to implement their protection element functionality. These microprocessors/microcontrollers have relatively limited computational capability. Analogue to Digital Converters (ADCs) are used to sample all inputs, but due to the low processing capabilities of the microprocessor it is possible to implement only a few mathematically-based protection algorithms. Digital devices however have a higher integration capability compared to static relays. Communication is still dominated by proprietary SCADA systems.

- Numerical relays[5] (1986) relied on the use of specialised Digital Signal Processors (DSPs) as the computational hardware. This gave numerical relays a significant increase in the computational power capabilities over digital relays. Analogue values of current transformers (CTs) and voltage transformers (VTs) are translated into number format via powerful Analogue to Digital Converters (ADCs). This in combination with significantly increased computational and memory capabilities allow for more complex mathematical algorithms to be used (also to be multi-functional). These mathematical algorithms are then used to implement the protection element functionality. This opens the door to maximum integration capability and has reduced the overall costs associated with substation installations. This is as a result that the functional elements are reduced to software module integration. A few years later, TCP/IP functionality is added to the numerical relays and this allowed for addressability over a Local Area Network (LAN).

Recent numerical relays based on specialized microprocessor hardware (PowerPC, ARM, DSP, and FPGA) and Real-Time Operating System (RTOS) platforms are being developed and include built-in capabilities for the IEC 61850 standard communication.

Table 2.2 has been constructed based on information provided by Areva T&D on device type and their features (AREVA T&D, 2002). The chronology of

[5] Digital and numerical relays are also known as microprocessor based relays

communication protocols related to the substation has been collected at the Michigan Technological University (http://www.mtu.edu/ece/) (Nordell, 2003).

Table 2.2: Relay type vs communication protocol evolution

Epoch	Device Main Categories	Sub-device Categories	Main Features	Communication Protocol
1950's	Electromechanical relays	-	Used an armature to realise protection functions. Single function Devices. Configured by selection and manual setting. Local indication via Flag.	Proprietary protocols for SCADA system
1960's	Static Relays	1st Generation	No moving parts to create its characteristic. Use discrete electronic components (transistors, diodes, resistor, capacitors, inductors, etc.) for its operating characteristics. Configuration via switches. Indication via LED. Usually implemented as single function.	Proprietary protocols for SCADA system. Modbus (1979/ open Protocol)
		2nd Generation	No moving parts to create its characteristic. Use linear and digital integrated circuits for signal processing and logic functions. Configuration via switches. Indication via LED. Usually implemented as single function.	
1980's	Microprocessor based relays	Digital Relays	Uses microprocessors/ microcontrollers to implement relay functions. Those microprocessors have limited processing capacity. Introduce Analogue/Digital conversion. Flexible Internal logic using DIP switches. Use of Keypad/LED interfaces.	Proprietary protocols for SCADA system (More than 100 protocols). Ex: Profibus (1989)
		Numerical Relays 1986 (Lundqvist, 2003)	Uses microprocessors/microcontrollers to implement relay functions. These microprocessors have higher processing capacity. Introduce specialised Digital Signal Processor (DSP) as the computational hardware. Sampling of relay inputs using A/D converter for use in mathematical algorithms. Multifunction devices (Protection, control, etc.). Integrated measurement and recording. Programmable Logic. Advance communication facilities for local or remote and possibility to interface to legacy protocol.	**Effort for communication standardisation** IEC 870 series/ 1990-1995 UCA 1.0/ December 1991 DNP3/ November 1993 UCA 2.0/ December 1998 (Introduction of device to device communication based GOOSE) ICCP /April 2002 IEC 61850/ 2003

This section, Section 2.2.1 presents a brief history of Substation Automation Systems with a particular emphasis on protection equipment technologies. It is important to understand the evolution of the technology and constant improvements within the electrical substation so as to assess the advantages and gains which have been made, and problems which have been mitigated.

The previous tables and comments focused on review papers and the chronological development of the technologies. The section that follows focuses on recent work which has been cited in the literature, and is closely related to the development of the current research project. Relevant topics are presented and analysed with respect to predefined criteria. A critical analysis of the papers is presented in Section 2.3.

2.2.2 Project literature background

The search strings that have been used to gather documentation are:

- “**Design of microprocessor relays**”
  (Table 2.3: Structural design of microprocessor/microcontroller based relays, simulation and test)
- “**Substation communication**”
  (Table 2.4: Paper review on substation communication)
- “**IEC 61850 substation**”
  (Table 2.5: Basic concept Review of IEC 61850 standard)
- “**IEC 61850 modelling**”
  (Table 2.6: IEC61850 and modelling)
- “**IEC 61850 software**”
  (Table 2.7: Comparison of method for embedded implementation of IEC 61850 requirements)
- **"Protection GOOSE**”
  (Table 2.8: Comparison of method for evaluating and testing protection systems based on GOOSE messages)

In the tables that follow (except for Table 2.5), the criteria used to effectively analyse the papers are:

- Method: This column refers to the theory promoted by the author of the paper in order to achieve its goals.
- Implementation: This column looks at the type of implementation done from the point of view of the Hardware, software and protection scheme.
- Performance evaluation: This column report results obtained from the system performance point of view.

- Achievement in paper: This column put the focus on what the author has successfully demonstrated.
- Drawbacks and Comments: This column creates a parallel between the author works and the current thesis.

Points presented above are the basis for comparative analysis and further literature review.

For all tables in this section the “N/A” abbreviation means Not Applicable. Note also that relay technology epochs have been dealt with in section 2.2.1.

Table 2.3: Structural design of microprocessor/microcontroller based relays, simulation and test

Paper[6]	Emphasis and objectives of paper	Method	Implementation	Performance Evaluation	Achievement in papers	Draw backs and Comments
(Mansour and Swift, 1986)Design and testing of a multi-microprocessor travelling wave relay	Paper emphasis is the design and testing of a digital ultra-high speed (UHS) travelling wave-based protective relay. (Software and hardware).	Dedicated Master-Slave multi-microprocessor (Parallel processing)	Hardware: Uses four (4) 6802 CPU-based on single board Software: Not specified. Protection: Algorithm is based on method of the Karrenbauer modal transform	Results demonstrate that using multi-microprocessor could increase the system response; e.g. An average tripping time of 3.8 ms have been achieved for a line-to-line fault	Relay demonstrated in paper have achieved ultra–high speed (UHS) operation, phase selection and faults classification capabilities.	The use of multi-microprocessor architecture opens the way to a highly integrated algorithm, reliability and enhanced performance.
(Baigent and Lebenhaft, 1991)Microprocessor-based protection relays: Design and application examples	Paper emphasis is on the design of a microprocessor-based relay (hardware module interconnection) and technical specification.	N/A	N/A	N/A	N/A	Paper is a design guide line that expands general formula used while designing a microprocessor-based relay application.
(Zhu et al., 2006) Generating Detailed Software Models of Microprocessor-Based Relays	The paper describes a unified approach for computer-based simulation of microprocessor-based relays.	Combined computer simulation of power systems and protective relay software model.	Hardware: N/A Software: The relay model code is written using FORTRAN 77 Protection: Protection algorithm and other design formula are well described.	Results demonstrate the efficiency of generating software models for microprocessor-based protective relays for existing and new relays**.**	A software relay model is developed using a detailed digital model, which closely follows the hardware and firmware architecture module of the physical relays.	The use of relay models permits closed-loop simulations of the relays and approximate actual relays' characteristics so that their performance can be evaluated for a variety of situations. A clear definition of relay pickup time and operating time is also provided.

[6] Table 2.3 (IEEE/ Key search): “Design of microprocessor relays”

(Sahai and Pandya, 2011) A Novel Design and Development of a Microprocessor-based Current Protection Relay	Paper presents a step-by-step novel design of an electrical protection relay using a microprocessor.	Single microprocessor	Hardware: ATMega32 microprocessor Software: - N/A Protection: - N/A	The developed relays have been tested for various conditions, and results obtained are within acceptable limits of operation.	Paper has demonstrated the design and assembling of relay modules in order to achieve protection functions.	Because the paper presents important technical details on hardware interconnection, it should be considered for future projects toward building of complete IEDs.

Table 2.4: Paper review on substation communication

Paper [7]	Emphasis and objectives of paper	Recommendations	Results	Analysis
(Dick, 1996) Review of communications standards for distribution automation applications	The paper is a review of communication standards with an emphasis on the Tele-control communication protocol used in substations.	N/A	N/A	N/A
(Chaturvedi, 2002) Substation IED communications	The paper presents a review of the substation communication IED-based UCA and a brief comparison to the DNP protocol.	N/A	N/A	N/A
(Uzair, 2008)Communication methods (protocols, format and language) for the substation automation and control	The paper presents a detailed report on the major protocols in use in the substation from Modbus to IEC 61850.	N/A	N/A	N/A

[7] Table 2.1 (IEEE/ Key search): Substation communication protocol and Substation protection device

Table 2.5: Basic concept Review of IEC 61850 standard

Paper[8]	Emphasis and objectives of paper	Recommendations	Results	Analysis
General review of IEC61850 concept				
(ABB review, 2010) Special Report IEC 61850	The paper delivers a report on all aspects of IEC 61850 Ed.1 including some difficulty for interoperability and device test procedures before giving a brief overview of IEC 61850 Ed.2.	N/A	N/A	N/A
(Apostolov and Vandiver, 2010) IEC 61850 process bus - principles, applications and benefits	The paper analyzes the functionality, the functional hierarchy and the schemes typically available in distance protection relays with a particular emphasis on the use of the process bus component	N/A	N/A	N/A
(Siemens AG, 2010) Aspects on IEC 61850 Edition 2.0 and Current Activities	The paper presents a general overview of IEC 61850 Ed.2 and places the emphasis on the difference between Ed.1 and Ed.2 of the IEC 61850 standard.	N/A	N/A	N/A
(Montignies et al., 2011) IEC 61850 in the oil and gas industries	The paper emphasis is on the monitoring and control of electrical installations. The author also clarifies some incomprehension with regards to IEC 61850.	N/A	N/A	N/A
(Apostolov, 2012) Impact of IEC 61850 on the engineering of protection schemes	The paper describes a new methodology to adopt when designing a protection scheme that takes into account IEC 61850 communication capabilities. A particular emphasis is placed on the configuration of SCL files.	N/A	N/A	

[8] Table 2.5 (IEEE/ Key search): IEC 61850 substation

Table 2.6: IEC61850 and modelling

Paper[9]	Emphasis and objectives of paper	Method	Implementation	Performance Evaluation	Achievement in papers	Draw backs and Comments
(Brand and Wimmer, 2003) Modelling interoperable protection and control devices for substation automation according to IEC 61850	The paper discusses modelling methods for control and protection functions. A particular emphasis is placed on the mapping of some protection element to the IEC 60870-5-103 fault indication representation.	Design of a protection scheme based on the process bus.	N/A	N/A	N/A	The paper presents the modelling of a protection scheme using conventional hard-wired method versus the process bus method.
(Kim and Lee, 2005) A study on IEC 61850 based communication for intelligent electronic devices	The paper presents the IEC 61850 basic approach to solve communication problems in substations with emphasis on the time-critical communication message- GOOSE	N/A	Hardware: PC platform (3 PCs) Software: MMS-EASE Lite stack, Ethereal-Network Protocol Analyzer Protection: N/A	The author has demonstrated a GOOSE transmission average of 4 ms	An experiment for GOOSE transmission time is set up using 3 PCs and results are analysed using Ethereal.	The author demonstrates the transmission of GOOSE messages within the accepted range of 4 mS.
(Ozansoy et al., 2009) The Application-View Model of the International Standard IEC 61850	The paper describes the application-view model of the IEC 61850 standard and presents the use of object oriented methodology and techniques for the implementation of the Logical Node (LN) application-view data model of the standard	Object Oriented Programming techniques based on C++	Hardware: PC platform Software: Microsoft C++ Protection: N/A	N/A	Paper has illustrated the implementation of the application view described in the IEC 61850 standard based on OOP techniques	The ACSI services described by means of flow charts enhances understanding, since the application view is very briefly explained in the IEC 61850 series.

[9] Table 2.3 (IEEE/ Key search): Design of microprocessor relays

(Yongli et al., 2009) Study on interoperable exchange of IEC 61850 data model	The paper describes two methodologies to exchange IEC 61850 data model, including the offline exchange based on SCL configuration tools, and the online exchange based on MMS	Offline exchange based on SCL configuration tools. Online exchange based on MMS	Hardware: PC platform Software: Windows OS, MMS-EASE Lite, Tamarack Windows Test Server, MMS-Ethereal Protection: N/A	N/A	The paper has demonstrated the exchange mechanism for IEC 61850 data model via MMS services and SCL file transfer	Enhance the understanding of mechanism for data model exchange between device and configuration tools.
(Aggarwal, 2011) IEC 61850 prototype design	The paper discusses the information model of the IEC 61850 protocol. A prototype design of the information model of IEC 61850 and the study record accessing times is performed with a particular emphasis on database and memory allocation	Real-time main memory database model for IEC 61850. TLSF dynamic memory allocator. Backup scheme for database fast crash recovery	Hardware: PC platform Software: C/C++ Protection: N/A	On-going research.	The paper describes an algorithm that deals with the allocation and management of system memory to data records inside the IED.	The paper enhances the importance of giving attention to data management and data integrity for IEDs.

Table 2.7: Comparison of method for embedded implementation of IEC 61850 requirements

Paper[10]	Emphasis and objectives of paper	Method	Implementation	Performance Evaluation	Achievement in papers	Draw backs and Comments
IEC 61850 Hardware/Software development						
(CESI RICERCA, 2006) Valutazione delle tempistiche associate ai messaggi di tipo GOOSE nell'ambito del protocollo IEC-61850.	The paper describes the development on a PC platform of a GOOSE application from the IEC 61850 standard specification, to a very detailed study of the GOOSE frame structure.	Development of an IEC 61850 based GOOSE application using the development environment SISCO MMS-EASE Lite for Windows	Hardware: Uses two PCs (one publisher and one subscriber) Software: Application written in C++ , using the SISCO MMS-EASE Lite; RTOS-Windows 2000	The implemented solution achieves an average GOOSE transfer time of 4ms.	An in-depth understanding of the GOOSE frame structure and its encoding/decoding rule is achieved.	The predefined stack library could reduce the development time of an IEC 61850 application. Accurate documentation and training for the stack library needs to be accessed.

[10] Table 2.7 (IEEE / Key search): IEC 61850 software"

(Kostic and Frei, 2007) Modelling and using IEC 61850-7-2 (ACSI) as an API	The paper debates the use of the Abstract Communication Services Interface as an Application (ACSI) Programming Interface (API).	Client/Server application using an Application Programme Interface based on the ACSI core.	Hardware: Use of one PC and two IEDs Software: The API-based ACSI is written in C++. It has been tested in a high level IEC 61850 browser application. Protection: N/A	The core ACSI API only defines client-server services currently.	The proposed ACSI API core has been implemented in C++ and tested in a high level application, an IEC 61850 browser	The use of ACSI as a single API for the developer could accelerate the development work in order to add IEC 61850 functionality to a device and with guaranteed interoperability.
(Nguyen-Dinh et al., 2007)A study on GOOSE communication based on IEC 61850 using MMS ease lite	The paper presents a general study of communications in SAS and supplies a detailed analysis of the process of transferring GOOSE messages supported by the MMS-EASE Lite tool from the user's point of view.	Client/Server application using MMS-EASE Lite	Hardware: PC platform Software: C language, MMS EASE Lite; MMS-Ethereal Protection: N/A	N/A	Paper has demonstrated the use of MMS-EASE Lite in order to add GOOSE functionality to a substation application.	N/A
(Zhang et al., 2008) The design of an Object-Oriented Embedded Platform for Substation Data Integration	The Paper presents an object-oriented embedded platform for substation data integration.	Method of Embedded Platform for Substation Data Integration (EPSDI) based on a multi-agent system architecture	Hardware: Embedded platform Software: Linux OS, UML, C++ Protection: N/A	The system build presents good system performance and stability.	The design of an object-oriented Embedded Platform for Substation Data Integration (EPSDI) is presented. System architecture based on MAS (Multi-agent system) technology and a publisher/ subscriber mechanism is proposed.	IEC 61850 software with a multi-agent system architecture presents good system performance and stability.

(Ting et al., 2011) Research of Relaying Protection System of Digital Substation	The paper deals with design methods for software used in protective relay devices to accommodate new technology based on IEC 61850.	Software development based on VxWorks and real-time techniques. Programmable protective function based on the IEC61131-3 standard.	Hardware: PowerPC 8247, ARM7 LCD, FPGA Software: N/A Protection: N/A	N/A	The paper demonstrates that the traditional foreground-background method is no longer suitable when the protection algorithm and the communication functions execute on the same processor, due to limited interrupt resources.	Assuming a single-chip microcomputer, the multitasking ability of the Real Time Operating System, allows for different software modules to execute independently. This increases stability and the overall response of the system. This approach is said to be more efficient compared to the traditional foreground-background method (interrupt-driven).
(Yang et al., 2011) Programmable Logic for IEC 61850 Logical Nodes by means of IEC 61499	The paper deals with the use of IEC 61499 (an open standard for distributed automation and control) in conjunction with IEC 61850 in order to enable online reconfiguration of a systems control module as well as communication parameters.	- Intelligent logical node (iLN) architecture - 52 Blocking	Hardware: PC platform, 3 controller based ARM7 Software: MATLAB and SIMULINK, ISaGRAF, SystemCorp PS-10 stack	co-simulation environment and the hardPLC of the ISaGRAF have demonstrated the expected result as the one for 52 Blocking application	- single line diagram modelled in MATLAB/SIMULINK - 3 microprocessors used to demonstrate execution at hardware level. - And a virtual device on PC to bridge with MATLAB /SIMULINK	- The IEC 61850 standardizes the data and communication in substation design, but the implementation of control has been left intentionally out of the scope of the standard.
(In et al., 2011) Design of one chip communication stack processor and MMS communication stack library based on IEC 61850	The paper deals with implementation of a MMS communication stack library and the development of an IEC 61850 one-chip communication stack processor of an IED. The single chip solution is composed of two processors with an efficient VHDL architecture for the development of IEC 61850 compliant devices.	Single chip design using multi-microprocessor	Hardware: PC platform, Uses 4 independent processor modules: IEC 61850 communication; microprocessor for logic process control; processor for CT/VT measurement; and Protection algorithm. Software: C++,, Protection: N/A	Supports only MMS (no support for GOOSE). Additional performance tests are underway.	A prototype IED board has been built that makes use of the one-chip solution for logic control and IEC 61850 communications. A method composed of two processors embedded into one single chip is described in order to increase reliability and accuracy of signals between the communication processor and the logic processor.	The use of the specific IEC 61850 libraries in MMS traffic simulation reduces the user’s development time. The one-chip solution ensures low cost, low power consumption and helps reduce the size of IEDs.

(Shujian et al., 2011) The research based on IEC61850 intelligent terminal of substation	The paper emphasis is on the development of an Intelligent Terminal Unit.	Method of ARM+ DSP dual-CPU structure	Hardware: ARM7 microprocessor, CPLD Software: C++ Protection: N/A	N/A	Requirement details for the design of an intelligent electronic device are given. Structure and function of different hardware modules are also explained.	Hardware- software modularization is suitable for IEC 61850 embedded development and enhancement
(Park et al., 2012) IEC 61850 Standard Based MMS Communication Stack Design Using OOP	The paper describes the development of an MMS communication stack library using Object Oriented programming techniques.	Object-oriented programming (OOP) based on C++ language for MMS stack design.	Hardware: PC platform Software: Microsoft Windows OS, C++ language	N/A	Paper shows the development of an MMS communication stack library based on IEC 61850 using C++ programming language	Using the proposed method, developers can understand and deal with applications more easily and have the advantage of maintenance, and module reuse.

Table 2.8: Comparison of method for evaluating and testing protection systems based on GOOSE messages

Paper[11]	**Emphasis and objectives of paper**	**Method**	**Implementation**	**Performance Evaluation**	**Achievement in papers**	**Draw backs and Comments**
Substation protection based Generic Object Oriented Service Event (GOOSE)						
(Zhang and Nair, 2008)Testing protective relays in IEC 61850 framework	The paper reviews the conventional protective relay testing methods and proposes new test approaches for testing the IEC 61850 features of relays.	GOOSE message speed test. Interoperability test GOOSE buffer managing ability test	Hardware: Real IEDs, PC Software: Vendor specific Protection: N/A	Bad performances for GOOSE transfer time (average of 14.70 ms). The author states that this could have been caused by poor performance of the communication processor	A test bench is proposed to test the GOOSE communication capability via a GOOSE speed test, and an interoperability test and buffer managing ability test.	It has been observed that GOOSE message performance largely depends on the CPU state of art and structures of communications modules. Also proprietary device configuration tools play an important role in interoperability.

[11] Table 2.8 (IEEE / Key search): “Protection GOOSE”

(Sidhu et al., 2010)Packet scheduling of GOOSE messages in IEC 61850 based substation intelligent electronic devices (IEDs)	The paper presents different packet scheduling algorithms for SAS IEDs using an OPNET simulation tool. The effect of the different packet scheduling schemes in IEDs on GOOSE message delays is analyzed in detail.	FIFO scheme. Strict Priority. Round Robin. Weighted Round Robin.	Hardware: PC platform Software: OPNET simulation tool- Protection: N/A	The simulation results show that the presence of a strict priority scheme in the IEDs causes a considerable reduction in the delay of GOOSE messages from 4.1476 mS to 0.50999**msec**	Paper has demonstrated that the use of FIFO technology in substations for IEDs could not allow to meet the time requirement ranging from3 to 4 mS.	For an IED designer, It is very important in order to meet some IEC 61850 requirements to consider the network card capabilities with respect to packet scheduling as this has an impact on the average system response time.
(Bonetti and Douib, 2010)Transfer time measurement for protection relay applications with the IEC 61850 standard	The paper investigates a practical method to measure the GOOSE transfer time and the total operation time of the direct inter-trip relay application.	Hardwired trip. GOOSE trip (loop back transfer time and direct inter-trip relay application).	Hardware: 2 commercial IEDs. Software: N/A Protection: N/A	The results demonstrate that GOOSE Trip technology is typically 3ms faster than hardwired technology.	The author proposed the use of the inter-trip test instead of round trip test for GOOSE transfer time measurement. Corresponding model setup is explained and demonstrated.	The author has demonstrated that the GOOSE trip signal is faster than hardwired method (3ms faster). But the IED application time has in impact on the total inter-trip scheme performance.
(Steinhauser et al., 2010) Performance Measurements for IEC 61850 IEDs and Systems	Paper describes methodology for GOOSE performance evaluation and method used by OMICRON engineers for accurate measurement of the GOOSE transfer time	Round-Trip Test. Rally Test.	N/A	N/A	The paper describes a procedure test formula used to determine different GOOSE performance evaluation times.	N/A

(Ali and Thomas, 2011)GOOSE based protection scheme implementation and testing in laboratory	The paper demonstrates the practical implementation and testing of protection schemes based on high-speed peer-to-peer communication using a GOOSE message model in a laboratory setup.	Distributed protection based GOOSE.	Hardware: PC platform, Real IEDs, CMC 256 Software: Test Universe (OMICRON) Protection: N/A	The test set up demonstrates the use and test of GOOSE message in the range of 4 ms for transfer time.	Paper demonstrates the realisation of a lab setup environment for GOOSE investigation with advanced tools from OMICRON.	The lab-test environment setup could be used for demonstration of GOOSE applications with appropriate test equipment
(Apostolov and Vandiver, 2011) IEC 61850 GOOSE applications to distribution protection schemes	The paper discusses the requirements for reduction in the duration of different short circuit faults with a particular emphasis on distributed protection schemes.	Peer-to-peer communication-based distributed bus protection	Hardware: Real IEDs Software: Test Universe Protection:- Sympathetic trip protection, Selective backup tripping , Breaker failure protection	The application of IEC 61850 GOOSE messages allows for reduction of the fault clearing times and minimizing the effect of short circuit	The paper demonstrates the benefit of using distributed protection scheme-based GOOSE instead of conventional hardwired technology.	For distributed protection scheme-based GOOSE, particular attention should be paid to test equipment and the publishing/subscribing mechanism.
(Sidhu et al., 2011b) Configuration and performance testing of IEC 61850 GOOSE	The paper deals with the use of GOOSE for a time-critical application, configuration of Ethernet switched network, and configuration of GOOSE within IED and system integration.	Hardwired trip. GOOSE trip.	Hardware: RTDS (GTNET); real IEDs Software: CACE Pilot; Wireshark Protection: N/A	Results demonstrate the reliability when using GOOSE under different background traffic. The GOOSE transfer time has remained within the range of 3 to 4 mS.	An experimental lab is set up to measure the round trip delay of GOOSE. The GTNET card from an RTDS is configured in promiscuous mode to capture the entire network traffic.	The observed trip time of GOOSE is almost equivalent to hardwired copper based traditional trip signal. Also, the use of IEEE 802.1Q frame format enhanced the performance of GOOSE over the network.

2.3 Comparative Literature review

From the diverse groups of papers presented in section 2.2.2, points that emerge from the literature review are:

- distinct device technology epochs (already considered in Section 2.2.1),
- structural design of microprocessor/microcontroller based relays,
- basic concepts of IEC 61850,
- IEC 61850 embedded implementation and performance evaluation of protective system based on GOOSE communication.

The sub-sections below elaborate further on the points mentioned above.

2.3.1 Structural design of microprocessor/microcontroller based relays

Technical documentation on the design of microprocessor/microcontroller based relays is usually not available in the public domain. Multi-national companies classify their manufacturing designs as being 'Top Secret'. This means that this classified information is inaccessible to the general public or researcher.

Notwithstanding this limitation of access to information, the IEEE website has some documents that provide clear insight into the structural arrangement of microprocessor-based relays. Table reports the work of Mansour and Swift (1986) on multi-microprocessor design-based protection relays. They propose a microprocessor method arrangement that implements a "two level master-slave hierarchical system with only one master and three slaves". The implemented protection algorithm is based on the "Karrenbauer modal transform" method. Their work demonstrates that using multi-microprocessor–based relays could increase the performance and system response of the protection relays. However, choosing a multi-microprocessor over a single microprocessor implementation approach depends on other factors such as processing power, system complexity, system integration facilities and final cost.

The work of Baigent and Lebenhaft (1991) on the design of microprocessor based relays provides additional information in terms of design specification for a microprocessor-based relay system. Zhu et al. (2006), has developed an interesting method for computer simulation of characteristics of microprocessor-based relays. The relay model is developed using the software language FORTRAN 77. The fact that the relay model as a whole can be simulated in advance allows for the detection of software design issues at an earlier stage before hardware implementation.

The work of Sahai and Pandya (2011) demonstrates a complete and more recent contribution into the design and assembly of microprocessor-based relays with functional modules. All stages that need to be considered in the design of a modern microprocessor relay are discussed from a modular perspective.

To partially conclude, there are different elements that need to be considered when developing an IED (protective relays) such as:

- Functions to be realized (protection, control measurement and monitoring, etc.);
- Processor structure arrangement and operating frequency (algorithm synchronisation, communication, etc.);
- Input/output (I/O) card design by means of galvanic isolation method (usually optical isolation);
- CTs and VTs interface design according to a pre-determined rated voltage.
- Integration of peripheral components (memory, ADC, I/O card unit, etc.) to the microprocessor core structure is usually done via the use of Complex Programmable Logic (CPLD).

The next section deals specifically with the communication aspects related to substations.

2.3.2 Effort of communication standardisation

After the introduction of what can simply be called microprocessor-based relays (digital and numerical relays), the number of open and proprietary protocols for communication simply exploded. By the 1980's, there are more than 100 proprietary protocols of communication in use in substations (Nordell, 2003). This makes it difficult to achieve interoperable communication between different vendor's equipment within the substation. Thus this requirement for standardisation of a communication protocol became the next target for the national and international bodies of standardisation (American National Standards Institute (ANSI), National Institute of Standards and Technology (NIST), IEEE, International Organization for Standards (ISO) and IEC). In 1986, the Electric Power Research Institute (EPRI) started this standardisation work that produced the first version of UCA (Utility Communication Architecture) six years later. More details on the standardisation effort are presented in the following section.

Table 2.4 reports the work of Dick (1996), Chaturvedi (2002) and Uzair (2008) in the early efforts at standardisation of substation communication. These efforts resulted in different steps toward standardisation. Table 2.9 which does not claim to be exhaustive, reports on these different steps.

Table 2.9: Communication standard evolution

Date of publication	Protocols or standards	Comments
1990-1995	IEC 870 series (Telecontrol communication protocol standards for Electric Power)	Harmonisation to some set of specific protocols to be used in SCADA systems.
December 1991	UCA 1.0 (Utility Communication Architecture)	Target interoperability based on object-oriented communication architecture.
November 1993	DNP3 (Distributed Network Protocol)	Derived from IEC 870-5-3 and 870-5-4 draft documents. DNP is optimized for serial communication.
December 1998	UCA 2.0 (Utility Communication Architecture)	Refinement of UCA 1.0
2003-2005	IEC 61850-Ed.1 (Communication networks and systems in substations)	Derived from UCA. Target interoperability, communication exchange among field devices, substation configuration language and much more. Further information is provided in CHAPTER One:
2011-2012	IEC 61850-Ed2 (Communication networks and systems for power utility automation)	Refinement and extension of IEC 61850-Ed.1 to not only the substations, but also the entire power utility system.

The introduction of the IEC 61850 standard has brought about many challenges for relay manufacturers as well as to research institutes. Manufacturers develop IEC 61850 compliant devices, where they have the obligation to ensure backward compatibility with previous technologies. Almost no research (or even teaching efforts) has been made in the academic sphere.

The next section reviews the references for different IEC 61850 methods of implementation by means of embedded systems. The basic concepts within the IEC 61850 standard are dealt with in Chapter 3.

2.3.3 Analysis of different methods for IEC 61850 implementation on embedded platforms

Implementation of the IEC 61850 standard onto an embedded platform requires detailed knowledge of the target platform, but it also requires a comprehensive understanding of the standard itself. The IEC 61850 standard is a multi-faceted standard that refers to many other standards (or common practices) depending on the considered issue.

Field experts when implementing the IEC 61850 standard on a specific embedded platform refer to a communication stack for all communication within the IEC 61850 standard. Secondly, the Substation Configuration Language (SCL) capabilities of the system are assessed to describe the line diagram and provides basic configuration for communication purposes. Any other aspect relating to the development of intelligent electronics devices is beyond the scope of the IEC 61850 standard series (e.g.

protection algorithm implementation, single or multi-microprocessor implementation, programmable logic control, etc.).

Table 2.7 details various projects on the IEC 61850 standard with an embedded implementation in terms of the software and hardware arrangement. An embedded implementation of the IEC 61850 standard can be approached using either of two methods. The first method according to the work of CESI RICERCA (2006)and Nguyen-Dinh et al. (2007), is the development of an IEC 61850 software application using an off-the-shelf (commercial) communication stack. The second method is the development of a personal IEC 61850 communication stack and successful integration into a specific embedded platform and is based on the work of Kostic and Frei (2007), Zhang et al.(2008), Shujian et al. (2011) and Park et al. (2012).

The computational speed and memory required for the deployment of control and protection algorithms of various complexities are also points that need consideration. The system integrator or engineer decides whether to use a multi-microprocessor or a single microprocessor system based on the complexity of the system.

The work of In et al. (2011) suggests the integration of a multi-microprocessor approach into a single chip solution in order to improve system performance. Ting et al. (2011) proposes a real time multi-tasking operating system approach. In this approach, a single microprocessor can still be used based on its processing speed (PowerPC) and on the multi-tasking ability of the RTOS that runs on it. Special attention needs to be paid to the software development environment. Usually, the programming language is chosen based on its ability to handle the communication stack integration process, the protection algorithm deployment process, and any other software module that might be considered as being part of the system integration.

The work of Yang et al. (2011) draws attention to the fact that programmable logic control is not considered to be within the scope of the IEC 61850 standard. Therefore, there is a requirement to refer to other standards such as the IEC 61499 (for distributed control systems). This work demonstrates the use of the IEC 61850 standard in conjunction with the IEC 61499 standard in order to enable the development of complete automated devices and system solutions. The choice of a single chip that includes a single microprocessor (SC143-chip from Beck-IPC) appears to be appropriate for the current research project. It is chosen due to the fact that it is one of the first embedded platforms commercially available for experimentation outside of vendor companies.

The next section reviews the references associated with the GOOSE communication protocol and evaluation of its performance.

2.3.4 GOOSE performance evaluation

The IEC 61850 standard has inherited the notion of Generic Object-Oriented Substation Event (GOOSE) communication from the Utility Communications Architecture (UCA). Most of the distributed protection schemes in use in IEC 61850 substations are based on the GOOSE communication protocol. GOOSE messages are generally used to rapidly exchange time-critical data in order to clear a fault and isolate the damaged area. During the implementation of a protection scheme based on GOOSE, the system engineer needs to ensure that in the case of an electrical fault, the protection scheme functions within a reasonable amount of time.

Different methods exist that evaluate the response time of a protective system when a fault occurs. There are also methods to evaluate the precise GOOSE transfer time. It is important to ensure that GOOSE transfer time is within the allowed time as this has a direct impact on the global system response time.

Table 2.8 cites recent research for various test methodologies to evaluate performance of GOOSE messaging in distributed protection schemes.

Zhang and Nair (2008) have demonstrated the importance of having an appropriate Central Processing Unit (CPU) processor and well designed software modules as these can introduce delays in the system response that in turn degrades the GOOSE transfer time. Also, Sidhu et al. (2010) illustrates the importance of the network interface card and the Ethernet driver capabilities with respect to packet scheduling as this also has an impact on the average system response time.

According to the work of Steinhauser et al. (2010) two methods can be utilized with regard to GOOSE transfer time measurement: the roundtrip test (ping-pong or loop-back test) and the rally test. The roundtrip test is widely used while measuring the GOOSE transfer time. A current and voltage signal injection test set can be used for this purpose. Using this method requires that some assumptions be made in order to reach an accurate result. The rally test on the other hand does not require the use of a test set, but it is more difficult to implement since it requires two devices to be under test at the same time. This test also requires an accurate mechanism to measure the duration of the rally test and has a counter for the number of loops that have completed during this time interval.

Sidhu et al. (2011) demonstrates via the use of the Real-Time Digital Simulator (RTDS) the measurement of the round-trip delay and concludes that the distributed protection system based on GOOSE communication is on average 2ms faster than the conventional hardwired technology.

Bonetti and Douib (2010) have proposed another test set-up called direct inter-trip test. This method is similar to the rally test, but uses a test set in addition to two separate relays. This approach takes into consideration the relay protection element applications. Also Ali and Thomas (2011) have demonstrated the use of specialised tools such as the OMICRON configuration toolbox delivered with the Test Universe software. The work developed by Apostolov and Vandiver (2011) extends the theoretical use of GOOSE to an implemented case scenario.

As can be seen later in this thesis (Chapter 6), the inter-trip test has been chosen over the basic round-trip test. This presents the possibility to observe the overall system response time to a simulated fault condition and at the same time provides the possibility of deducing the GOOSE transfer time.

2.4 Conclusion

Chapter 2 presents the project background and basic knowledge of the current situation in substation automation systems. The challenge to locate well-documented online literature can be explained by the fact that the IEC 61850 standard is fairly novel. Its multidisciplinary aspects (Protection, Data Networking, Embedded Systems, etc.) make it especially difficult for electrical engineering departments at universities to develop and implement devices which are IEC 61850 compatible.

Today, there are non-standardised IEDs (not IEC 61850 compliant) and standardised IEDs currently in use in modern substations. There are also substations that are in various stages of evolution. Some substations have limited communication capabilities while others employ self-diagnostics and problems can be anticipated and corrected before the operators even know a problem exists (Wolf, 2009b).

The following question can be asked: Why is there a trend towards greater substation automation? The reasons can be formulated as follows:

- Requirement for acquisition, distribution, and management (with network security) of large amounts of real-time data (non-operational and operational).

- Requirement for standardized, multifunction interoperable devices (to reduce installation, maintenance and upgrade capital expenditures).
- Use fewer devices to wire and configure (reduce cabling cost).
- To develop systems which are addressable, “Plug and Use” with self description capability (reduce initial installation cost), and subsequently reducing the reconfiguration, scalability and maintenance costs.
- Able to integrate the monitoring and protection functions at the SCADA level. (Nelson, 2005).

From the literature research conducted it appears that system integration and interoperability of an embedded device requires detailed understanding of the IEC 61850 standard and its technological impact on the modern substation automation system as well as the design of the computational platform within modern IEDs. The development of the embedded device also requires detailed knowledge of the hardware/software target platforms including the Integrated Development Environment (IDE) software to be used. An efficient method for the development of an IEC 61850 compliant device is to use a communication stack appropriate for the hardware target, as this reduces the development complexity and shortens the time-to-market.

Only a few companies have developed communication stacks that are compliant with IEC 61850 and can be deployed on embedded platforms. These software communication stacks are said to be hardware independent, however their deployment onto a specific hardware platform still requires significant integration work and a detailed understanding of the hardware and the Real Time Operating System (RTOS) deployed on the hardware target. Due to on-going developments of these communication stacks it is vitally important that version revisions and functional capabilities are kept track of as deployed on the various platforms. Not doing so may lead to interoperability issues.

The main project goals are to successfully deploy a protection scheme based on GOOSE messaging by making use of the PIS-010 communication stack. This is demonstrated using commercial IEDs from vendors in conjunction with the DK60 board from Beck-IPC. As stated earlier the deployment of such an embedded application requires comprehensive knowledge of the IEC 61850 standard as well as the system target platforms. Detailed understanding of the field of real time embedded software development, object-oriented programming, networking, system modelling, electrical protection and system simulation is also required.

Chapter 3 provides more details on the IEC 61850 standard and focuses on the use of the GOOSE protocol for protection specifically.

CHAPTER THREE:
ANALYSIS OF THE IEC 61850 STANDARD

3.1 Introduction

The integration, adaptation or adding of IEC 61850 functionality to modern substation devices requires detailed knowledge of the IEC 61850 standard and other related standards.

This chapter provides a practical overview of the IEC 61850 standard (Edition 1). A brief history leading to the development of the IEC 61850 standard is presented in section 3.1.1 while section 3.1.2 addresses the SAS deficiencies prior to the advent of the IEC 61850 standard. The basic IEC 61850 standard philosophy and objectives are presented in section 3.2. The IEC 61850 modelling concept is presented in Section 3.3 where particular emphasis is placed on object-oriented terminology and the IEC 61850 data model. This section also explains the Abstract Communication Service Interface (ACSI). Details with regards to communication aspects are discussed in Section 3.4. Section 3.5 details the Substation Configuration Language (SCL). Section 3.6 provides an overview of the other related standards to consider in this research work. Conclusion - section 3.7.

3.1.1 Brief substation automation evolution review

As described in Chapter Two, electro-mechanical devices - although having served the industry reliably over decades, have the following drawbacks:

- Require significant cabling;
- Single function devices only; and
- Not addressable, etc.

Non-standardized IEDs (not IEC 61850 compliant) have been introduced in Substation Automation Systems since the start of the 1980’s (Lundqvist, 2003). The consequence of a non-standardized environment is that vendors have adopted differing approaches to substation automation systems with diverse and proprietary communication protocols and equipment being deployed (Nordel, 2005).

One of the main challenges that electric power facilities are currently facing, is the issue of interoperability among Intelligent Electronic Devices (IEDs) supplied by different vendors during the expansion of an existing substation. Electrical engineers initially developed the “Universal Translator” (data concentrator) (Wolf, 2009a) to take data from different generations of sensors and devices found in older (legacy) substations and convert them into a suitable format.

Another concern in attempting to ensure interoperability among IEDs supplied by different vendors and the issues arising as a result of attempts at integration of legacy devices is the inability to dynamically address IEDs. In 1995 the IEC (International Electrotechnical Commission) after identifying the relevant challenges within substations, initiated the development of a communication standard that all vendors of substation devices would implement (IEC 61850, 2010). A special working group (WG10) was created inside the IEC Technical Committee 57 (TC57[12]) that consisted of 60 members from different countries (IEC 61850, 2010).

Issues in substation automation systems that the WG10 group consider are: a common architecture network; common communication protocol; configuration support; common method/format for storing complete data; interoperability; exchangeability and fast communications among field devices.

Another specific concern that this group addresses is the communication network and systems within the electrical substation in order to increase the reliability of the European effort (IEC 60870-5) (Digital Bond, 2010).

The following section retraces the efforts directed at standardization that had already begun long before the WG10 had been established.

3.1.2 Addressing Substation Automation deficiencies

According to the Digital Bond (2010) website, the history of the IEC 61850 standard in addressing substation automation deficiencies can be detailed as follows: “In 1988 EPRI and IEEE initiated the Utility Communications Architecture (UCA) project under the Integrated Utility Communication (IUC) program. The objective of the UCA project is to make provision for interoperability between control systems employed to monitor and control the electric power utilities. Initially the UCA project focused on communication between control centres, and communication between substations and control centres. EPRI and IEEE carried out the UCA project in collaboration with the Pacific Gas and Electric Company and the Houston Light and Power Company. The result of this collaboration was a standard communications architecture referred

[12] The TC57 is responsible for maintaining the communication standards for the power utilities (Montignies et al., 2011a).

to as UCA version 1.0. But this standard did not provide a detailed description of how the UCA communication architecture was to be practically implemented and used in field devices. In 1997 EPRI and IEEE joined efforts with Working Group 10 (WG10) of the IEC Technical Committee 57 (TC57) to build a common international standard for electrical utility communications. These efforts lead to UCA version 2.0 in 1998, with thorough specifications of object models of field devices".

The efforts of the Working Group 10 (WG10) of the IEC Technical Committee 57 (TC57) - based on concepts and definitions of the UCA architecture, lead to the creation of a standard for the design of electrical substation automation named IEC 61850 (Digital Bond, 2010).

The IEC 61850 standard (Edition 1) consists of the following fourteen parts presented in Table 3.1 and published between 2003 and 2005 (IEC 61850, 2010):

Table 3.1: Composition of the IEC 61850 standard (Edition 1)

Part number	Title
Part 1	Introduction and overview
Part 2	Glossary
Part 3	General requirements
Part 4	System and project management
Part 5	Communication requirements for functions and device models
Part 6	Configuration language for communication in electrical substations related to IEDs
Part 7-1	Basic communication structure for substation and feeder equipment - Principles and models.
Part 7-2	Basic communication structure for substation and feeder equipment - Abstract communication service interface (ACSI).
Part 7-3	Basic communication structure for substation and feeder equipment - Common Data Classes.
Part 7-4	Basic communication structure for substation and feeder equipment - Compatible logical node classes and data classes.
Part 8-1	Specific communication service mapping (SCSM) - Mappings to MMS (ISO/IEC9506-1 and ISO/IEC 9506-2) and to ISO/IEC 8802-3.
Part 9-1	Specific communication service mapping (SCSM) - Sampled values over serial unidirectional multidrop point to point link.
Part 9-2	Specific communication service mapping (SCSM) - Sampled values over ISO/IEC 8802-3.
Part 10	Conformance testing.

The next section presents the philosophy behind the IEC 61850 standard and its main objectives.

3.2 IEC 61850 philosophy and objectives

According to the Digital Bond (2010) website: “The IEC 61850 standard was designed to provide a robust architecture network, common communication protocol suite, common data format and naming convention, interoperability, fast communications among field devices, guaranteed data delivery within a pre-defined time, configuration support, and defines complete testing requirements for substation equipment”.

It is a standard that encompasses multiple disciplines, and that makes use of existing standards as well as commonly accepted communication principles and accepted protection methods. The IEC 61850 standard is the primary source of information, for this research work and has been used as a guideline for implementation and adaptation.

To summarize, the IEC 61850 was driven by three main philosophical concepts which are:

- Virtualisation: Development of a method to create a generic substation model of all relevant components and functions (Section 3.3 on Conceptual Modelling philosophy, page 43).
- A method which allows flexibility for future communication needs by incorporating service and mapping mechanisms which will accommodate this. (Section 3.4 on Data communication philosophy, page 58).
- Exchange of information through XML files for device capability and system architecture needs for engineering the Substation Automation System (SAS) (Section 3.5 on System configuration philosophy, page 65).

The points listed above can be viewed as the most pertinent aspects for different institutional bodies or individuals interested in the IEC 61850 standard.

Table 3.2: IEC 61850 and different roleplayers

Order of interest	Philosophy driving keys	Concerned Person or institution	Comments
1	Method to create a generic model of substation	Main substation equipment manufacturer and individual field expert (e.g. ABB, SIEMENS, SEL…)	These institutions are qualified to propose or suggest new device models to be included in future releases of the standard. They are generally members of the IEC Technical Committee 57 (WG10)
2	Definition of the exact form of communication to be supported.	1- Main communication stack developer (e.g. SISCO, TRIANGLE WORKS, SYSTEMCORP)	1- These institutions are responsible for the implementation of the standard into a concrete protocols suite.
		2- Substation Application developer	2- These software engineers are responsible for the integration of the communication stack into a specific software application to provide devices with IEC 61850 communication capabilities.
3	SAS configuration reconfiguration and Testing	Protection/ commissioning engineer.	Persons responsible for substation commissioning and future maintenance of the system

The objectives of the IEC 61850 standard can be stated as follows:

- Interoperability between all communicating equipment;
- Free Configuration of the system;
- Long Term Stability with respect to technological evolution.

The following section briefly examines each of the IEC 61850 philosophies introduced above. Each philosophy is directly followed by its supporting structure.

3.3 Conceptual Modelling philosophy

The IEC 61850 standard allows for a physical substation to be modelled (Section 3.2) into a virtual substation (Hammer and Sivertsen, 2008). Figure 3.1 below, illustrates the conceptual modelling approach of the IEC 61850 standard which consists of the modelling of all components and functions in the substation to clearly define their logical structure and behaviour. The modelling within the IEC 61850 standard is based on the Unified Modelling Language (UML) standard. The virtualisation of SAS components and its functions are the first steps in the IEC 61850 modelling approach which produces the Abstract Communication Service Interface (ACSI). The ACSI layer is explained in Section 3.3.3. Secondly, the IEC 61850 standard defines the communication mapping procedure which is explained in Section 3.4.1.

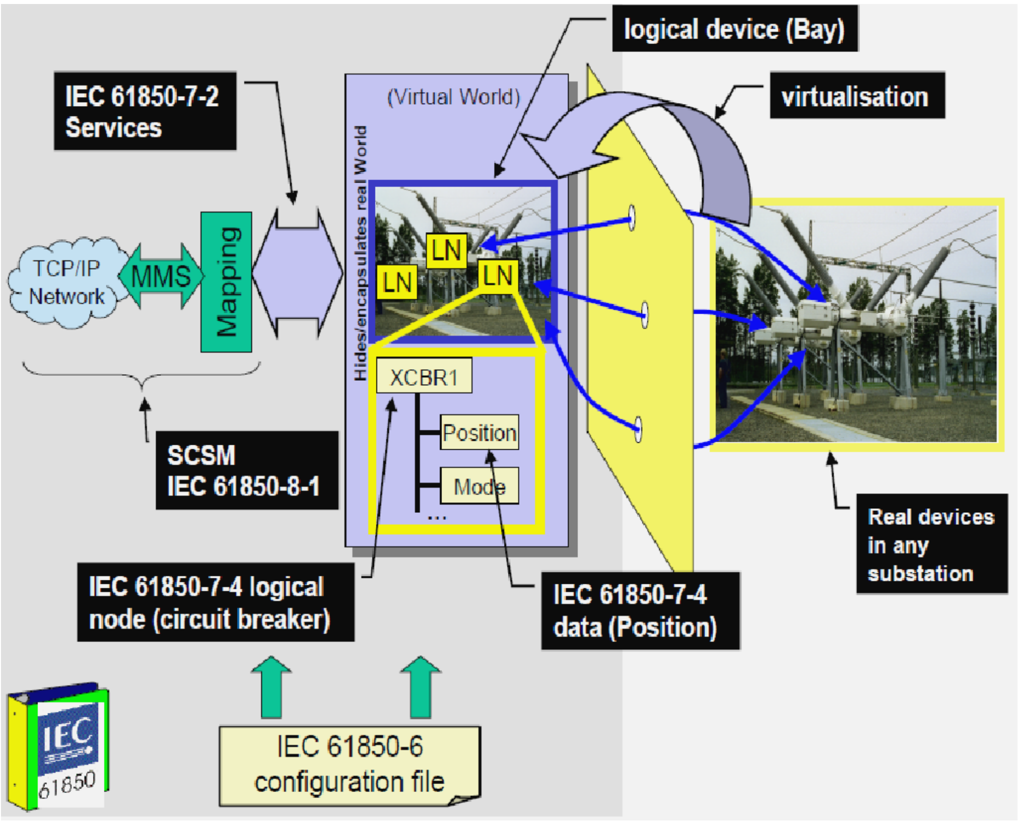

logical device (Bay)
virtualisation
IEC 61850-7-2 Services
(Virtual World)
Hides/encapsulates real World
TCP/IP Network
MMS
Mapping
LN
LN
LN
XCBR1
Position
Mode
...
SCSM IEC 61850-8-1
Real devices in any substation
IEC 61850-7-4 logical node (circuit breaker)
IEC 61850-7-4 data (Position)
IEC 61850-6 configuration file
IEC 61850

Figure 3.1: IEC 61850 conceptual modelling approach
(IEC 61850-7-1, 2003, p.15)

The virtualisation is not only a philosophical concept that represents the modelling foundation of the IEC 61850, but it has been an engineering process in which field specialists model existing substation automation equipment and functions. Within Edition 1 of the IEC 61850 standard most of the functional elements or device functions have already been modelled.

The virtualisation is a process executed in order to standardize a SAS device or function. Once a particular device or function has been standardised there is no need for one to reinvent the wheel. Instead, the considered device or function is clearly defined in the IEC 61850 standard and shall be used as such. However, apart from the mandatory data required for the modelling, system engineers still have to go through the virtualisation process in line with the IEC 61850 specification in order to refine a particular model by the inclusion of appropriate Optional Data that help to enhance the device or functional description.

The prescribed specification of Data Models (Logical Node Classes and Data Classes), with the specific naming convention of the IEC 61850 standard allow for interoperability. Most of the UML diagrams in the IEC 61850 standard define the data structure with different levels of abstraction in order to specify the IEC 61850 object model.

The standard does allow for expansion and addition of any new model if there is a need to do so using the same virtualisation process and appropriate procedures to include the new model into a future release of the standard. These procedures are outside the scope of the current research project. Additional information on the development of new IEC 61850 Data models for a device or substation function can be found in the IEC 61850 standard.

From Figure 3.1, one can see that the building blocks and main objects of IEC 61850 are logical nodes (LN). It is the logical node that allows/caters for virtualisation of substation components into a suitable data model for software engineering. A logical node consists of Data Objects (DO), and each DO has a certain number of Data Attributes (DA).

Within an IEC61850 compliant substation, IEDs are considered to be server devices that provide certain client services (e.g. reporting, GOOSE), but some IEDs do implement client functionality as well. The standard only defines the server role (logical nodes, data, control, etc.) located in the server, and the service requests exchanged. The client role is complementary [13](IEC 61850-7-1, 2003, p.56) , but not fully considered in Edition 1 of the standard.

The Abstract Communication Service Interface (ACSI) defines the data model and specific IEC 61850 services delivered by a server-object. As stated earlier, the data model and services in IEC 61850 are based on an object-oriented[14] approach that makes use of the Unified Modelling Language (UML). The ACSI layer is explained in greater detail later in Section 3.3.3.

The IEC 61850-7 part of the standard contains some ready-to-use prototypes which are organised in terms of Logical Node Classes, Data classes and Common Data Classes. This means that the complete virtualisation process is not required for

[13]Clients and their internal structure and functions are not defined in the IEC 61850 Ed1.standard.

[14] The fact that IEC 61850 uses an object-oriented modelling approach to define substation components does not imply that the IED's firmware has to be written using object-oriented programming languages such as Java or C++.

devices and functions that have already been standardised such as the Circuit Breaker.

There are more than 91 Logical node classes that have been defined and grouped into 13 logical node groups with respect to their substation functionality. Table 3.3 gives the list of Logical Node Groups (and Prefixes used in association with them) as specified in the IEC 61850 standard (Edition 1):

Table 3.3: List of Logical Node Groups

Group Indicator	Logical node groups
A	Automatic Control
C	Supervisory control
G	Generic Function References
I	Interfacing and Archiving
L	System Logical Nodes
M	Metering and Measurement
P	Protection Functions
R	Protection Related Functions
S	Sensors, Monitoring
T	Instrument Transformer
X	Switchgear
Y	Power Transformer and Related Functions
Z	Further (power system) Equipment
A	Automatic Control

3.3.1 Object-Oriented constructs for Modelling within IEC 61850

The IEC 61850 standard refers to concepts of object, class, attribute (properties or states), inheritance and methods (function) which are common object-oriented concepts. Object-oriented concepts are used in the IEC 61850 to model actual physical-devices and associated substation functions with respect to the data-view and the communication-view.

According to the IEC 61850 series, a substation function is a task performed by the substation automation system. The substation functions can be distributed onto many devices (IEDs) where the smallest component of the function (logical node) communicates with the rest of the function, or the substation functions can be deployed onto a single IED, meaning that all logical nodes needed for its realization are localized in the same device i.e. the standard has the mechanisms to accommodate centralized or distributed system designs and implementation.

Most substation devices and basic functions[15] have already been modelled within Edition 1 of the IEC 61850 standard. However, if a relevant substation function or device has not been modelled, there are procedures that clarify how this may be accommodated in IEC 61850-7 of the standard.

The building blocks used to define the logical structure of a device within the IEC 61850 is referred to as a logical node (LN). The logical node is the smallest part of a substation function that exchanges data. From an object-oriented programming point of view, a logical node is an object defined by its data and methods (IEC 61850, 2010). Figure 3.2 provides an overview of the logical node and its corresponding data elements encapsulated inside the physical device.

The container of the logical model is the Physical Device that contains one or more Logical Devices, each of which contains one or more Logical Nodes, each of which contains a pre-defined set of Data Classes, each of which contains data (Proudfoot, 2002).

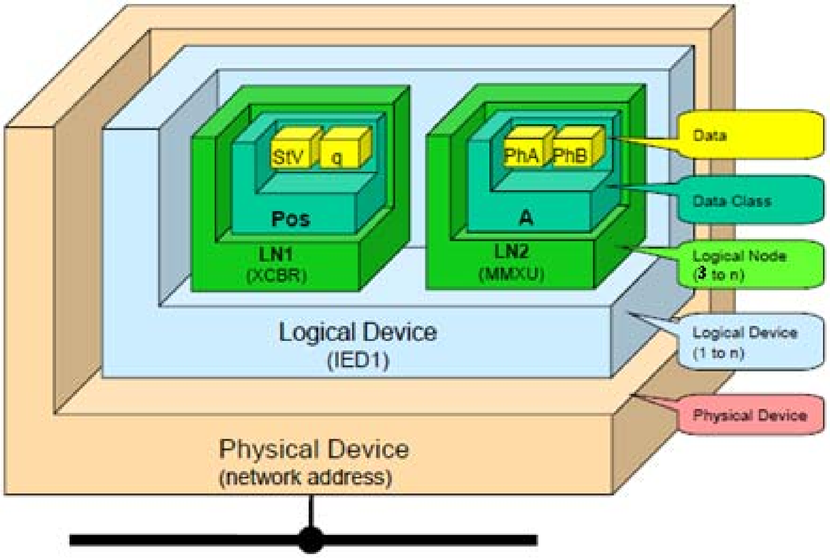


Figure 3.2: Encapsulation principle of IEC 61850
[Adapted from (Proudfoot, 2002)]

As a real world example, consider a dealership that sells vehicles. The vehicle is considered to be an object. In most dealerships, there is more than one brand of vehicle (Mercedes, BMW, Toyota …). Vehicles can be categorised by means of *class* with respect to their brand. Each vehicle has its own engine that can be viewed as an

[15] The term "substation function" does not have the same meaning as "object function" which refers to methods applied at a higher abstraction level to an abstract object in the software code.

object on its own. Each engine has *attributes* such as engine speed and torque. A *method* to change the speed of the engine is by changing the position of the accelerator pedal or the brake. Figure 3.3 describes the analogy between the IEC61850 data model and the real world example considered above.

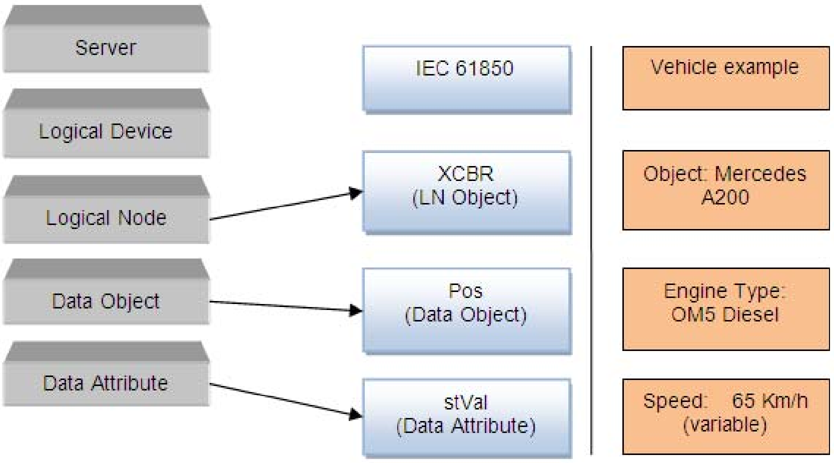


Figure 3.3: IEC 61850 Data Model vs real world example

Figure 3.4 provides a hierarchical view of a logical node's (LN) internal elements using the XCBR LN for Circuit Breakers as an example. Note that data information (attributes) within a data-object is grouped by means of Functional Constraints (FC).

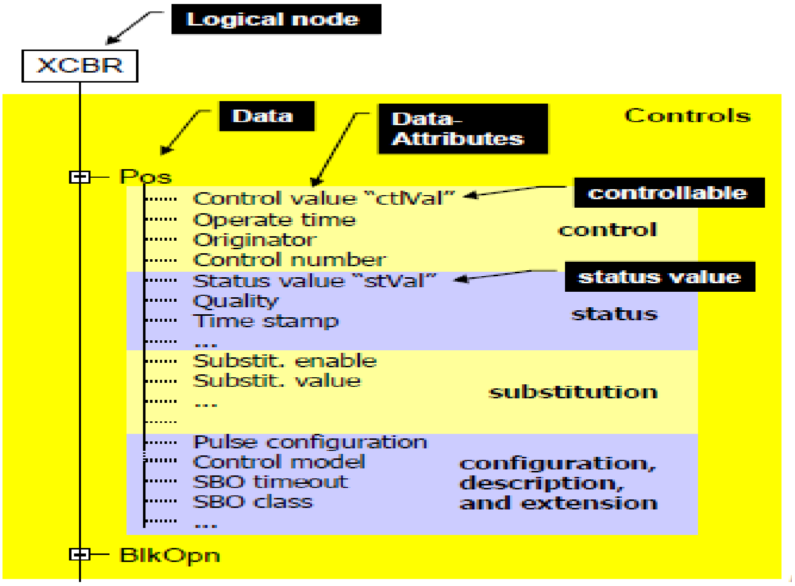
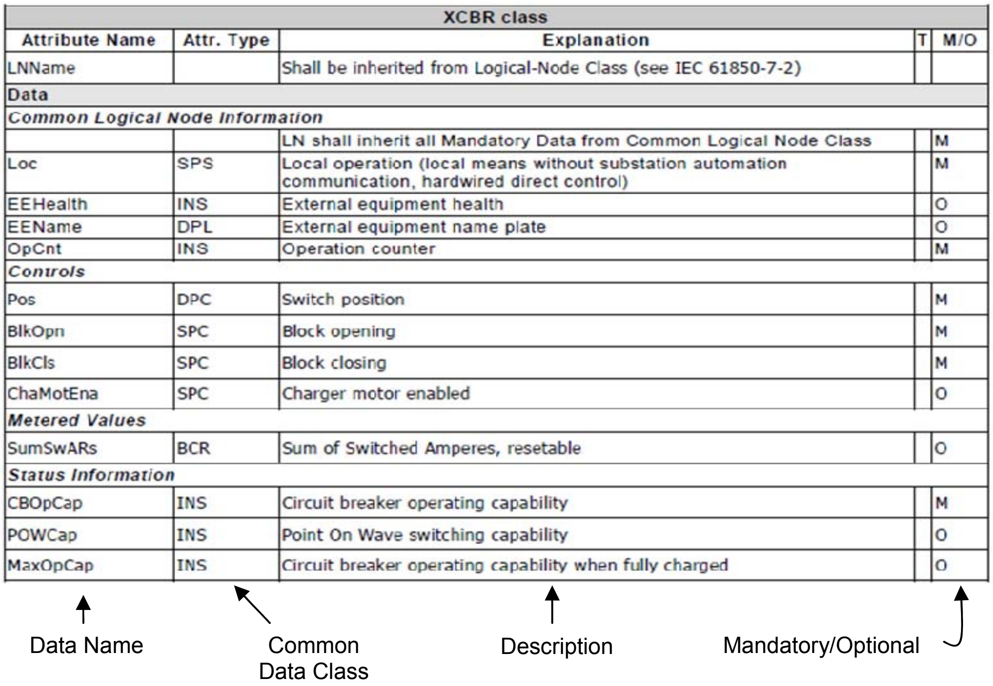


Figure 3.4: Position information depicted as a tree
(IEC 61850-7-1, 2003, p.19)

Figure 3.5 displays an example of a Logical Node Class definition (e.g. Circuit Breaker - XCBR). A Logical Node Class is a template for the development of a logical node. Parameters specified as Mandatory must be detailed, but those specified as

Optional may be inserted as per the designer's needs. A logical node class is a collection of data objects. Each data object belongs to a particular data class (e.g. controllable double point-DPC), and each data class is implemented using a collection of basic or composite data.



XCBR class				
Attribute Name	Attr. Type	Explanation	T	M/O
LNName		Shall be inherited from Logical-Node Class (see IEC 61850-7-2)		
Data				
Common Logical Node Information				
		LN shall inherit all Mandatory Data from Common Logical Node Class		M
Loc	SPS	Local operation (local means without substation automation communication, hardwired direct control)		M
EEHealth	INS	External equipment health		O
EEName	DPL	External equipment name plate		O
OpCnt	INS	Operation counter		M
Controls				
Pos	DPC	Switch position		M
BlkOpn	SPC	Block opening		M
BlkCls	SPC	Block closing		M
ChaMotEna	SPC	Charger motor enabled		O
Metered Values				
SumSwARs	BCR	Sum of Switched Amperes, resetable		O
Status Information				
CBOpCap	INS	Circuit breaker operating capability		M
POWCap	INS	Point On Wave switching capability		O
MaxOpCap	INS	Circuit breaker operating capability when fully charged		O

Figure 3.5: XCBR Logical Node Class definition
Adapted from (IEC 61850-7-4, 2003, p.59)

For example, the Data-Object "Pos" (Position) of the Logical Node Class XCBR (Circuit breaker) belongs to the "controllable double point-DPC" Data Class (Attribute type). The DPC Data Class is composed of a basic data type such as "Boolean" or a complex data type such as "Double Point Position- Dbpos" (Section 4.4.4.1 on page 106/ IEC 61850 modelling method).

The device data model is used to serve as a basis for the IEC 61850 system configuration philosophy - this is explained in Section 3.5.

The next section deals with the naming convention implemented in the IEC 61850 standard.

3.3.2 IEC 61850 Naming convention

When combined with the conceptual philosophy of “virtualisation” the rigorous specification of a naming convention in IEC 61850 assists in the goals of standardization and interoperability. The IEC 61850 naming convention defines a set of rules to properly address an object.

Figure 3.6 provides an example of the naming convention adopted in IEC 61850. The logical device name can be adopted independently of the IEC 61850 standard. The second part identifies the logical node in which the object (data object) is. The first letter in the “logical node” section signifies the group of the node. In this case “X” signifies switchgear and the remaining letters (CBR1) - circuit breaker one (SUITTIO, 2010). [Note: The “logical node” section also may have a prefix (Q0XCBR1) that can be used to add a description with respect to what the logical node is related to, and may have a suffix call instance number (XCBR1), to signal the difference between instances of the same logical node] (IEC 61850, 2010). Generally, Data Objects are grouped by means of Functional Constraint (FC) defined in IEC 61850-7-2. In the case of Figure 3.6, “ST” refers to the functional constraint - status information.

There needs to be a differentiation between the object name and the object reference that is the complete path to reach a specific data attribute. Additional information about the naming convention can be found in IEC 61850-1-2 (clause 19).

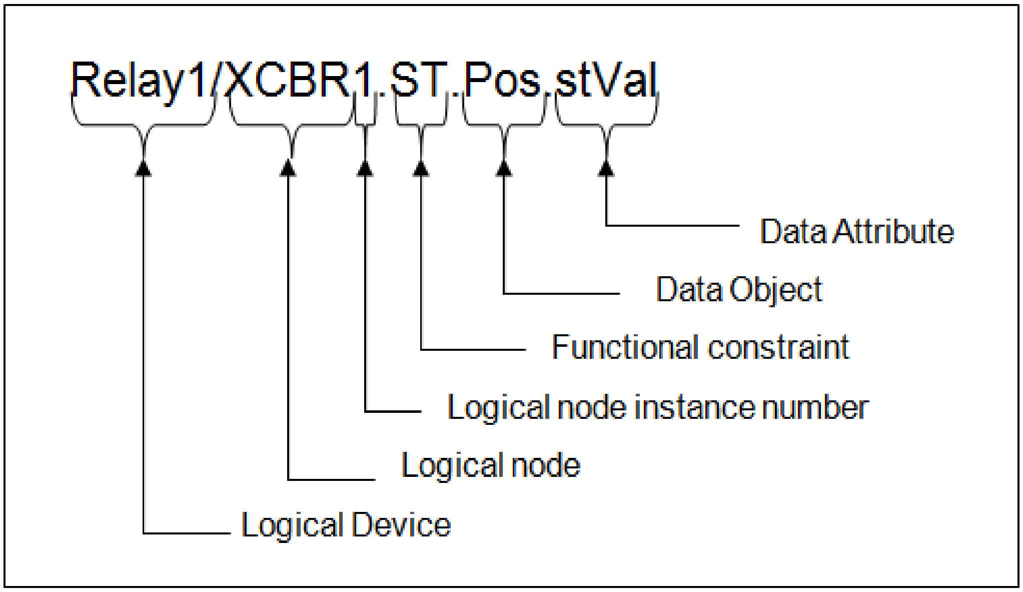


Figure 3.6: IEC 61850 Naming convention

If the MMS command (Read or Write) is used to access a data object, the separation delimiter “$” is used instead of the dot point in the naming convention (Adamiak et al., 2012).

3.3.3 Abstract Communication Service Interface

The Abstract Communication Service Interface (ACSI) is explained in greater detail in section 7-2 of the IEC 61850 standards (see Figure 3.1). The ACSI is one of the main factors to consider for interoperability.

The basic architecture of IEC 61850 ACSI mapping to the Open Systems Interconnection (OSI) model is shown in Figure 3.7 with the addition of an Abstract Layer of Generalised Communication and Specific Communication Services Mappings (SCSM). These two layers are added above the 7 layers of the OSI model (Makadam, 2008). The SCSM is discussed in Section 3.4.1.

The IEC 61850 architecture provides a neutral interface between the application object and its related application services. This architecture allows process and user applications to be designed independently from the communication theory (Ozansoy, 2006).

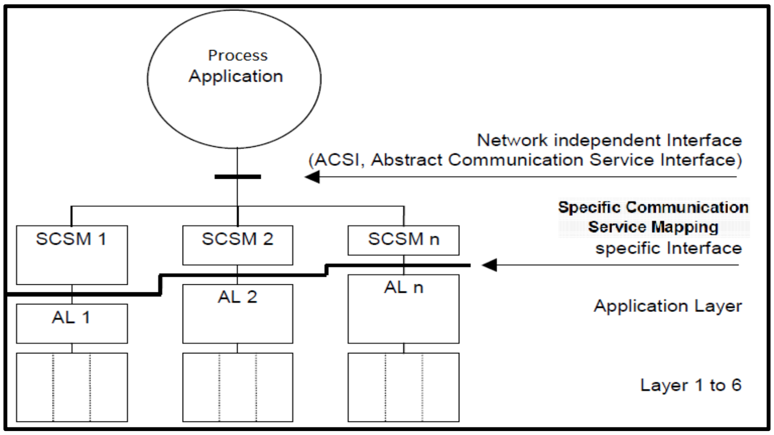


Figure 3.7: ACSI mappings to OSI communication model
(IEC 61850-7-1, 2003, p.66)

The ACSI provides a number of abstract interfaces. Some of them describe communications between a client and a remote server. Other interfaces are provided for communication between an application in one device and a remote application in another device. The communication between a client and a remote server may be for device control, logging of events, publisher/subscriber … (Hammer and Sivertsen, 2008).

With reference to Figure 3.1- the Abstract Communication Service Interface (ACSI) has been has been highlighted by a blue/purple block. Figure 3.8 below now presents a more detailed model of the ACSI. It is comprised of the information model (basic services model) and the information exchange model (specific services model) (Hammer and Sivertsen, 2008).

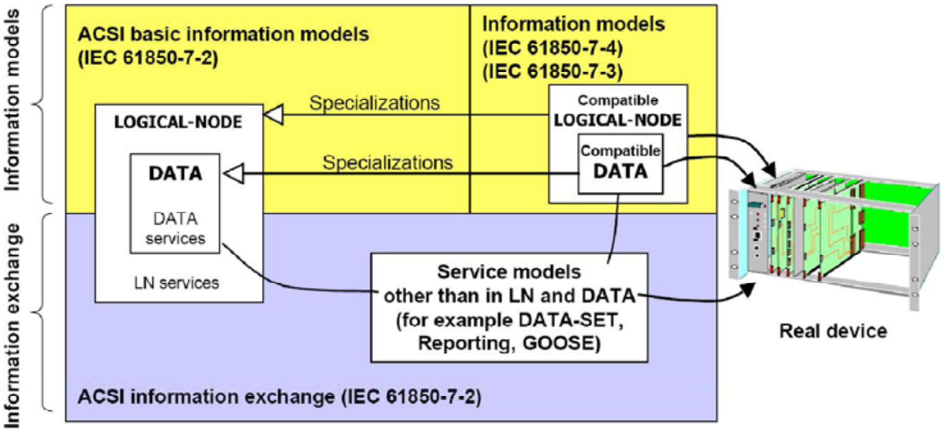


Figure 3.8: Conceptual model of ACSI
(IEC 61850-7-2, 2003, p.15)

Please refer to Figure 3.14 to confirm where the ACSI is situated relative to the 7 layers of the OSI model.

A description of the ACSI information and information exchange models is presented in the following section.

3.3.4 Information Models

The information model is the first sub-layer of the Abstract Communication Service Interface (ACSI) presented earlier in Figure 3.7 and Figure 3.8.

The information model represents the elements by which a physical-device is virtualised. It is the direct output result of the virtualisation process. The information model is composed of the following (Hammer and Sivertsen, 2008):

- Server: Represents the external visible behaviour of a device - all other ACSI models are part of the server.
- Application association: Allows a client to be associated (connected) with a server.
- Logical Device (LD): Contains the information produced and consumed by a group of domain-specific application functions. Functions are defined as logical nodes contained in an appropriate LD. This means that the logical device contains logical nodes with the same type of functionality, e.g. several

nodes with protection related functionality can be grouped into one logical device and nodes with control functionality can be grouped into another. A physical device in this way can consist of one or several logical devices (Hamrén, 2007).

- Logical Node (LN): contains the information produced and consumed by a domain-specific application function - for example, overvoltage protection or circuit-breaker.
- Data: provide a means to specify typed information - for example, position of a switch with quality information and timestamp, contained in LNs.

Figure 3.9 presents the basic structure of the information model.

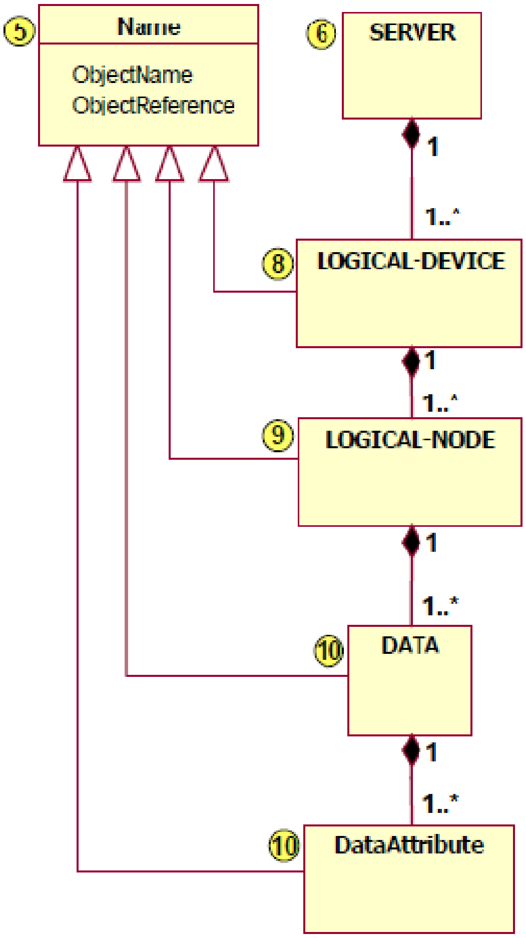


Figure 3.9: Basic information model of the ACSI
(IEC 61850-7-2, 2003, p.16)

A description of each element that comprises the information model can be found in part 7-2 of the IEC 61850 standard.

3.3.5 Information Exchange

The information exchange is the second sub-layer of the Abstract Communication Service Interface (ACSI) presented earlier in Figure 3.7 and Figure 3.8.

The IEC 61850 standard allows for two groups of communication services between entities within the substation automation system. One group utilizes a client-server model accommodating services such as Reporting and Remote Switching. The second group utilizes a peer-to-peer model based on a Publisher/Subscriber mechanism for Generic Substation Event (GSE) services associated with time-critical activities such as fast and reliable communication between Intelligent Electronic Devices (IEDs) used for protection purposes. Figure 3.10 illustrates interrelationship between Client/Server and Publisher/Subscriber services.

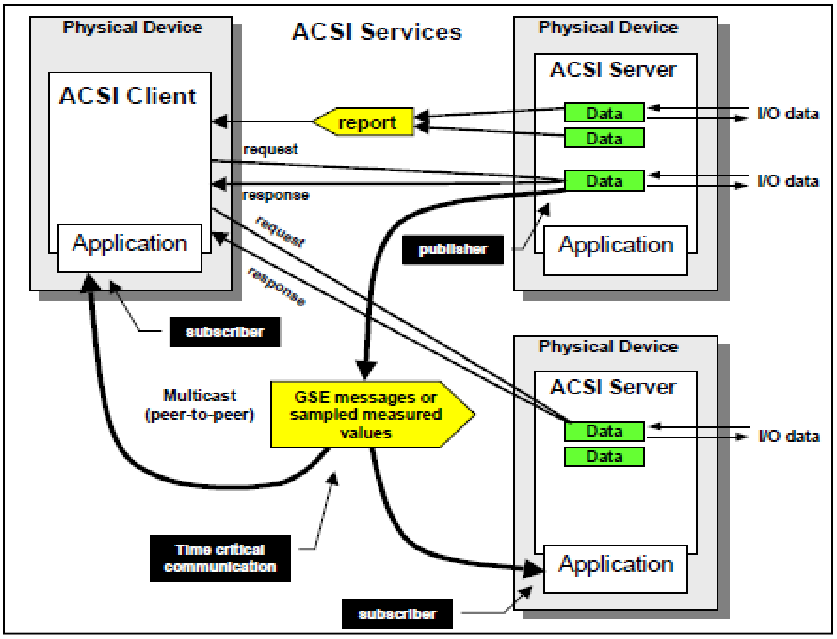


Figure 3.10: ASCI communication methods
(IEC 61850-7-1, p50)

The information exchange represents the elements by which a virtual device is configured in order to communicate with the outside world. It is composed of the following (Hammer and Sivertsen, 2008):

- DATA-SET - permits the grouping of data and data attributes;
- Substitution - supports replacement of a process value by another value;
- SETTING-GROUP-CONTROL-BLOCK - defines how to switch from one set of setting values to another one and how to edit setting groups;

- REPORT-CONTROL-BLOCK and LOG-CONTROL-BLOCK - describes the conditions for generating reports and logs based on parameters set by the client;
- Control blocks for generic substation events (GSE) - supports a fast and reliable system-wide distribution of input and output data values;
- Control blocks for transmission of sampled values - fast and periodic transfer of samples - for example, of instrument transformers;
- Control - describes the services to control - for example, devices;
- Time and time synchronization - provides the time base for the device and system;
- File transfer - defines the exchange of large data blocks such as programs.

Please refer to Figure 3.11 below to identify the interrelationship between some of the elements mentioned above - as depicted through the use of a UML diagram.

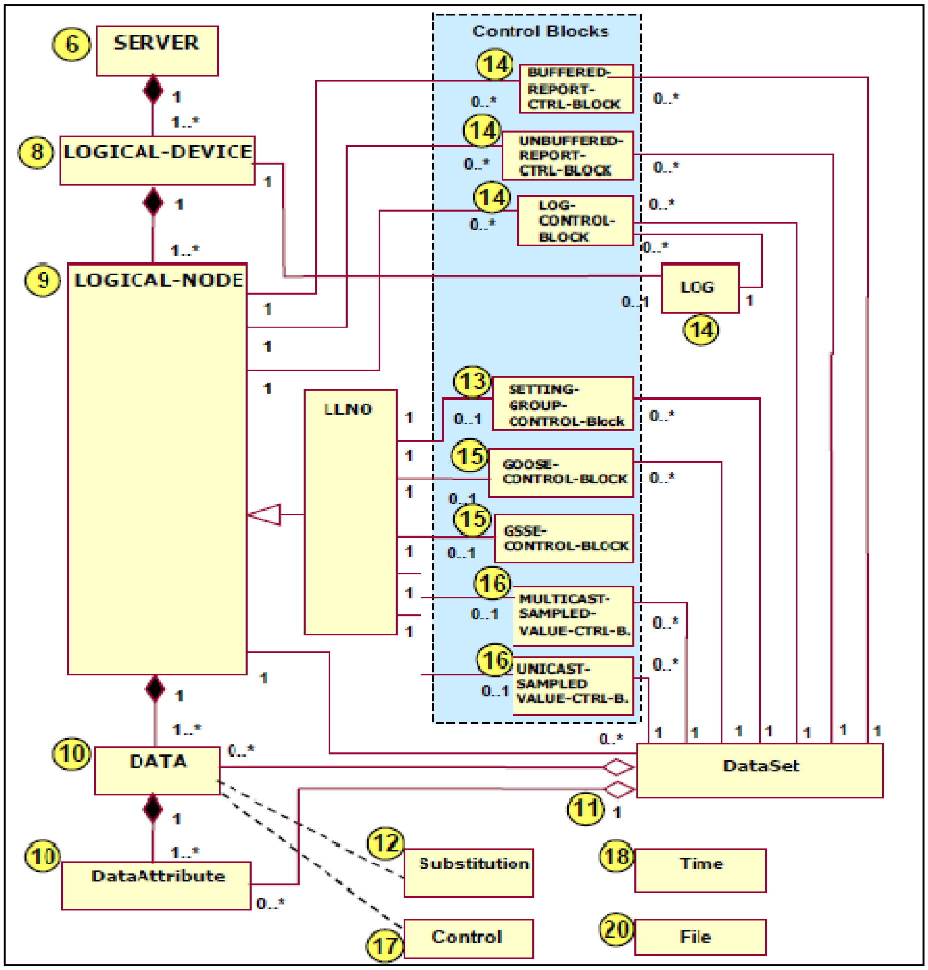


Figure 3.11: Conceptual service model for ACSI

(IEC 61850-7-2, 2003, p.17)

A good description of each element that comprises the ACSI can be referenced in part 7-2 of the IEC 61850 standard or in (Hammer and Sivertsen, 2008).

The next section briefly describes the client/server communication.

3.3.6 Client-Server Architectures

The client/server architecture represented in Figure 3.12 exhibits one possible relationship between two software applications in which the client requests a service from the server. The client/server relationship can apply to two software applications running on a single computer or two programs running over a communication network (Ozansoy, 2006).

The client/server is generally known as a synchronous type of messaging since the client is blocked once it makes the request until the corresponding reply arrives. But the possibility offered by object-oriented technology enables the development of asynchronous client/server architecture (Ozansoy, 2006). This means that a client can issue a request and process some other task in parallel while waiting for the server to reply. This type of architecture allows for distributed applications over the network with a greater reliability and service delivery. The asynchronous client/server architecture is made possible by the implementation of an Application Programming Interface (API) and a mechanism for message queuing with different levels of priority (Ozansoy, 2006).

The implementation of a client/server architecture is required in IEC 61850 to enable the Abstract Communication Service Interface (ACSI) - which is one of the essential mechanisms to ensure interoperability among devices from different manufacturers. A detailed structure analysis of the type of server involved in this project is tabled in Chapter 4 of this work.

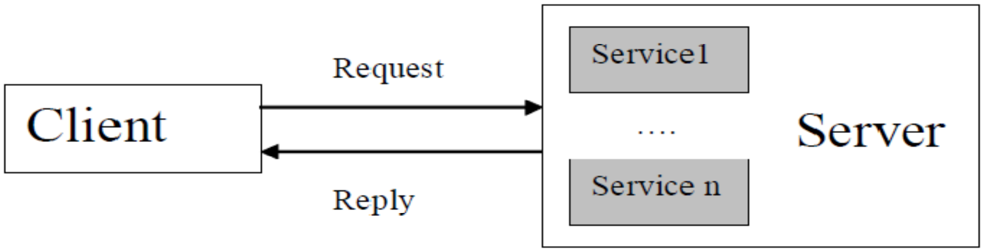


Figure 3.12: Client/Server communication model
(Ozansoy, 2006)

The following section describes the publisher/subscriber communication architecture.

3.3.7 Publisher-Subscriber Architecture

The publisher/subscriber architecture presented in Figure 3.13 shows the relationship that exists between a software application known as the publisher and another software application known as the subscriber.

The publisher/subscriber model is an asynchronous type of many-to-many communication (Olovsson and Lejdeby, 2008).

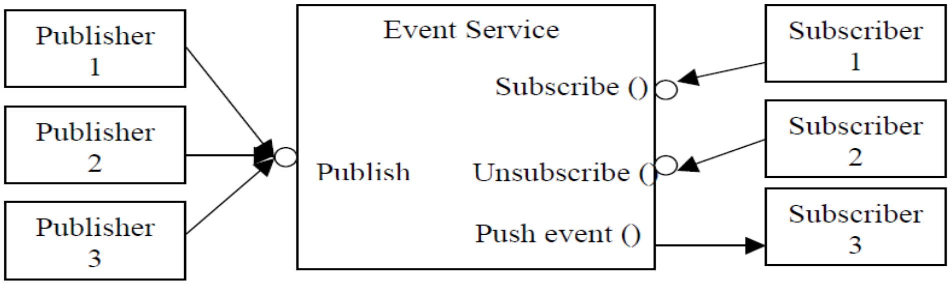


Figure 3.13: Publish/Subscribe communication model
(Ozansoy, 2006)

The publisher/subscriber architecture is an addition to the client/server architecture previously defined. In the sense of IEC 61850, a publisher/subscriber mechanism is only possible via the logical node zero (LLN0[16]) of a Server. This implies that a publish/subscribe communication can occur only if client/server architecture has been constructed previously. This makes available ACSI services that are important for IEC 61850 system interoperability.

Figure 3.11 illustrates the fact that an IEC 61850 publisher/subscriber model can be defined between peer devices only if a server exists. If the server is destroyed, all elements dependant on that server cease to exist.

For example, a component of the GOOSE control block (GoCBRef, GoEna…) can be accessed and manipulated only in the client/server communication architecture (Figure 3.11) using the IP address of the considered device.

The next section deals with the data communication philosophy for an understanding of how information is exchanged among all devices.

[16] Chapter 5 will provide more information into the role played by Logical node zero within the IEC 61850 philosophy.

3.4 Data communication philosophy

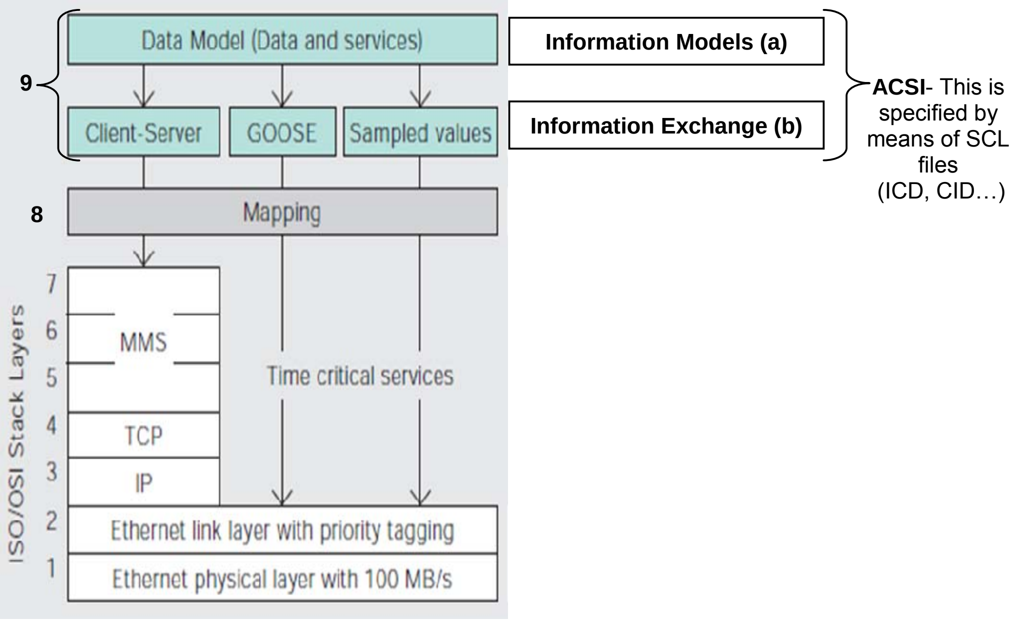
3.4.1 Specific Communication Service Mapping (SCSM)

There are two groups of information exchange systems that have been defined in the IEC 61850 standard. The first one is based on the Client/Server architectural structure only and the second one has added features based on the Publisher/ Subscriber architectural structure.

To accommodate these services, the IEC 61850 standard deploys in its specification a layered communication structure on top of the traditional OSI model. This structure is composed of two layers.

From an embedded point of view, if an application wants to send information to the outside world, this information needs to pass through two IEC 61850 layers (ACSI and SCSM) before entering the layers contained within the OSI model.

Figure 3.14 illustrates the IEC 61850 layers (layer 9) in the Abstract Communication Service Interface (ACSI). This first layer is composed of 2 sub-layers namely the information model (a- Data Model and basic service models) and the information exchange model (b- specific service models). The second layer (layer8) corresponds to the Specific Communication Services Mapping (SCSM). Reference can also be made to Figure 3.1. Figure 3.7 and Figure 3.8.



Data Model (Data and services)
Information Models (a)
9
Client-Server
GOOSE
Sampled values
Information Exchange (b)
ACSI- This is specified by means of SCL files (ICD, CID…)
8
Mapping
ISO/OSI Stack Layers
7
6
MMS
5
Time critical services
4
TCP
3
IP
2
Ethernet link layer with priority tagging
1
Ethernet physical layer with 100 MB/s

Figure 3.14: Layered structure of the IEC 61850 standard

Adapted from (ABB review, 2010)

[Note: It is also imperative to note that the protocol stack is an implementation of a computer networking protocol suite (SV, GOOSE, MMS…). The suite is the definition of the protocols, and the stack is the software implementation of them (Adaptive Digital, 2012)].

The Specific Communication Services Mapping (SCSM) is a mapping of the information model and Abstract communication Services to a known protocol - such as the Message Manufacturing Specification (MMS) protocol, deployed at layers 7 down-to 5 of the OSI 7-layer stack Figure 3.15.

The SCSM also provides access to lower layers of the OSI model, as can be seen in Figure 3.15 where GOOSE and Sampled Value (SV) messages are mapped directly to the Data Link Layer after passing through the ACSI and SCSM layers. This indicates that the ASN.1 decoding and encoding of GOOSE/SV messages is the responsibility of the IEC 61850 communication stack (ASCI and SCSM layer) since both of these are not mapped via the Presentation Layer of the OSI model.
Figure 3.15 gives an overview on the paths followed through the OSI stack by various IEC 61850 message types.

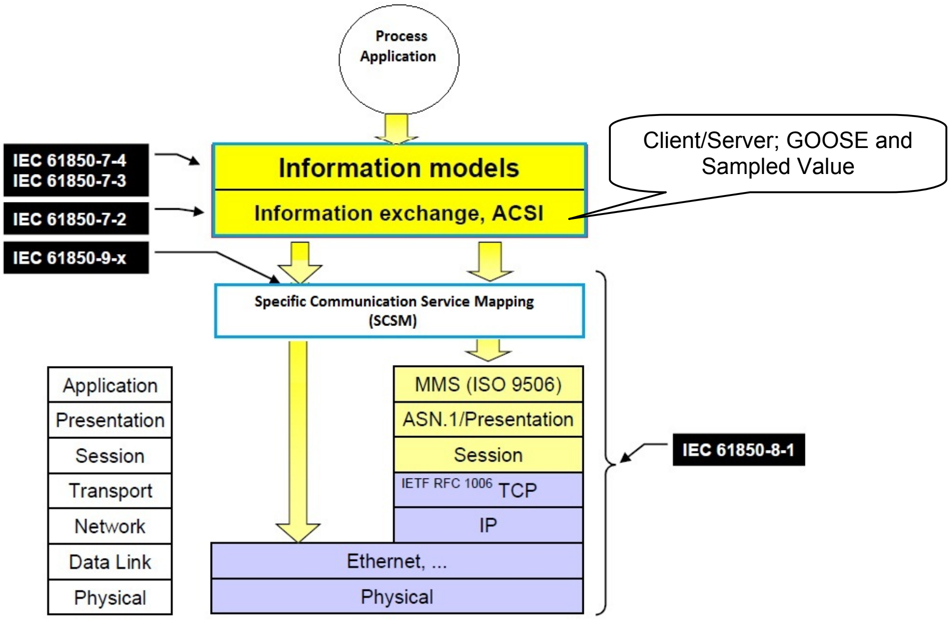


Figure 3.15: Mapping of IEC 61850 specifics layers to OSI model

Adapted from (IEC 61850-7-1, 2003, p.22)

3.4.2 Definition of a GOOSE message

The Generic Object Oriented Substation Event (GOOSE) is provided to report any change of state of an IED to other peer devices (IEC61850-2_p13).

Figure 3.16 below, shows the mapping profile of the IEC 61850 protocol suite. It is acknowledged that communication protocols such as GOOSE and Sampled Value messages are mapped over two layers of the OSI model which are: the Physical layer (layer 1) and the Data-link Layer (layer2).

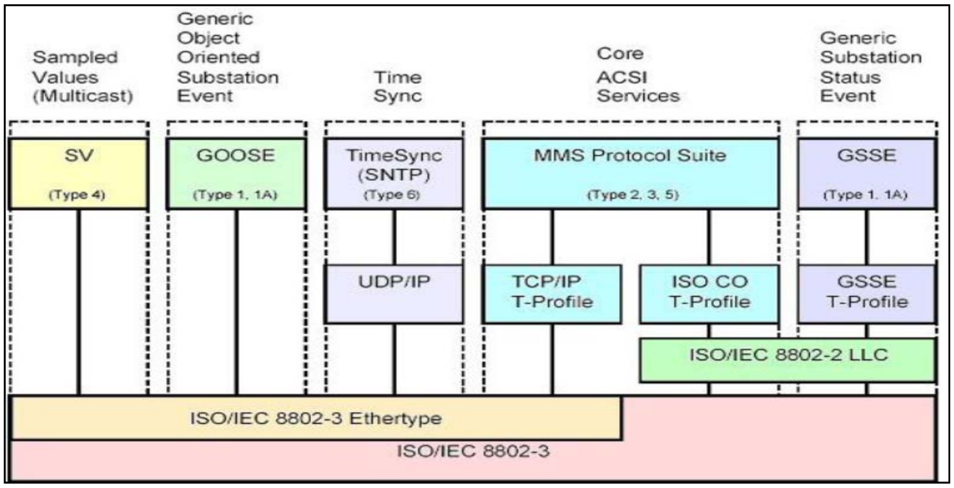


Figure 3.16: Protocol mappings profile
(Taken from Seclab, (Zhang and Gunter, 2009)**)**

A GOOSE message allows for high speed trip signals to be issued with a maximum probability of delivery within a specific time range. Table 3.4 gives information about the time requirements for different applications used within the substation.

Table 3.4: IEC 61850 message types and performances

Type	Applications	Performance Class	Requirements (Transmission Time)
1A	Fast Messages (Trip)	P1	10 ms
		P2/P3	<3 ms
1B	Fast Messages (Other)	P1	100 ms
		P2/P3	20 ms
2	Medium Speed		100 ms
3	Low Speed		500 ms
4	Raw Data	P1	10 ms
		P2/P3	3 ms
5	File Transfer		≥1000 ms
6	Time Synchronization		(Accuracy)

(Hou and Dolezilek, 2012)

Services associated with the GOOSE communication model (IEC 61850/ Edition 1) are as expressed in Table 3.5. Note that services such as Get GOOSE Control Block Values (GetGoCBValues) are mapped to the MMS Client/Server profile.

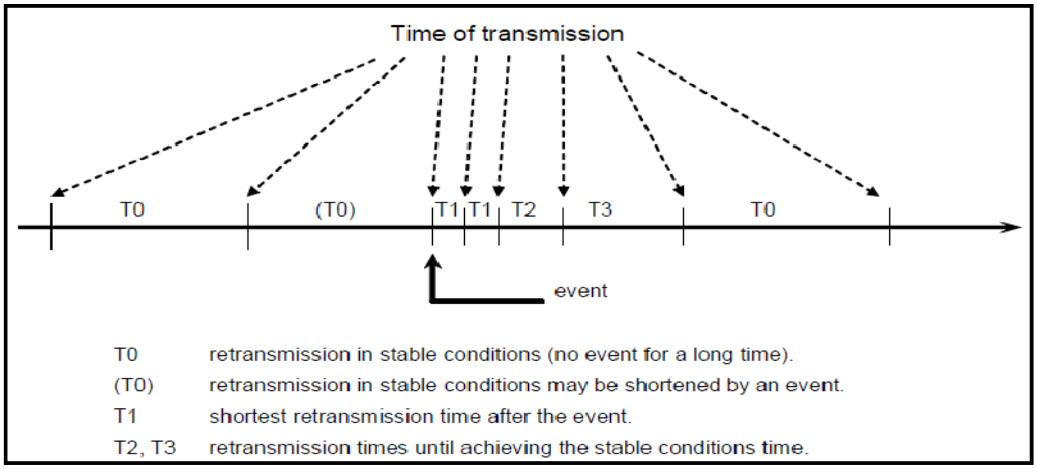
Table 3.5: Services associated with the GOOSE communication model

IEC 61850-7-2 Model (p19/ Edition 1)	Associated Services
Generic Substation Event GOOSE	**IEC 61850-7-2 Services requiring Client/Server Communication Profile (mapped to MMS/ IEC 61850-8-1, p23/ Edition 1)**
	GetGoCBValues SetGoCBValues
	Services requiring GSE Management and GOOSE communication profile (IEC 61850-8-1, p23/ Edition 1)
	GetReference GetGOOSEElementNumber SendGOOSEMessage

The next section focuses on the GOOSE retransmission strategy.

3.4.3 Retransmission strategy for GOOSE message

The GOOSE message is published continuously and repetitively – before, during, and after an Event. After the Event is detected, the repetition rate is initially increased, then slowed down – back to what it was before the Event occurred (see Figure 3.17). According to the (IEC 61850-8-1, 2004, p.68), the first repetition delay value is an open value (defined by the manufacturer). The repetition algorithm can be geometric or linear.



Time of transmission
T0
(T0)
T1
T1
T2
T3
T0
event
T0 retransmission in stable conditions (no event for a long time).
(T0) retransmission in stable conditions may be shortened by an event.
T1 shortest retransmission time after the event.
T2, T3 retransmission times until achieving the stable conditions time.

Figure 3.17: Transmission time for GOOSE
(IEC 61850-8-1, 2004, p.68)

3.4.4 Logical Interfaces and Substation Network Topology

Figure 3.18 presents the architecture and system interface between equipment/ or logical functions in a modern substation automation system with respect to the IEC 61850 standard (IEC 61850-5, 2003, p.15). Each number within a circle represents a logical interface (not necessarily a physical interface).

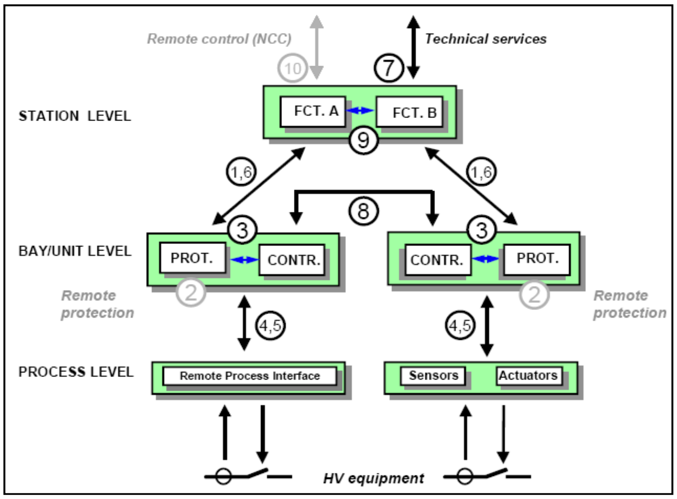


Figure 3.18: Interface model of a substation automation system
(IEC 61850-5, 2003, p.15)

The numbered interfaces are presented as follows:

1: protection - data exchange between bay and station level.

2: protection - data exchange between bay level and remote protection (greyed out: beyond the scope of Edn.1 of the standard).

3: data exchange within bay level.

4: CT and VT instantaneous data exchange (especially samples) between process and bay level.

5: control - data exchange between process and bay level.

6: control - data exchange between bay and station level.

7: data exchange between substation (level) and a remote engineer's workplace.

8: direct data exchange between the bays especially for fast functions such as interlocking (GOOSE…).

9: data exchange within station level.

10: control - data exchange between substation (devices) and a remote control centre (greyed out: beyond the scope of Edn.1 of the IEC 61850 standard).

[Note: activities and interfaces represented by grey dots and lines are not supported or specified within IEC 61850 Edition one].

The substation communication network can be structured either as a bus, ring or star topology. Depending on the configuration of the substation the network topology can be a combination of the three models. In most cases a common configuration is a ring topology with a redundant link at the switch level.

The IEC 61850 standard does not impose the use of a particular technology for signal interconnection (fibre optic, copper wire or wireless); however fibre optic is generally preferred as it presents many advantages in terms of communication speed, electromagnetic isolation of the signal and cyber security.

Station Bus refers to an application domain where relays and RTUs would attach to the LAN. Figure 3.19 illustrates how IEC 61850 compliant IED's are currently typically connected at the station bus level. Sensors such as VTs and CTs, and actuators such as switches and circuit breakers are directly connected to the IEDs using dedicated hardwiring.

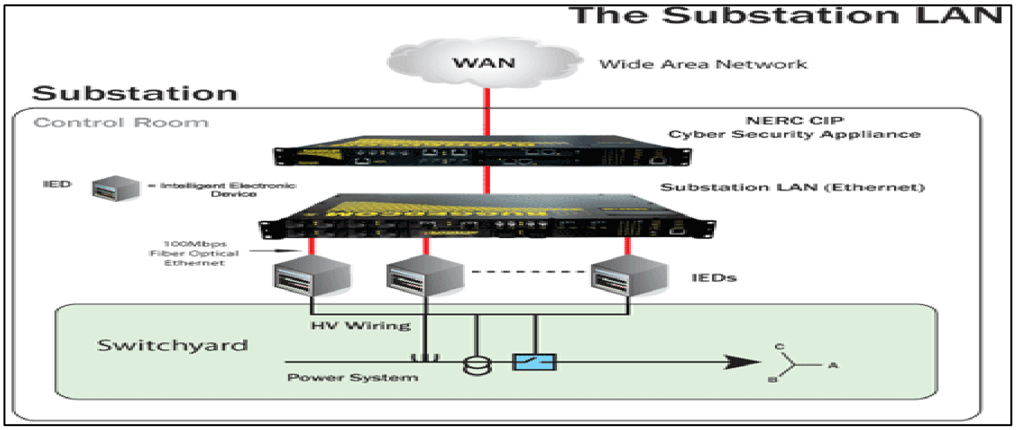


Figure 3.19: Current use of standardised IEDs
(Ruggedcom Russia, 2010)

Process Bus refers to devices such as CT/VT Merging Units (MU) providing sampled measured values of current and voltage via the LAN. Figure 3.20 represents how the IEC 61850 standard would be implemented with merging units at the process bus level. This type of architecture uses less hardwired cables.

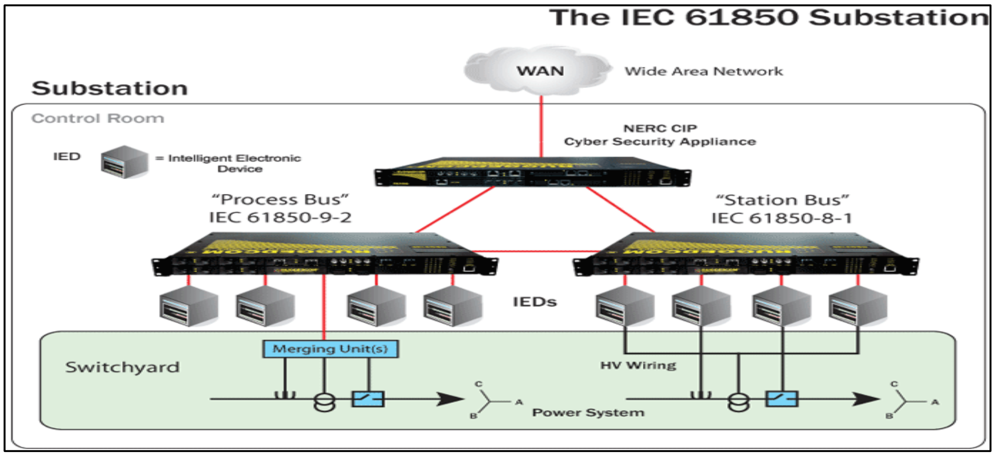


Figure 3.20: SAS architecture compliant IEC 61850
(Ruggedcom Russia, 2010)

[Note: The merger between compliant IEC 61850 IEDs and legacy devices can be achieved through proctocol converter. This topic is not within the scope of this project].

The IEC61850 standard does not define implementation of the concepts and protocols inside any device (concept of the Black Box) (Apostolov, 2010). What is also not defined is the mapping of internal control signals to a device hardware interface, such as a binary output to trip a circuit breaker. The IEC 61850 is a communication standard that models the structure of data to be communicated on the network and the communication model to be considered depending on the application, but it is not a standard for software modelling or software code development. Software modelling or development problems are left as local issues for IEC 61850 equipment manufacturers. Developers of custom IEC 61850 applications therefore require intimate knowledge in the fields of Protection, Networking, Software Engineering, Control Systems and Embedded Systems. The development of an IEC 61850 compliant device or application is implemented according to three main views which are (IEC 61850, 2010):

- Application functional view[17] ;
- Application data modelling or device view (IEC 61850 series);
- Communication view (IEC 61850 series).

Chapter 4 provides more details regarding each of these views. Table 3.6 provides a list of important standards directly related to the current research project.

Table 3.6: Project related standards

Main Standard	Sub standard	Comments
1- IEC 61850	-	Standard for communication in Substation Automation System
2- OSI Model (ISO/IEC 7498-1)	2.1 ASN.1- BER	For Data encoding and decoding
	2.2 IEEE 802.1Q	For Priority tagging/ VLAN definition
	2.3 8802-3	For Ethernet frame definition
3- PSS-05-0 (Software Engineering standards)	3.1 ANSI/IEEE 1058.1	Software Management Project
	3.2 ANSI/IEEE 610.12-1990	Software configuration Management
4- UML 2.2	-	Unified Modelling Language. For definition of project requirement and detailed software modelling
5- IEC 62271	Part 3	Standard for High-voltage switchgear and controlgear- Digital interfaces based on IEC 61850

[17] Include all aspects related to the software application program or application subroutine and substation functions or protection scheme. This is beyond the scope of the IEC 61850 series.

The next section provides appropriate details on the system configuration approach and the structure supporting it.

3.5 System configuration philosophy

The substation configuration philosophy is fundamental for the SAS functional specification, IED capability description and SA system description (IEC 61850-6, 2004). This philosophy is supported by a structured Substation Configuration Language (SCL) for the purpose of system design, communication engineering and information exchange between engineering tools for the various devices in a standardized way (IEC 61850-6, 2004). The SCL files are specified by the IEC 61850 standard to be written in the Extensible Mark-up Language (XML), and to have a specific format in terms of the information it contains.

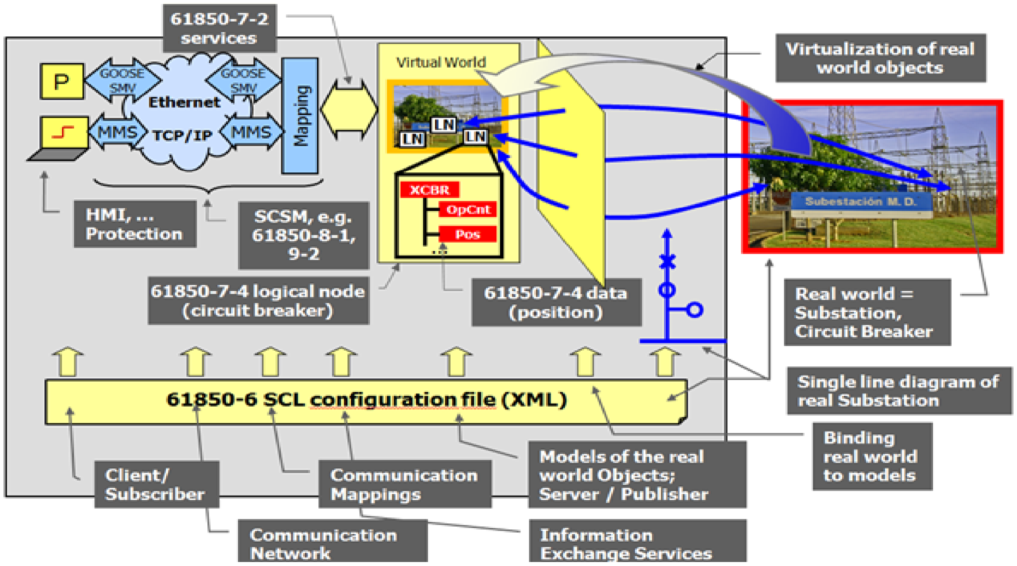

61850-7-2 services
P
GOOSE SMV
Ethernet
GOOSE SMV
MMS
TCP/IP
MMS
Mapping
Virtual World
LN
LN
LN
XCBR
OpCnt
Pos
...
Virtualization of real world objects
Subestación M. D.
HMI, ... Protection
SCSM, e.g. 61850-8-1, 9-2
61850-7-4 logical node (circuit breaker)
61850-7-4 data (position)
Real world = Substation, Circuit Breaker
61850-6 SCL configuration file (XML)
Single line diagram of real Substation
Binding real world to models
Client/ Subscriber
Communication Mappings
Models of the real world Objects; Server / Publisher
Communication Network
Information Exchange Services

Figure 3.21: Importance of Substation configuration Language
(Schwarz, 2012)

Figure 3.21 is a more detailed illustration with respect to the role of the XML-based SCL files than that shown in Figure 3.1.

The configuration process requires the system engineer to have adequate and reliable software engineering tools to define (Create/Import/Export) an object model describing the IEDs, the communication connections, and location in the switchyard. The considered tools also have the responsibility to represent this model in a standardized way to an output file to be exchanged between engineering tools from different vendors. Figure 3.22 depicts the reference model for information flow in the configuration process.

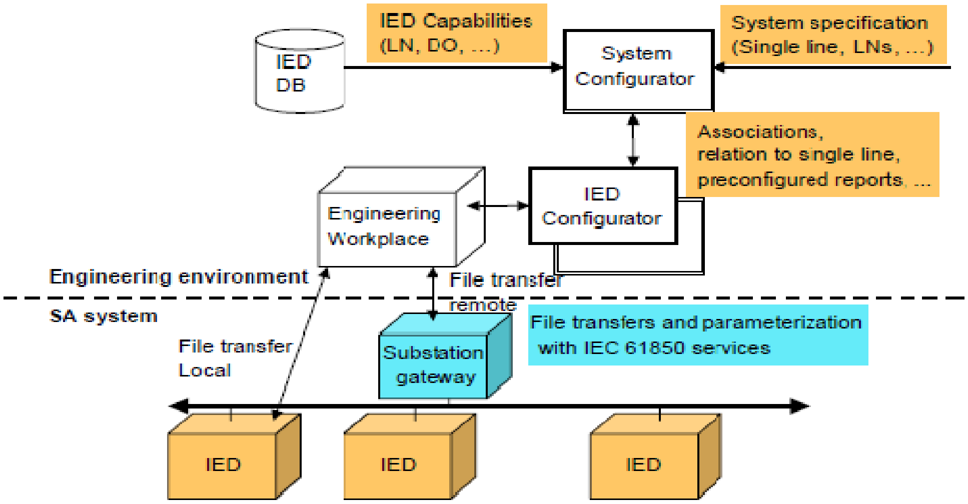

IED Capabilities (LN, DO, ...)
System specification (Single line, LNs, ...)
IED DB
System Configurator
Associations, relation to single line, preconfigured reports, ...
Engineering Workplace
IED Configurator
Engineering environment
File transfer remote
SA system
File transfer Local
Substation gateway
File transfers and parameterization with IEC 61850 services
IED
IED
IED

Figure 3.22: Reference model for information flow in the configuration process
(IEC 61850-6, 2004, p.11)

Additional information about the structure that allows for configuration of the SAS is detailed later in this chapter.

The next section describes the Substation Configuration Language and its purpose in support of the IEC 61850 system configuration philosophy.

3.5.1 Substation Configuration Language (SCL)

The IEC 61850 defines an information exchange mechanism via SCL (Substation Configuration Language) files that allow for configuration and reconfiguration of a substation. The SCL file format is of an XML (Extensible Mark-up Language) type. It describes communication-related IED configurations, IED parameters, communication system configuration, Abstract Communication Service Interfaces (ACSI), function structures and the relation between them (Hammer and Sivertsen, 2008). Table 3.7 presents all SCL file types used in the substation and their roles (IEC 61850, 2010).

Table 3.7: SCL file extension description

Extension	Name	Description
xxx.**icd**	IED Capability Description	Defines complete capability of an IED. Vendors are required to deliver a commercially available IED with an ICD file.
xxx.**cid**	Configured IED Description	Defines protocols, parameter values and data structure used for configuration of the IED at start up. Using this file is optional.
xxx.**ssd**	System Specification Description	Complete specification of the SAS with single line diagrams, required functions and does not include the IED description.
xxx.**scd**	Substation Configuration Description	Complete specification of the SAS including the IED description.

Figure 3.23 represents a UML diagram overview of the SCL schema. In other words it defines what needs to be considered in the composition of an SCL file with respect to

its purpose. Please refer to APPENDIX A for additional information on the UML diagram specification.

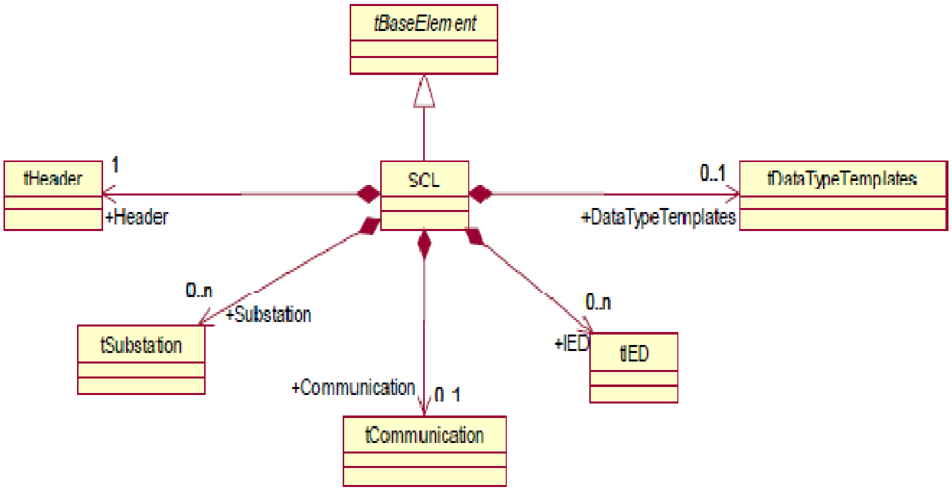


Figure 3.23: UML diagram overview of SCL schema
(IEC 61850-6, 2004, p.21)

In general, an SCL file is composed of the following sections:

- **Header**: contains the “id” and the “nameStructure” of an SCL file.
- **Communication system description**: contain all addresses and security parameters.
- **Substation description**: Layout of the topology of the substation in terms of substation name, voltage level and bay unit.
- **IED description**: gives the full description of an IED (Services supported …)
- **Data type templates**: Instantiate type used for Logical Node, Data Object, Data Attribute and Enumerated Data.

3.5.2 Role played by software tools in the substation configuration process

Figure 3.24 presents an example of a substation engineering process using SCL files and dedicated configuration tools.

The IEC 61850 standard (Edition 1) does not constrain the use of the CID file as the device configuration file to be loaded at device start up. Some manufacturers use the ICD or SCD file format as the final configuration file for their device. Nevertheless the IEC 61850 standard considers the use the CID file type as being the logical choice for device configuration.

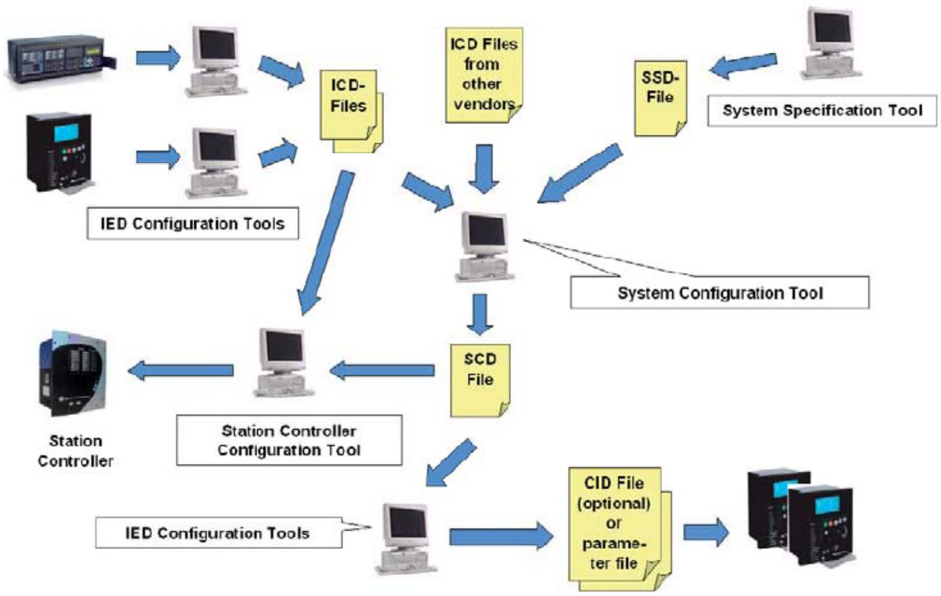

ICD-Files
ICD Files from other vendors
SSD-File
System Specification Tool
IED Configuration Tools
System Configuration Tool
SCD File
Station Controller
Station Controller Configuration Tool
IED Configuration Tools
CID File (optional) or parameter file

Figure 3.24: Conceptual substation engineering process using SCL
(Goraj and Herrmann, 2007)

The following section covers other standards related to the IEC 61850 standard for communication, data encoding, software engineering and data modelling.

3.6 Other related Standards

3.6.1 OSI Model

According to OSI model official website: "The OSI model are seven generic vertically stacked theoretic layers each providing data transmission functionality to the layer above or below. Each layer contains a set of systems, standards or protocols that communicate with corresponding entities in higher layers" (OSI Model-org, 2012). The OSI model is also known as ISO/IEC 7498-1. The following table provides a functional description of each layer.

Table 3.8: OSI Model and IEC61850 profile

Data Unit	Layer Name	Layer Function	Technology involve
Data	7. Application	Network process to application	FTP, HTTP, SMTP, SNMP, NTP, DHCP, Telnet, MMS, …
	6. Presentation	Data representation, encryption and decryption	XDR, ASN.1 …
	5. Session	Inter-host communication	RPC, NetBIOS, ASP…
Segment	4. Transport	End-to-end connections and reliability, Flow control	TCP, UDP, SCTP…
Packet	3. Network	Path determination and logical addressing	IP, ICMP, ARP, RARP…
Frame	2. Data Link	Physical addressing	Ethernet, Token ring, ISDN, IEEE 802.11, PPP…
Bit	1. Physical	Media, signal and binary transmission	100BASE-T, 1000BASE-T, T-carrier/E-carrier, 802.11 physical layers …

The IEC 61850 recommends the implementation of the OSI model with respect to what is called a profile [IEC 61850-8-1, p19]. There are two groups of profiles which are the Application-Profiles (A-Profiles) and Transport Profiles (T-Profiles). A-Profiles represent the protocols and agreements in regards to the upper 3 layers (Application, Presentation and Session) of the OSI Reference Model (ISO/IEC 7498-1). The T-Profiles represent the protocols and agreements in regards to the lower 4 layers of the OSI reference model. Additional information can be found in the IEC61850 standard section-8-1 on page 19.

3.6.2 ASN 1 standard

The Abstract Syntax Notation ONE (ASN. 1) standard for “Data networks and open system communications” is an international standard used to define protocols of communication by means of encoding rules. Information on ASN.1 with respect to the IEC 61850 standard can be found at IEC 61850-8-1, page111 (see APPENDIX B). The GOOSE protocol is defined using the ASN.1 /Basic Encoding Rule (BER) known as X.690-0207 (ISO/IEC 8825). Section 3.4.2 provides more detail regarding the structure of the GOOSE message and the ASN.1 encoding rule method.

3.6.3 Software Engineering Standards

Since the IEC 61850 standard does not specify which software development tools to use when developing an IEC 61850 application or even how software modules for such applications should be written, reference to the Software Engineering Standards is required. This is particularly important to choose a method for the development of any software application. Thus, it is important to understand that the IEC61850 standard is not a method for software application development.

The Software Engineering Standard refers to a list of different standards, procedures and rules that a programmer needs to know and follow. These are many standards for software engineering development, and the choice of a particular standard requires careful consideration. There is also a need to identify what software modelling and development tools are required.

The software development process also known as the software development life cycle is not to be confused with the IEC 61850 Data modelling. For the current research the software design and software algorithm is considered as the first and second step respectively within the software development life cycle. When considering the customized hardware that is implemented, the software development life cycle is in reality a subset for the complete development of an embedded intelligent circuit breaker node system.

The PSS-05-0[18]standard is developed and maintained by the European Space Agency (ESA) via the ESA's Board for Software Standardisation and Control (BSSC) since 1984. The PSS-05-0 is concerned with all software aspects of a system, including its interfaces to other software applications or hardware (ESA Board, 1991).The standard used for data and software modelling is the Unified Modelling Language (UML 2.0) and it is also used in the IEC 61850 standard. It should be noted that the object-oriented language is not the only software platform required to develop a system that complies with the IEC 61850 standard.

At first, the decision to use an Object-Oriented language (like C++) should be motivated by the number of subroutines that are required to run at the same time (need for concurrency) and not because the IEC 61850 specifies its data model using an object-oriented approach. Secondly, most Real Time Operating Systems (RTOS) have the ability, when the program is broken into small executable tasks (xx.exe, yyy.exe, kkk.exe…), to run each software module concurrently. The language used to develop the software application in this research project is based on the "C" language. Additional information and justification is presented in Chapters 4 and 5 regarding to the software development tools used in this research work.

3.6.4 Unified Modelling Language (UML)

The Unified Modelling Language (UML) is a standard mainly used for software modelling. It was developed by the Object Management Group (OMG) since 1997 to suit the modelling of object-oriented systems in the field of informatics, but it is used for other domain as well such as data structure modelling (SourceMaking, 2012). The current version of UML is UML 2.2.

The UML 2.2 specifications are composed of fourteen type diagrams depicted in Figure 3.25:

[18] The BSSC have replaced the PSS-05-0 in 1999 with the European Cooperation for Space Standardisation (ECSS-E-40 /Software Engineering standard). The ECSS-E-40 places a particular emphasis on earth-space applications.

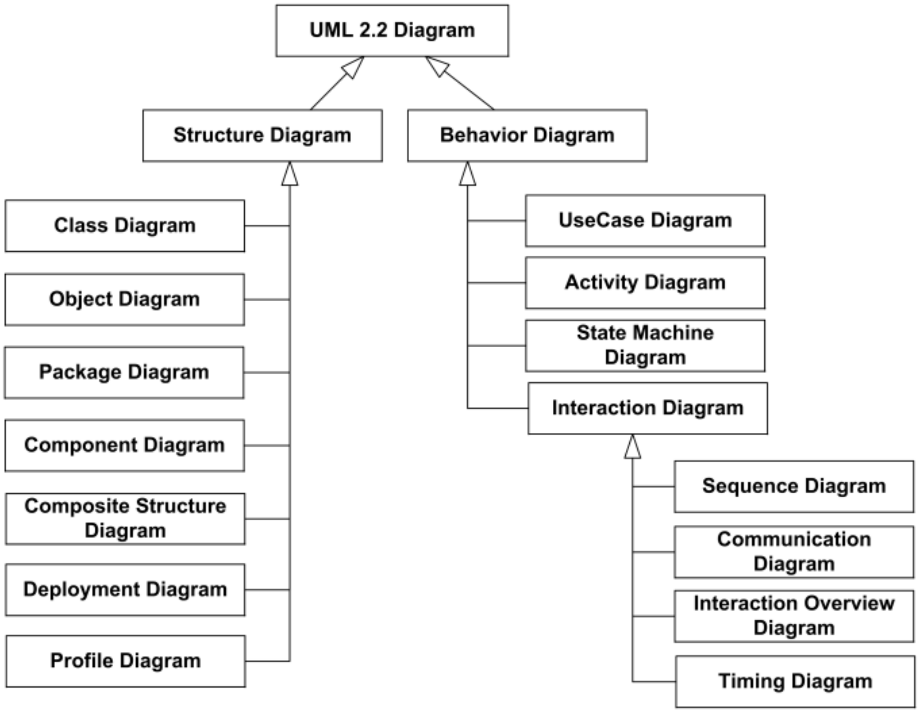


Figure 3.25: UML 2.2 diagram organisation
(OMG UML, 2012)

The designer needs to identify which selection of UML diagrams to use in order to clearly define the system in terms of technical specifications, architecture and behaviour conception. The software modelling portion of this research project using UML diagrams is presented in chapters 4 and 5.

3.7 Conclusion

The IEC 61850 standard covers not only communication, but also qualitative properties of engineering tools, measures for quality management and configuration management (Ozansoy, 2006).

More importantly, the IEC 61850 standard specifies a common reference model to exchange data configuration of Intelligent Electronics Devices (IEDs). This exchange is made possible by the use of the Substation Configuration Language (SCL) files that allow for the transfer of IED configuration from one software engineering tool to another to integrate an IED within the Substation Automation System (SAS) (Martin and Nguyen, 2004).

The benefits that are derived from the IEC 61850 standard are:

- Introduction of a standardized substation architecture and communication protocol in terms of addressability via a communication bus (*reducing cabling cost)*.

- Interoperability of equipment provided by different vendors.
- Easy maintenance and reconfiguration of the overall architecture of the substation and/or functions perform by all associated equipment.
- Capacity added within the substation to detect faults or inactive IEDs within its own architecture. This prevents the substation from mal-operation due to faulty IEDS left unattended within the substation architecture, under the assumption that they are working properly.

The parts of the IEC 61850 standard that are used within this research work are reported in Table 3.9.

Table 3.9: IEC61850 sections to be considered for the current project

Considered section of the standard	Comments
IEC 61850-1: Introduction and overview	Provide a general overview of the standard
IEC 61850-2: Glossary	Very important to quickly find the meaning of an abbreviation or a concept.
IEC 61850-5: Communication requirements for functions and device models	Provide the basic communication requirements for functions and device models
IEC 61850-6: Configuration language for communication in electrical substations related to IEDs.	It is required to have a deeper understanding of the SCL to design a custom ICD file.
IEC 61850-7-1: Basic communication structure for substation and feeder equipment - Principles and models.	Show the basic communication structures model and principles behind it.
IEC 61850-7-2:- Abstract communication service interface (ACSI).	Show the hierarchical structure of a logical device to implement and the service interface.
IEC 61850-7-3: …- Common Data Classes.	Present all the common Data classes; Data Attributes semantics and functional constraints.
IEC 61850-7-4: ...- Compatible logical node classes and data classes.	Present all existing type of Logical nodes and their Data Object semantics.
IEC 61850-8-1: Specific communication service mapping (SCSM)- Mappings to MMS (ISO/IEC9506-1 and ISO/IEC 9506-2) and to ISO/IEC 8802-3.	Present the Mapping to MMS and to ISO/IEC 8802-3. (for GOOSE)
IEC 61850-10: Conformance testing.	Detail the design path to verify correct implementation of the standard.

It is important to please note that the following are out of the scope of the IEC 61850 standard:

- Control logic and other PLC functionality of IEDs;
- Binding of physical input/output to communication modules of IEDs.
- Regulations on how a programmer should write the source-code (or implement the algorithms) of the software (Proces_Application.exe) that runs within an IED.

The next chapter introduces the main hardware target included in the DK61 embedded development kit.

CHAPTER FOUR:

CASE STUDY IMPLEMENTATION: CONTEXT AND DEVELOPMENT ENVIRONMENT

4.1 Introduction

This chapter presents the DK 61 embedded kit development environment (hardware and software) from Beck IPC and the context within which it will be used.

Section 4.2 deals with the project context.

Section 4.3 gives a general description of the common IED structure before introducing the DK61 embedded development kit (Hardware and RTOS) that hosts the software application for an Intelligent Circuit Breaker Controller. The embedded system will perform its role of controller through implementing a limited subset of Logical Node functions which a fully fledged IED multi-function IED might contain. An overview of the IEC 61850 communication stack used in this work is presented in Section 4.3.7 followed by a brief review of different substation hierarchical views in Section 4.4.3.

Section 4.4 comments on the general description for industrial circuit breakers as a physical device in section 4.4.1, then discusses the virtualization of the physical device through the IEC 61850 modelling process, and the tools and method used.

The chapter ends with section 4.5 – the conclusion

4.2 Project context

The current research work focuses on the experimental development of an actuator for a circuit breaker depicted in Figure 4.1 (highlighted in pink). This is achieved using an embedded platform that is presented in Section 4.3.3. The use of GOOSE messages to demonstrate the protection characteristics using a protection scheme - the functioning of which is distributed between a commercially available IED and the application developed on the embedded-systems platform, is of main concern.

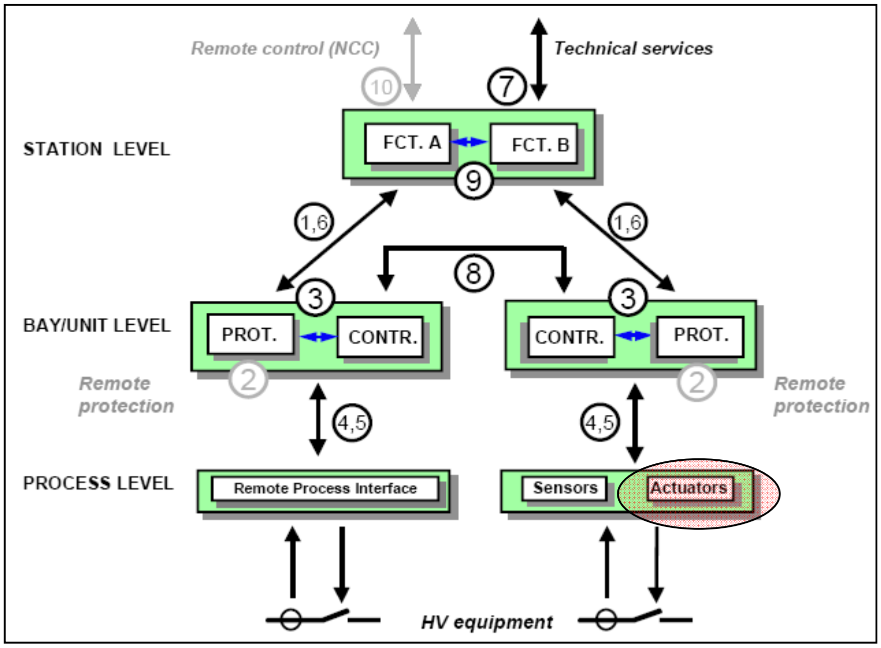


Figure 4.1: Interface model of a substation automation system
(IEC 61850-5, 2003, p.15)

Interfaces eight, four and five have been defined in IEC 61850-5 as follows (For a detailed description of all interfaces consult Section 3.4.4):

IF8: direct data exchange between the bays especially for fast functions such as Interlocking;

IF4: CT and VT instantaneous data exchange (especially samples) between process and bay level; and

IF5: control-data exchange between process and bay level.

The process bus has originally been defined for Sampled Values, but not exclusively so. GOOSE messages can also be used at the process bus level for an instantaneous trip (from bay to process level) or to send the status position of a circuit breaker element for example (Sidhu et al., 2011a).

An industrial implementation of the IEC 61850 standard-specified interfaces in Figure 4.1 could be of the form given in Figure 4.2 in which the considered actuator has been identified as being a breaker-IED. For the definitions with regard to Figure 4.2 refer to the “Glossary of Terms” at the beginning of this document (page xvii).

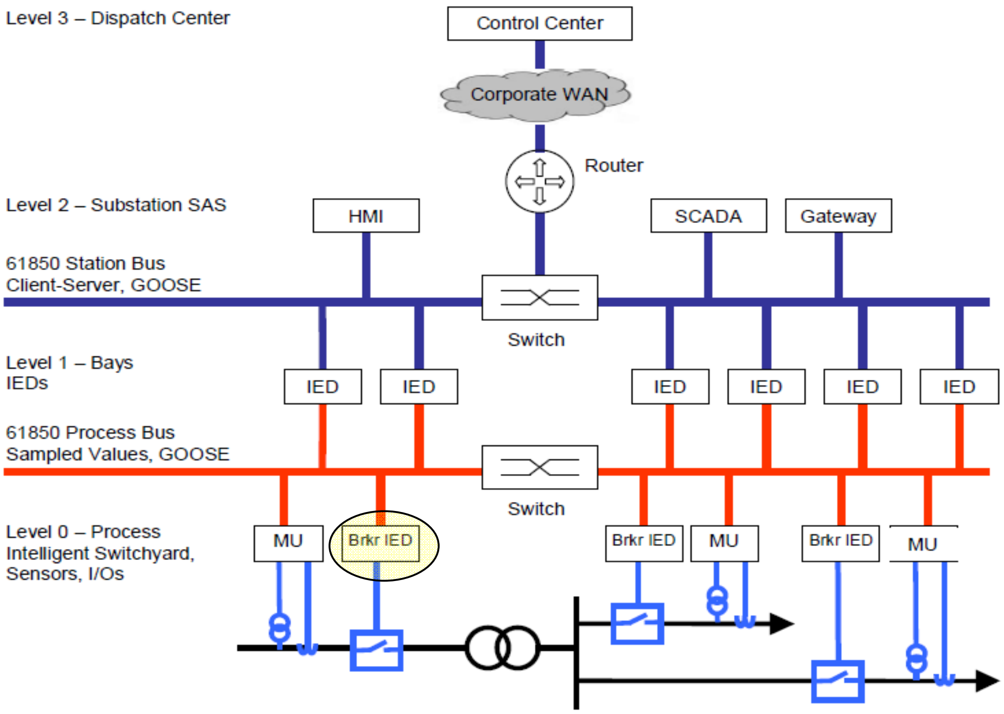


Figure 4.2: Architecture of IEC 61850 SAS with Station Bus and Process Bus
(McGhee and Goraj, 2010)

Note that when GOOSE and SV messages are present on the same communication bus it is crucial to segregate the different traffic types since Sampled Values needs to have its own bandwidth allocation. This can be achieved at the network switch level with the configuration setup of two VLANs. Each of these VLANs (carrying GOOSE and SV separately) is required to have its own ID (VID) (IEC 61850-9-2, 2004, p.14).

The current situation in industry is though not as depicted in Figure 4.2, but is as depicted in Figure 4.3 (See also Figure 3.19, p63). In such an implementation where there is no process bus - the primary equipment (breaker actuators, CTs and VTs) are connected to secondary equipment at the bay level (bay controller, protection relays etc.) using long lengths of copper cables hardwired in place.

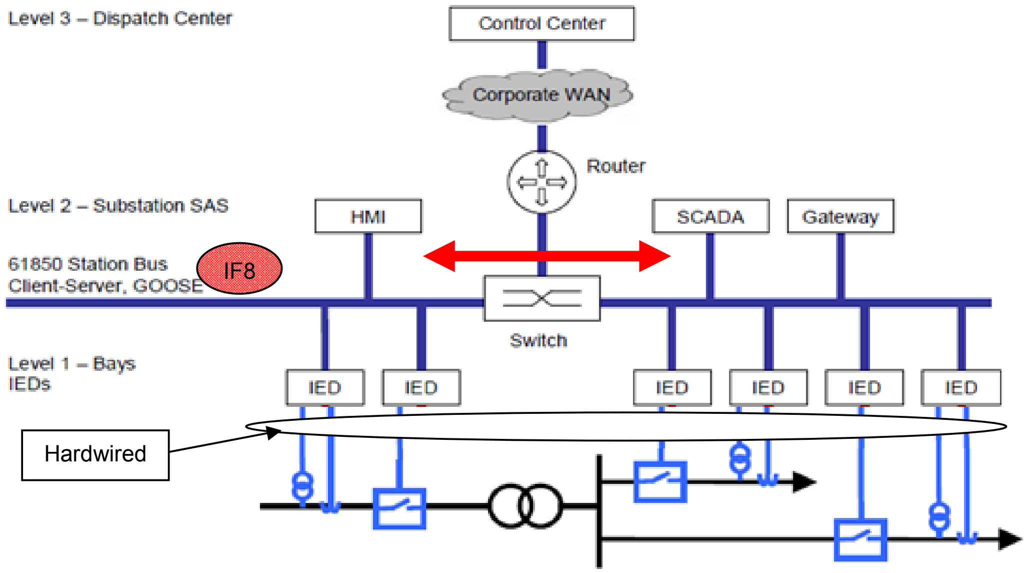


Figure 4.3: Current SAS architecture implementation
Modified from (McGhee and Goraj, 2010)

Figure 4.4 represents the possibility for use of distributed or non-distributed schemes: Scenario (a) – only IED_1 used and assumed connected to trip a circuit breaker: If current and voltage transformers are connected to the CTs and VTs elements (TCTR and TVTR) of IED-1 at the bay unit, and the circuit breaker logical node is also part of the same physical device, a fault detected by the considered device can cause the device to issue a Trip command to the circuit breaker (if programmed to do so, and if the circuit-breaker is hardwired to IED_1). The protection scheme is then said to be non-distributed – all actions and commands under control of IED_1. This operation does not require the use of the GOOSE message exchange.

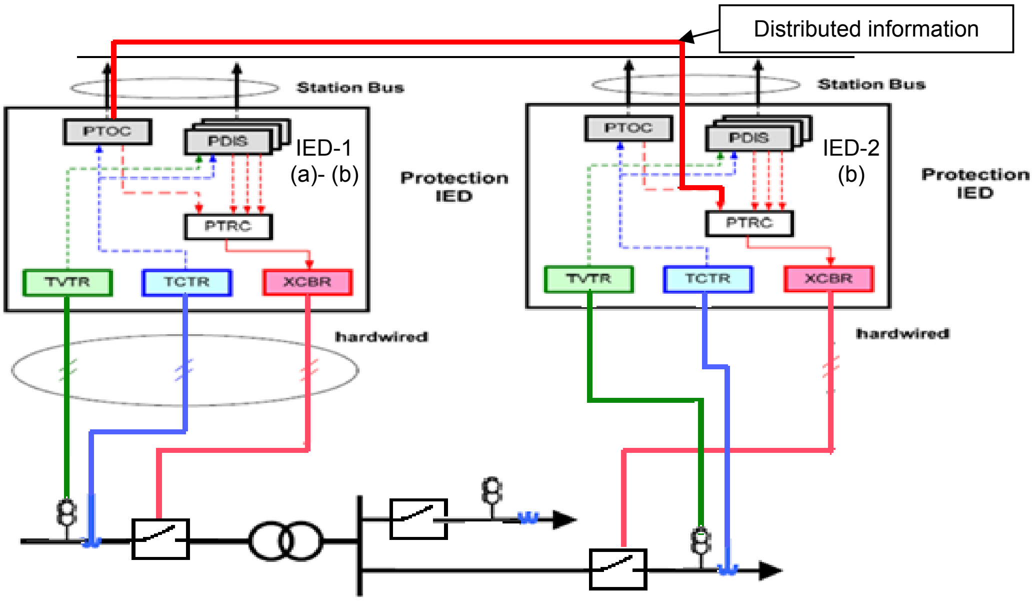


Figure 4.4: Non distributed (a) vs distributed scheme (b)

Scenario (b) – IED_1 connected to IED_2 across a communications channel and IED_2 assumed connected to trip a circuit breaker: IED-1 can send critical information via GOOSE message to IED-2 that is hardwired to a conventional circuit breaker controller for an interlocking or tripping action to be processed. In this case, the protection application scheme is then said to be distributed across IEDs (see Figure 4.4).

The distributed application mentioned in Figure 4.4 would normally be implemented on a peer-to-peer level using communication of information across interface 8 (between bay-level entities) mentioned in Figure 4.1.

The scope of this research project proposes an embedded-systems based actuator deployed at the process level (highlighted in pink on Figure 4.1), and connected to an emulated circuit breaker. Communication with a commercially available IED placed at bay-unit level is now assumed to be communicating with the embedded-system platform at the Process level (Figure 4.1).

Figure 4.5 ilustrates the topology for the case study that is of concern for this research work.

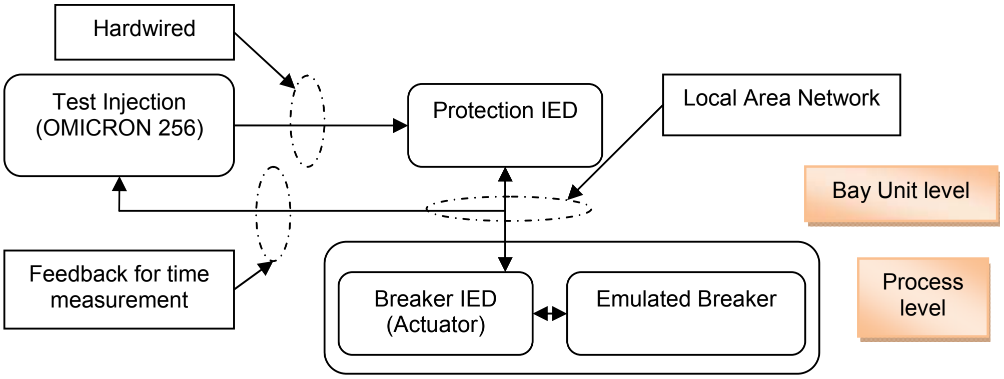


Figure 4.5: Project case study

In the sections to follow the Actuator and emulated circuit breaker depicted in Figure 4.5 above are described. The structure of a commercially available IED versus the embedded system development platform which will be used on the research project is discussed in section 4.3, with the data modelling and hardware interfacing of the emulated circuit breaker discussed in section 4.4.

4.3 Introduction to the DK61development kit

Sections 4.3.1 through 4.3.2 briefly define and review the basic structure of a commercially available station bus IED (protection IED).

Sections 4.3.3 through 4.3.7 elaborate on the DK60 embedded systems development kit used on the research project – its description, the microcontroller and its development environment, RTOS utilized, and the communication stack.

4.3.1 Definition of an Intelligent Electronic Device

The term “Intelligent Electronic device” refers to microprocessor-based relays introduced in the substation in the 1980’s.

According to the IEC 61850 standard an Intelligent Electronic Device (IED) is “any device incorporating one or more processors, with the capability to receive or send, data/control from, or to, an external source, for example electronic multi-function meters, digital relays, controllers” (IEC 61850-2, 2003). An IED can also be “a device capable of executing the behaviour of one or more, specified logical nodes in a particular context and delimited by its interfaces” (IEC 61850-2, 2003). As per these definitions and their physical location within substation communication network structure, IEDs devices can be grouped into three categories as follows:

- Station level- IEDs: A local or remote Human Machine Interface (HMI) based computer is usually available at the top of the hierarchy of the substation. The computer where the HMI is installed constitutes a full IED device on its own.

- Bay level- IEDs: Protection relays available at the bay level are multifunctional IEDS able to perform protection, measurement (…) and control function.

- Process level- IEDs: The IEC 61850 has introduced new type of IEDs with the definition of the process bus communication. Those new IEDs are called *Merging Unit* and *Actuator*. The “actuator” categories can be divided into two sub-types of device: Intelligent Circuit Breaker controller and Intelligent switch controller.

The actuator node prototype for the circuit breaker developed in this research work falls into the category of Process level- IEDs. It is important to understand how the introduction of this new type of device modifies the design structure of the conventional circuit breaker. A more detailed explanation about the circuit breaker IED is in Section 4.4.1.

4.3.2 General structure of Intelligent Electronic Devices

Figure 4.6 shows the internal architecture of a typical IED (basic protection relay) that could be used at the station bus (assuming no process bus). In this figure, generic elements needed to build an embedded system are displayed. Elements that are specific to the domain of substations are also detailed (e.g. Trip module CTs and VTs).

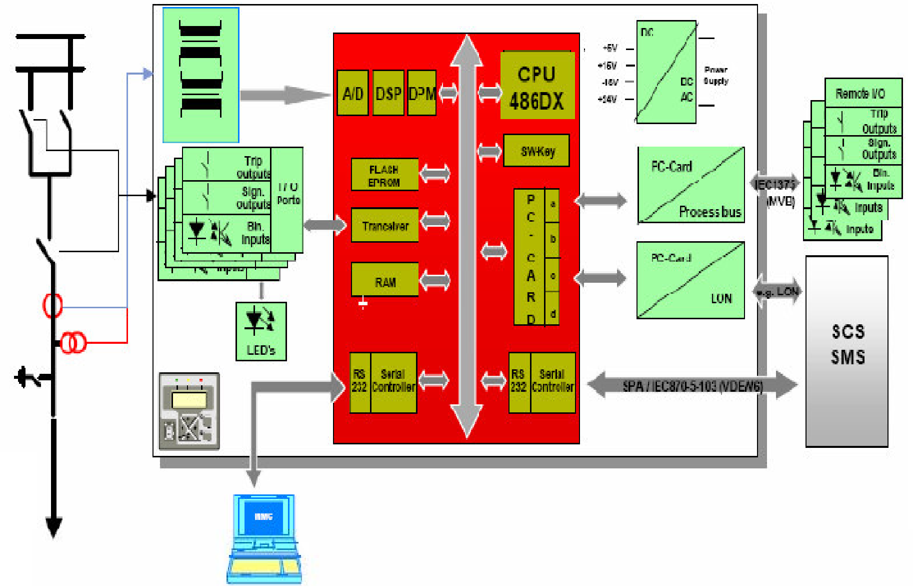


Figure 4.6: Internal structure of IEDs
(ABB- KIRRMANN H., 2004)

Protection of the control motherboard circuit is often achieved through optical isolation.

4.3.3 DK 61 Development Environment description

For a developer an intuitive way to advance in the development of an IEC 61850 application is to identify and define the problem clearly. This is very important as the IEC 61850 is a standard dealing with multidisciplinary issues. Failing to identify the proper implementation issue or solution requirement might result in time wasted on investigating information that is not relevant.

Firstly, for the project development - identification of the product platform is crucial (SystemCorp_b, 2010):

- Is the product platform, a hardware-specific application (e.g. ARM, Coldfire or Power PC)?
- Is Linux the defined operating system on the product platform?

- Is Window's PC platform (IEC 61850 DLL) a viable option?
- Is there a requirement for Linux X86 PC platforms?
- Is the IEC 61850 single-chip solution (DK61 Kit) viable?
- What telecommunication hardware technology is required for your product (Optic-fibre, wireless, cable…)?

Secondly, the application deployment has to be investigated:

- Is there a requirement for MMS only (Lite), or MMS with Publisher/Subscriber, GOOSE, Reports, Logging and Sample Values (extended)?

Finally, the application communication models require attention:

- Is there a need for the Server only, Client only, or a Server/Client application?

Answers to these questions justify the identification of a chip with an on-board IEC 61850 communication stack (see Figure 3.14, page 58). The Beck IPC Embedded web controller (SC143-IEC-LF) is included in the DK61 development kit and has been identified as a viable option.

The DK61 development kit presented in Figure 4.7 is a development environment that allows for the development of different types of software applications that require network interfaces and a special communication stack for fast and reliable control commands. It has been developed by Beck IPC GmbH[19] (http://www.beck-ipc.com).

The IEC 61850 protocol stack delivered with the DK60 board is developed and maintained by SystemCorp (http://www.systemcorp.com.au/). Online documentation of the IEC 61850 Application Programme Interface (API) that allows for fast development of process applications using the "C" programming language is available at the following address http://www.systemcorp.com.au/PIS10API/index.html.

[19] BECK was founded in Wetzlar in 1992, and is now based in Pohlheim-Garbenteich near Gießen (Germany) (Beck IPC-GmBH, 2010)

Figure 4.7: DK61 Development Kit
(Beck IPC-GmBH, 2010)

For more details on the composition of the DK61 Kit and the proposed method to start using it, please refer to APPENDIX M.

All documentation related to the DK 61 kit can be freely downloaded from Beck IPC website at http://www.beck-ipc.com/en/download/index.asp.

There is also online documentation for the Real Time Operating System API at: http://www.beck-ipc.com/files/api/scxxx/index.htm?cache=850807617.

The DK60 board is the targeted embedded system platform in this project. It represents the basic IED hardware, except that it does not have an Analogue to Digital Converter (ADC), and other circuitry to interface to CTs, VTs. The DK60 board was first commissioned with the SC123 chip in 2005 (Beck IPC-GmBH, 2010). It is now produced with the SC143 chip. Figure 4.8 gives an overview of the internal hardware architecture of the DK60 board.

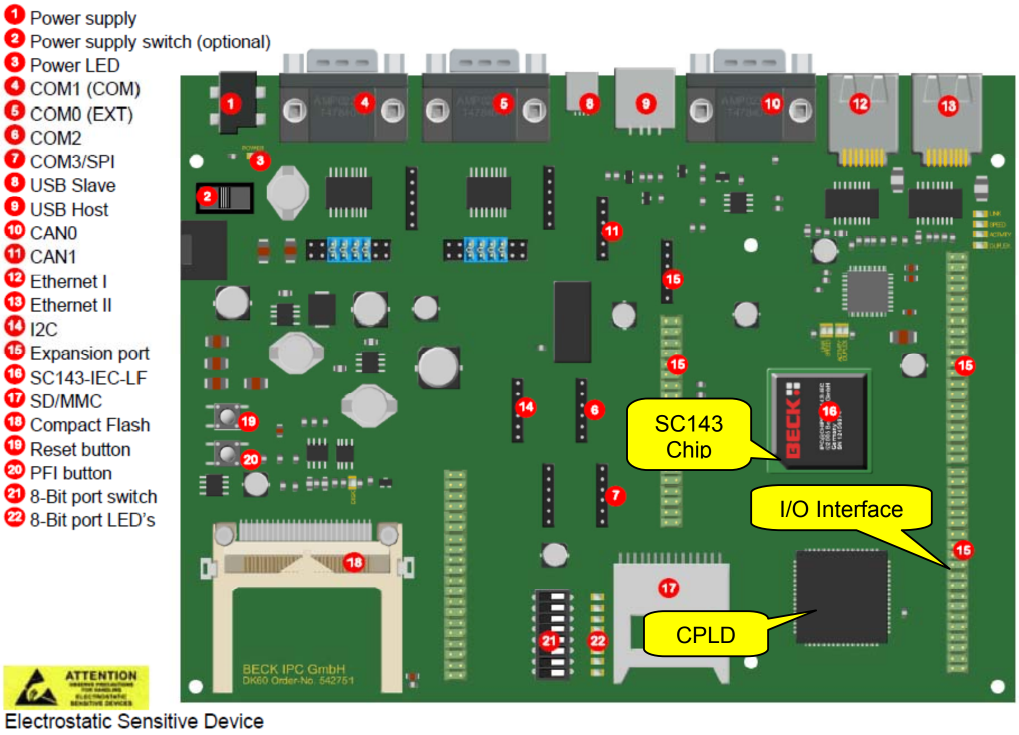


Figure 4.8: DK60 board standard configuration
(Beck IPC-GmBH, 2010)

Please refer to the manual of the DK60 board for more details.

4.3.4 Tools for development of an IEC 61850 application

The IEC61850 application developed in this project is deployed on the DK60 board using the following software tools:

- Paradigm C/C++:

The software application code for the intelligent circuit breaker node is written in C and compiled with Paradigm. The software development tool called Paradigm C/C++ is a tool that allows for code development, executable generation and debugging of software applications for embedded systems. With Paradigm C/C++, an executable can be made to suite a particular memory model type. A memory model can be Small, Medium, Compact, Large or even Huge (especially for IEC61850 application) (Paradigm, 2010). Please note: after installation of Paradigm C/C++ for Beck IPC chip the latest CLIB version must be installed.

- ICD Designer and XML Marker:

ICD Designer and XML Marker are both software tools used to design files based on XML syntax. ICD Designer is the provided for use with the DK60 board, but this software is not free. XML Marker is a general XML tool for XML file manipulation and is available free of charge. For security reasons, the installation of the full version of ICD Designer software is conducted with the online technical assistance of SystemCorp and therefore will not be described in this document. Please consult http://www.systemcorp.com.au/or the SystemCorp website for documentation on the use of ICD Designer.

- Chiptool:

The Chiptool application is the main software configuration tool for communication with the SC143 on the DK60 board. It is a Beck IPC specific tools that provides a number of services directly related to the DK60 board such as the "ping" function, Telnet, FTP, HTTP, IP configuration and much more.

4.3.5 The SC143 Embedded Web microcontroller

The IPC@CHIP® SC1X3 family is a combination of hardware and software including the pre-installed real time operating system, TCP/IP stack, web server, FTP server and Telnet server (Beck IPC-GmBH, 2010). The SC143 microcontroller chip of Beck IPC is an advanced web microcontroller used mainly in the telecommunication industry. Using the SC143 chip allows for deployment of embedded software in C/C++ as well as Programmable Logic Control (PLC based CoDeSys/IEC61131-3) within the DK60 project development environment. The SC143 Chip microcontroller is an embedded controller that has a 16-bit 186 processor on-board and is suitable for automated systems that requires Web-based or LAN communication possibilities. The operating frequency is configurable from 25 MHz up to 96 MHz (Beck IPC-GmBH, 2010). Table 4.1 summarise some of the important characteristic of the SC143 chip.

Table 4.1: SC143 basics characteristics

Characteristics (SC143)	Value
Processor type	16-bit 186
Clock frequency	Programmable PLL provides up to 96 MHz using one 25MHz Clock
Data Bus	16-bit
Address Bus	24-bit
RTOS type	Proprietary inspired by DOS
Software compatibility	80C186 microcontrollers SC186/AM186 (x86 instruction set)
Embedded Ethernet controllers	Two 10/100Mbps Ethernet Controllers with one built-in PHY and one MII PHY interface, 32 byte FIFOs

Additional information is available in the SC143 manual on the Beck IPC website. Figure 4.9 presents the SC143 block diagram.

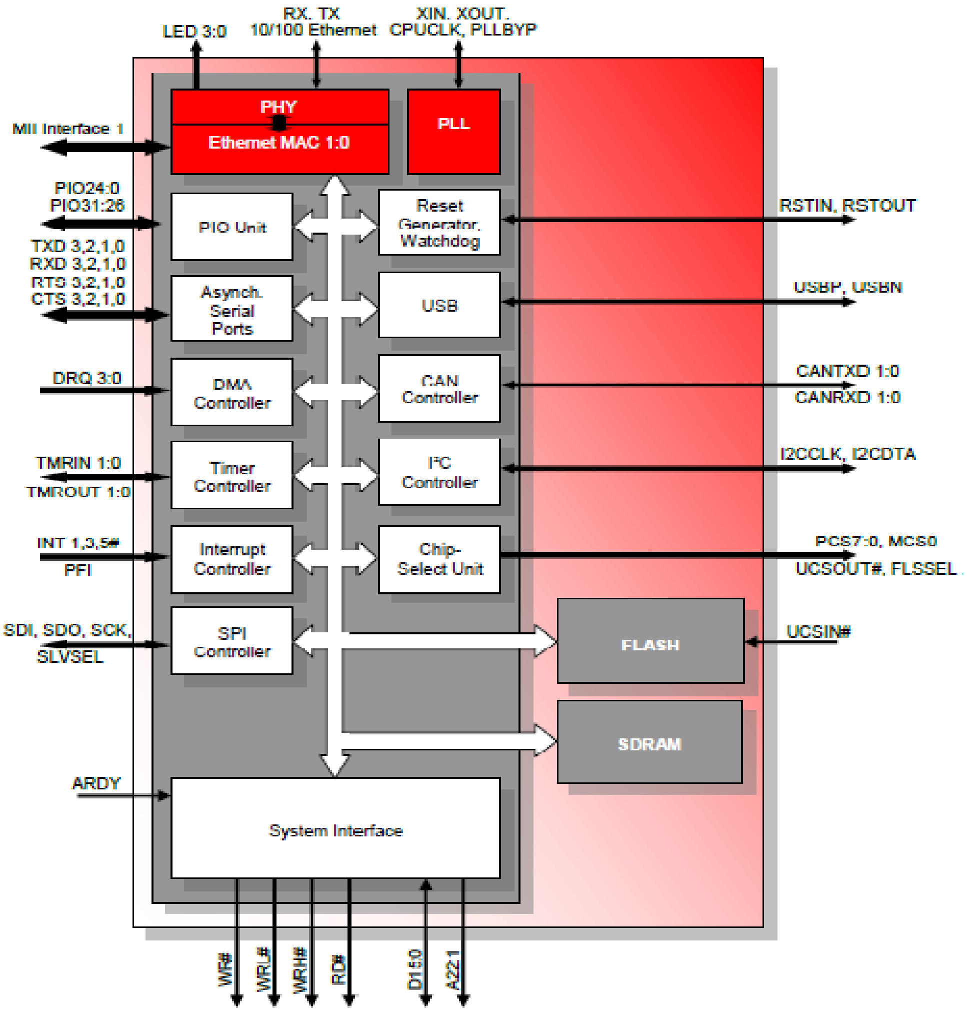


Figure 4.9: SC143 Block diagram
(Beck IPC-GmBH, 2010)

Note that hardware logic functions outside the scope of the SC143 chip are implemented on the DK60 board using Complex Programmable Logic (CPLD). The CPLD type on the DK60 board is XILINX XC95144XL-5C. The hardware reconfiguration of the DK60 board is implemented using the SPI/S37 interface on the DK60 board with original VHDL files from Beck IPC and appropriate tools from Xilinx (appropriate parallel cable IV and the XILINX ISE WebPACK 6) not included in the DK61 Kit. Additional information is provided in a separate application note available on the Beck IPC website. The complete VHDL code for this project is available in APPENDIX E .

In this project, a CPLD address decoder is used to read a certain memory area of the SC143 chip and duplicate the corresponding value to the output unit via the expansion port header (see Figure 4.8).

4.3.6 IPC@CHIP® RTOS architecture description

The IPC@CHIPRTOS is a proprietary Real Time Operating System compatible with most basic DOS commands. Figure 4.10 shows the internal software architecture that comprises the preloaded RTOS for an SC1x3 chip.

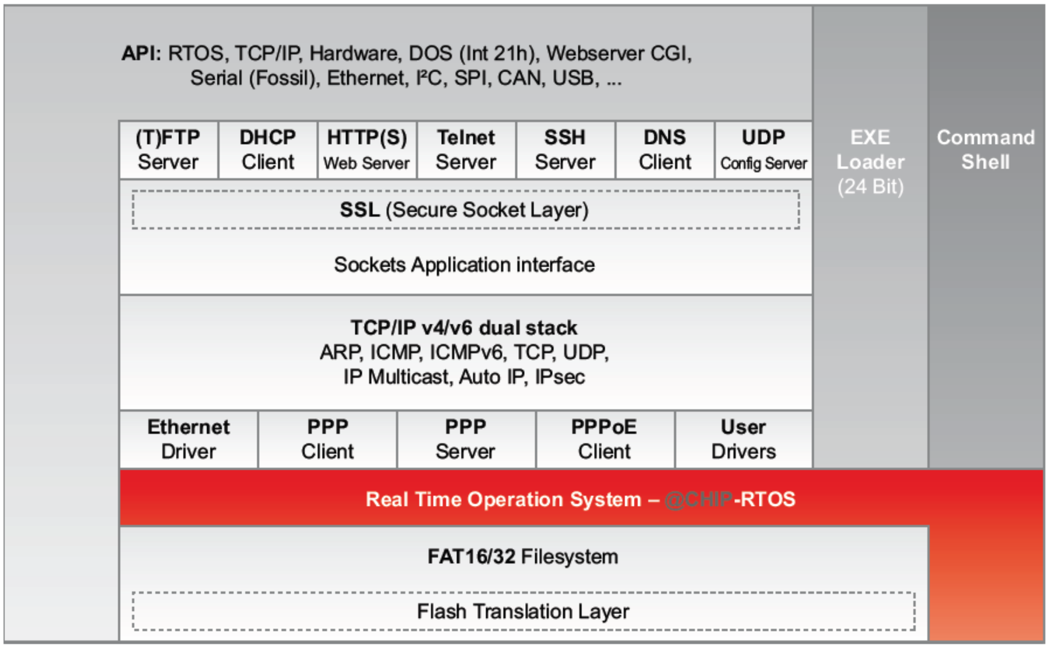

API: RTOS, TCP/IP, Hardware, DOS (Int 21h), Webserver CGI, Serial (Fossil), Ethernet, I^2C, SPI, CAN, USB, ...
(T)FTP Server
DHCP Client
HTTP(S) Web Server
Telnet Server
SSH Server
DNS Client
UDP Config Server
SSL (Secure Socket Layer)
Sockets Application interface
TCP/IP v4/v6 dual stack
ARP, ICMP, ICMPv6, TCP, UDP, IP Multicast, Auto IP, IPsec
Ethernet Driver
PPP Client
PPP Server
PPPoE Client
User Drivers
EXE Loader (24 Bit)
Command Shell
Real Time Operation System – @CHIP-RTOS
FAT16/32 Filesystem
Flash Translation Layer

Figure 4.10: IPC@CHIP® RTOS architecture
(Beck IPC-GmBH, 2010)

In order to facilitate the development of a software application, different Application Program Interfaces (APIs) are available. It should be noted that specific “C” header files known as “@CHIP-RTOS CLIB” (see section 4.3.4) first need to be installed to make use of these APIs. These header files include standard and Beck-specific function definitions. As there are more than fourteen distinctive groups of APIs for the IPC@CHIP RTOS, explicit details for each of them cannot be presented here.

Table 4.2 provides a list of APIs for the IPC@CHIP RTOS.

Table 4.2: API list for the IPC@CHIP® RTOS

API Denomination	Description
BIOS API	Interface definition for the BIOS Interrupts.
CAN (Controller Area Network) API	The CLIB API definition for sending and receiving data on the CAN (Controller Area Network) buses
CGI (Common Gateway Interface) API	The CGI ("Common Gateway Interface") API provides access to the CGI implementation of the IPC@CHIP® Web server.
External Disk API	CLIB API for an external disk B: and D: drive. This interface allows you to add an external B: or D: drive
Fossil API for Serial Ports	API definition for access to the serial ports.
Helper Functions	This API includes Character Output Functions, Memory Allocation Functions and Data Type conversion functions.
Hardware (PFE, HAL) API	API functions for access to the IPC@CHIP®'s hardware. e.g. pfe_enable_pcs-> Enable programmable chip selects
I2C Bus Interface API	CLIB API for access to the IPC@CHIP®'s I2C bus
Packet Driver API	Packet driver API for direct access to the IPC@CHIP® Ethernet device.
RTOS API	This API provides access function to the RTOS of the IPC@CHIP e.g. RTX_Create_Task ->Create and start a task
SPI API	CLIB API for access to the IPC@CHIP®'s SPI bus.
TCP/IP API	This API provides access to the IPC@CHIP® TCP/IP stack's socket interface for programming TCP/IP applications.
USB (Universal Serial Bus) API	CLIB API of the IPC@CHIP® RTOS dual-mode USB-driver.

Additional information regarding the IPC@CHIP RTOS APIs can be obtained from the online documentation available athttp://www.beck-ipc.com/files/api/scxxx/.

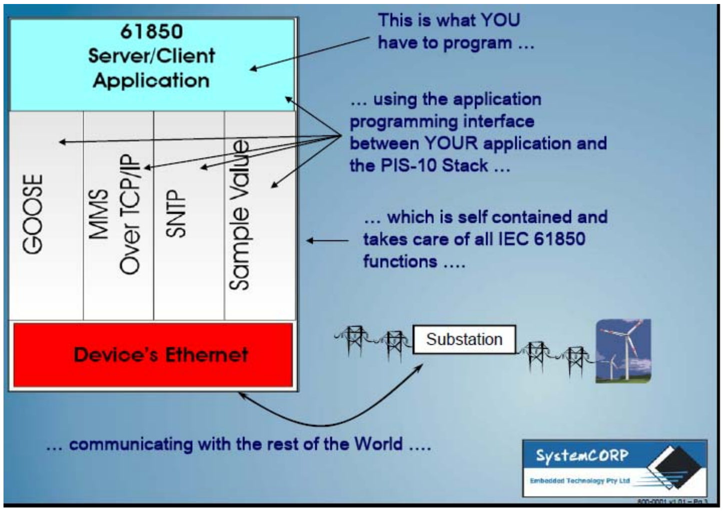
Also note that in addition to Beck specific APIs, knowledge of standard C Libraries may be used (DOS.H, STDIO.H, etc.).

4.3.7 Overview of the PIS-010 communication stack and its IEC 61850 API

The DK61 kit has an IEC 61850 Application Programming Interface (API) and a communication stack (PIS-010.lib) developed by SystemCorp. Note that the IEC 61850 API is not to be mistaken with specific Beck IPC APIs discussed in section 4.3.6. The function of the IEC61850 API and the PIS-010 stack is to provide any IEC 61850 application programmer with features that enable them to create, from a communication point of view, a functional IEC61850 application within a short period of time. The PIS-010 stack is also supported on the following platforms: Ubuntu Linux (x86,x86-64), Microsoft Windows XP,2003/2008 and Vista; Other platforms are available on request.

The PIS-010 stack library is ANSI-Standard C compatible and can be configured via SCL file. It is an event driven stack (Interrupt based), so there is no need to poll for data (SystemCorp_b, 2010).

Figure 4.11 presents the communication protocol contained in the PIS-010 stack. The MMS view over TCP/IP must be considered first while before the algorithm. The MMS communication to the GOOSE or sampled value message, or SNTP may then be associated. This is because the publisher/subscriber mechanism has been implemented within the PIS-010 stack as an added feature. Publisher and subscriber functionality are associated respectively with the server (publisher) and the client (subscriber). If there is therefore a requirement for thePIS-010 protection relay device to subscribe to an incoming GOOSE message, the client functionality in addition to the default server functionality has to be implemented.



This is what YOU have to program ...

... using the application programming interface between YOUR application and the PIS-10 Stack ...

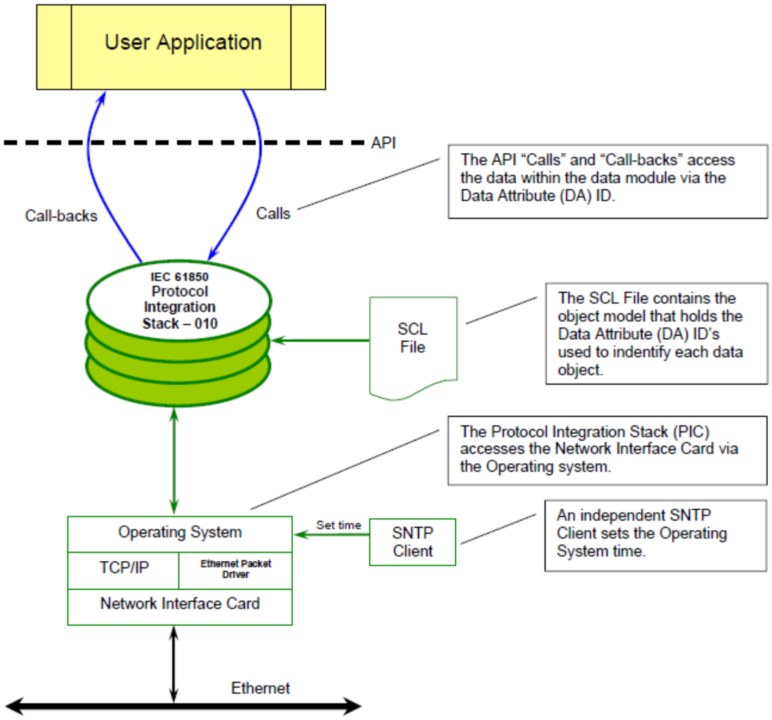
... which is self contained and takes care of all IEC 61850 functions ....

... communicating with the rest of the World ....

Figure 4.11: PIS-010 protocol composition
(SystemCorp_b, 2010)

The API provides the programmer with more flexibility when integration of the system to the main software core specific protocol of communication (GOOSE, SV, MMS, SNTP…) and other specific communication services using the IEC 61850 stack (PIS-010) is considered. Therefore the software development time is reduced and implementation is accelerated since the required application protocol is already provided within the PIS-10 stack.

Figure 4.12 demonstrates the interaction between the user applications, the API, the communication stack, relevant SCL file, the real-time operating system and the network interface card. Version 1.36 of the PIS-010 stack has been used in the implementation for the current research project.



The API "Calls" and "Call-backs" access the data within the data module via the Data Attribute (DA) ID.

The SCL File contains the object model that holds the Data Attribute (DA) ID's used to indentify each data object.

The Protocol Integration Stack (PIC) accesses the Network Interface Card via the Operating system.

An independent SNTP Client sets the Operating System time.

Figure 4.12: Interaction with the IEC61850 PIS-010 Stack
(SystemCorp_b, 2010)

Detailed documentation and a description of the IEC61850 API provided with the SC143 chip can be downloaded from the Beck website, or retrieved from the supplied CDs with the DK61 Kit (Beck_Lite).

The IEC 61850 protocol Integration stack (PIS-10), implements the ASCI layer and the SCSM on top of the OSI model. Note that the OSI model is implemented as part of the Operating System (OS) basic communication structure.

The PIS-10 stack bypasses the RTOS and creates a connection at the data link layer (via a specific driver provided by Beck IPC) to handle GOOSE and SV communication. Please refer to section 4.4.4 for a description of SCL files for this project using ICD designer.

The only way to access the PIS-010 stack is to make use of the IEC 61850 API provided by SystemCorp. The IEC 61850 API is organised in a hierarchy of functions. The User Application communicates with the PIS-010 via the IEC 61850 API using function calls and call backs. Figure 4.13 has been developed as documentation for this project so as to conveniently present the overall system information flow together with the internal IEC 61850 API organisation.

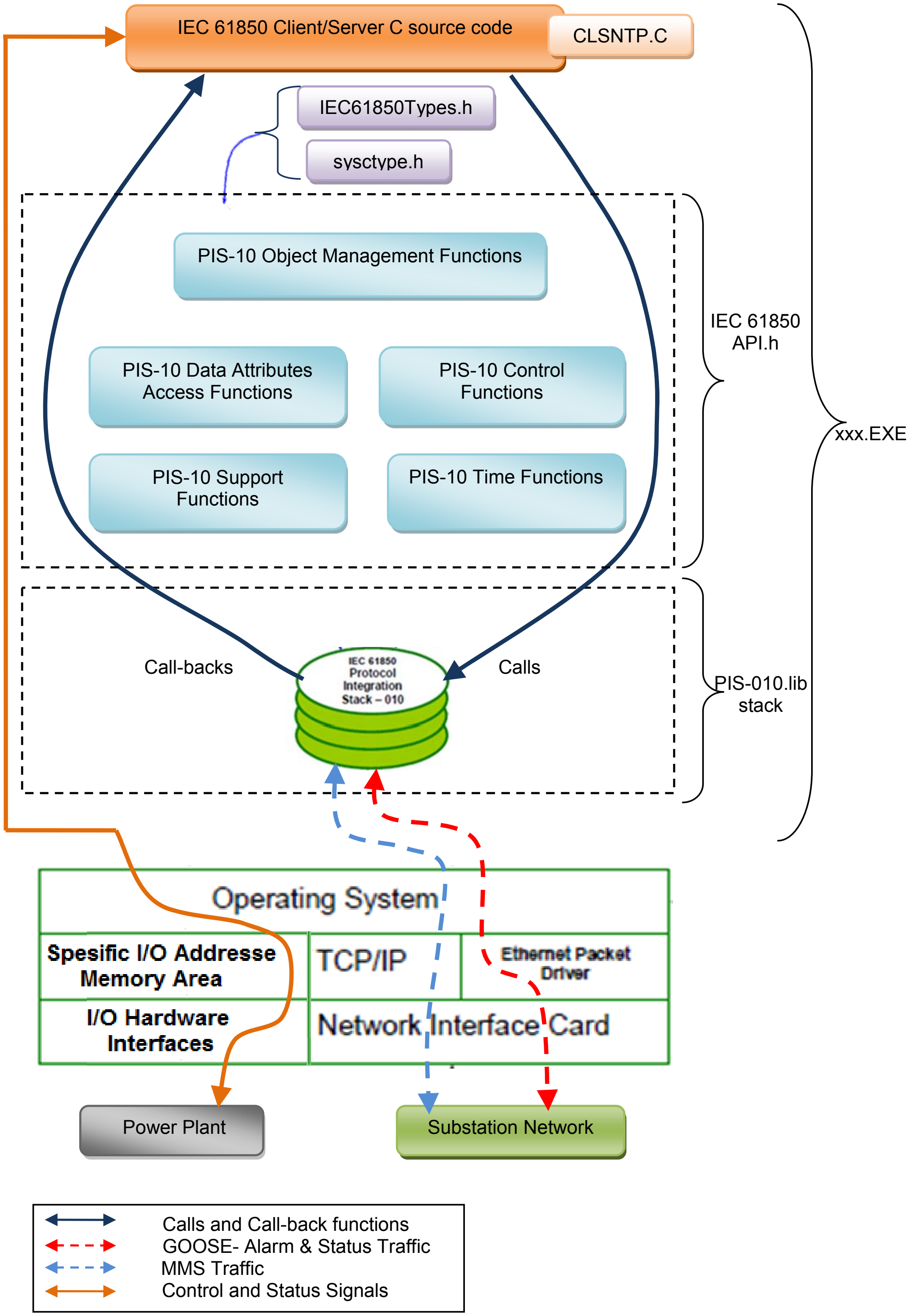


Figure 4.13: Application communication description

The “PIS-10.lib”, the “IEC61850Types.h”, the “sysctype.h” and the “CLSNTP.C” (Client for Simple Network Time Protocol) are SystemCorp intellectual property.

These files should be included by the software developer as part of the software project in Paradigm C/C++ IDE.

Please to gain access to those file you are requested to register on SystemCorp website (http://www.systemcorp.com.au/registration/registers.html).

The following tables list all IEC 61850 API functions implemented with the PIS-010 stack (version 1.36)

Table 4.3: PIS-10 Object Management Functions

PIS-10 Object Management Functions		
No	IEC 61850 API	Purpose
1	IEC61850_Create	API to create a client or server object with call-backs for reading, writing and updating data objects
2	IEC61850_LoadSCLFile	API to read the SCL XML file to get the configuration of server or client
3	IEC61850_Start	API to start the server or client
4	IEC61850_Stop	API to stop the server or client
5	IEC61850_Free	API to delete a client or server object created

Table 4.4: PIS-10 Data Attributes Access Functions

PIS-10 Data Attributes Access Functions		
No	IEC 61850 API	Purpose
1	IEC61850_Read	Read the value of a specified data attribute
2	IEC61850_Write	Write the value to a specified data attribute
3	IEC61850_Update	Update the value of a specified data attribute

Table 4.5: PIS-10 Control Functions

PIS-10 Control Functions		
No	IEC 61850 API	Purpose
1	EC61850_ControlSelect	Indicate intention to perform Operate on identified control
2	IEC61850_ControlOperate	Perform Operate function on selected control/s with specified value
3	IEC61850_ControlCancel	Cancel existing selection of controls

Table 4.6PIS-10 Support Functions

PIS-10 Support Functions		
No	IEC 61850 API	Purpose
1	IEC61850_GetLibraryVersion	Returns the version of the library
2	IEC61850_GetLibraryBuildTime	Returns build time for Library
3	IEC61850_ErrorString	Returns a descriptive string for an error code
4	IEC61850_PrintDataAttributeID	Print specified data attribute identifier to standard output
5	IEC61850_PrintDataAttributeData	Print specified data to standard output

Table 4.7: PIS-10 Time Functions

PIS-10 Time Functions		
No	IEC 61850 API	Purpose
1	IEC61850_SetTimeQuality	Set time quality information in PIS10 stack
2	EC61850_GetTimeQuality	Retrieve time quality information from PIS10 stack
3	IEC61850_SetTime	Set current time to PIS10 stack
4	IEC61850_GetTime	Retrieve current system time from PIS10 stack

4.3.7.1 Overview of a PIS-010 Server

In the field of Substation Automation Systems, field devices such as protection relays, merging units and actuator nodes are assumed to be the server devices by default. The client is usually an HMI (Human Machine Interface) situated at the top of the SAS hierarchy (section 3.4.4). Figure 4.14 depicts the state machine of a PIS-010 server. For this project, basic exchange functions (Read and Write) are used in order to exchange data information. Please refer to the online documentation for other available functions on the server side at http://www.systemcorp.com.au/ .

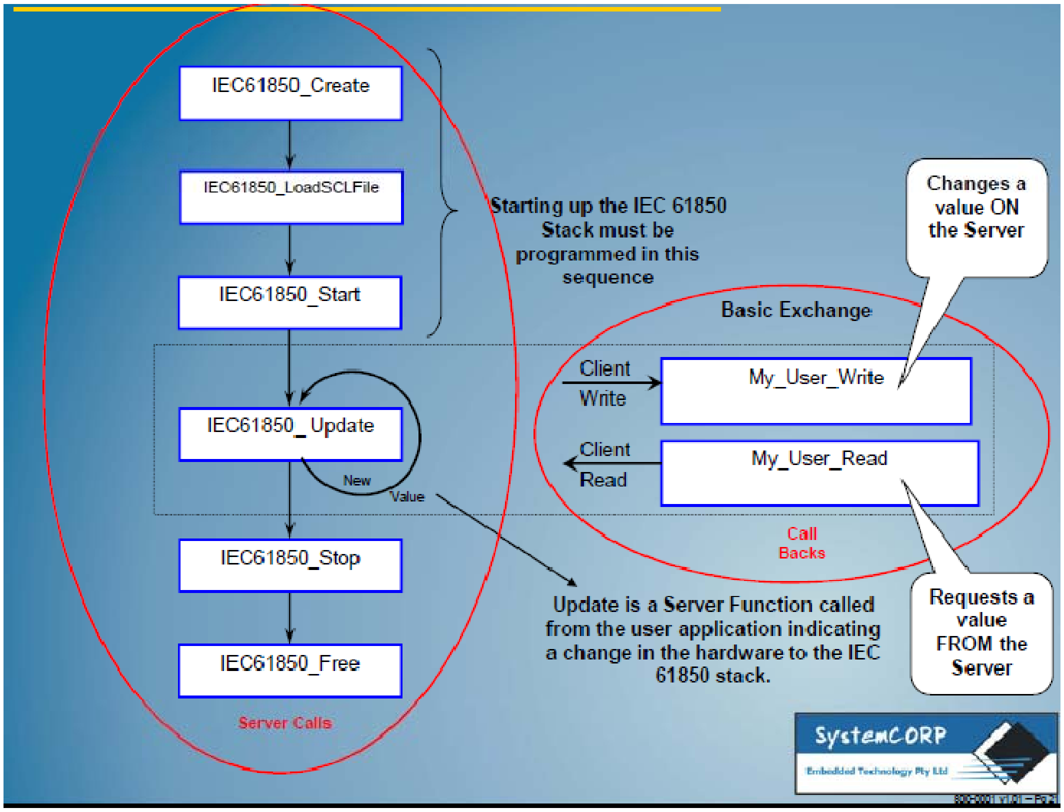
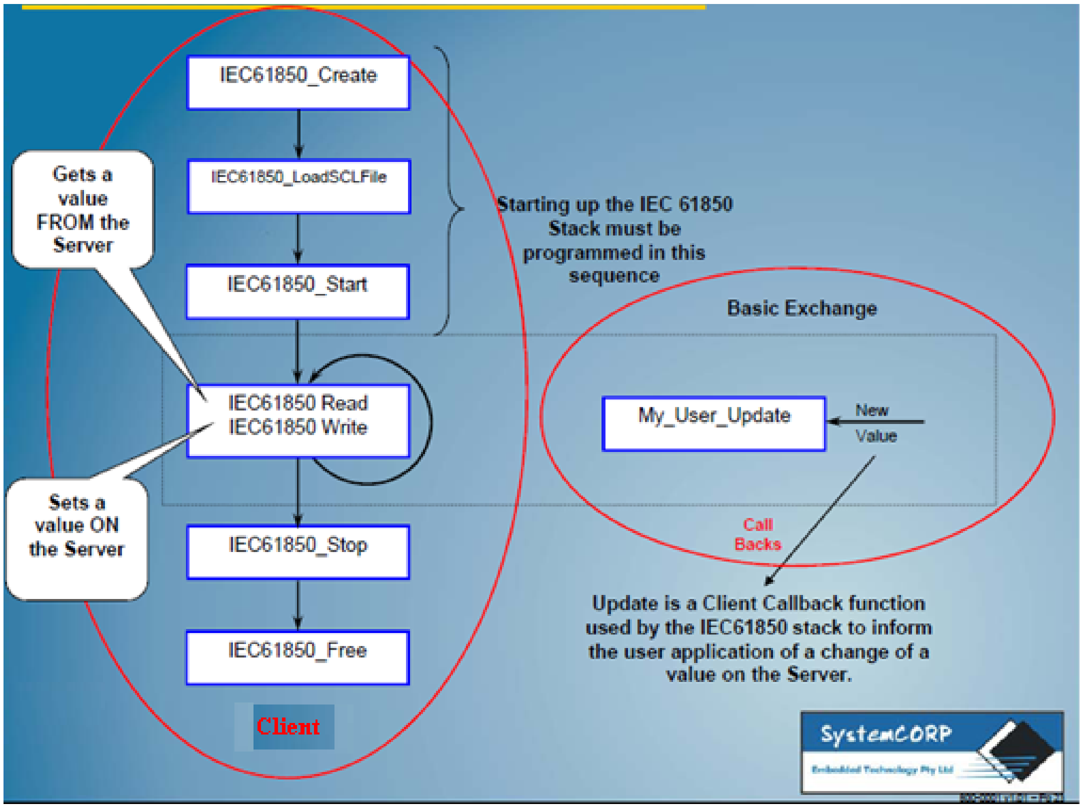

IEC61850_Create
IEC61850_LoadSCLFile
IEC61850_Start
IEC61850_ Update
New
Value
IEC61850_Stop
IEC61850_Free
Server Calls
Starting up the IEC 61850 Stack must be programmed in this sequence
Changes a value ON the Server
Basic Exchange
Client Write
My_User_Write
Client Read
My_User_Read
Call Backs
Requests a value FROM the Server
Update is a Server Function called from the user application indicating a change in the hardware to the IEC 61850 stack.
SystemCORP
Embedded Technology Pty Ltd

Figure 4.14: PIS-010 Server Overview
(SystemCorp_b, 2010)

The publisher/server is configured using a Configured IED Description file (CID). Refer to section 4.4.4.3 for more detail on engineering the system architecture using the ICD Designer tool.

4.3.7.2 Overview of a PIS-010 Client

As stated earlier, the client device is usually an HMI situated at the top of the SAS hierarchy (section 3.4.4). Figure 4.15 shows the state machine of a PIS-010 Client.



Starting up the IEC 61850 Stack must be programmed in this sequence

Update is a Client Callback function used by the IEC61850 stack to inform the user application of a change of a value on the Server.

Figure 4.15: PIS-010 Client Overview
(SystemCorp_b, 2010)

Although the focus in this research project is on the GOOSE message (publisher and subscriber), there is a need to deploy the server and client modules inside the Intelligent circuit breaker controller device in order to access the desired functionality. This is due to the internal architecture of the PIS-10 stack from SystemCorp. The client/subscriber is configured using a Substation Configuration Description file (SCD). Refer to section 4.4.4.4 for more detail on engineering the system architecture using the ICD Designer tool.

Particular details which explain the design process of each SCL files type required in this project is presented in section 4.4.4.

The next section describes the circuit breaker as a Physical Device, and the virtualization process followed in IEC 61850.

4.4 General description of circuit breaker and model development

4.4.1 General description of circuit breaker

4.4.1.1 Circuit breaker definition

A circuit breaker is an electromechanical device used to complete, maintain, and interrupt in a short amount of time currents flowing in a circuit under normal or faulted conditions (US-Department of Labor, 2010). According to ABB breaker application guide, a circuit breaker used in an electrical substation should comply with the following requirements:

- In the stationary closed position, the circuit breaker shall conduct its rated current without producing impermissible heat rise in any of its components.
- In its stationary positions, open as well as closed, the circuit breaker must be able to withstand any type of overvoltage within its rating.
- The circuit breaker shall, at its rated voltage, be able to make and break any possible current within its rating, without becoming unsuitable for further operation (ABB AB, 2010).

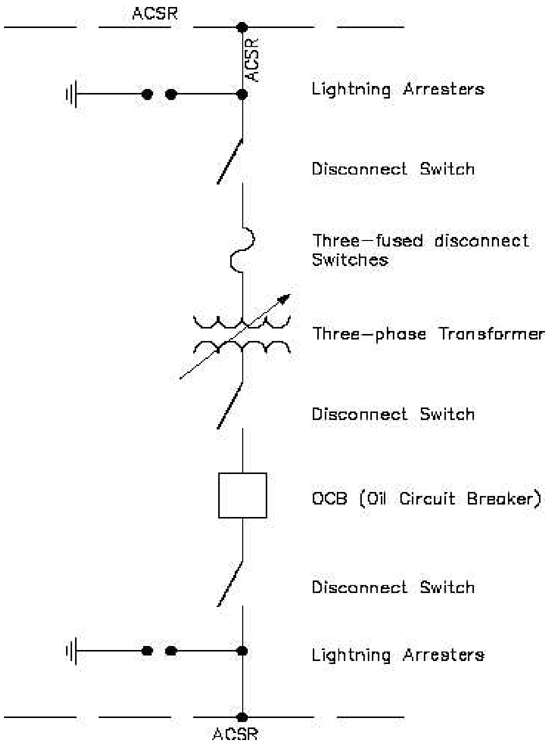


Figure 4.16: Single line diagram
(Nuclear Power Training, 2012)

From this definition it can be seen that the circuit breaker can be used for coupling of busbars, transformers, capacitor banks, transmission lines, etc.

4.4.1.2 Circuit breaker function

The main function of a circuit breaker is to interrupt fault currents and in so doing protect electrical and electronic equipment. The circuit breaker may also be used for intentional switching such as energizing and de-energizing of shunt reactors and capacitor banks, for maintenance of electrical equipment and transmission lines (ABB AB, 2010).

According to ABB, a circuit breaker shall operate under the following circumstances:

- Terminal faults, short-circuits in the vicinity of or near the circuit breaker;
- Short-line faults, short-circuits to ground along the transmission line within a few kilometres of the circuit breaker;
- Out-of-phase conditions at which different parts of the network are out of synchronism; and
- Intentional switching of capacitor banks, shunt reactor banks, no-load transformers, no-load lines and cables (ABB AB, 2010).

4.4.1.3 Types of circuit breaker

There are many types of circuit breakers from simple electronics switch based transistors via home automation relays to high voltage circuit breakers used in an electrical substation.

In general, circuit breakers found in substations are composed of two coils, one is used to close and the second one is used to open/trip the circuit breaker. These two coils are energised separately by high pulse signals to open or close the circuit breaker. Pulse commands to trip or closed the circuit breaker are transmitted from two distinct output contacts on a relay device. This ensures that the coils of a high voltage circuit breaker are not permanently energized.

The amount of current that substation circuit breakers need to interrupt, induce an arcing phenomenon at the separating contact. This arc acts as a close circuit; this means that the current will continue to flow even if breaker’s contacts are physically separated. To extinguish this arc, manufacture uses different isolating mediums and other techniques resulting in different types of circuit breakers within the substation (SF6, Vacuum, Compressed Air and Oil circuit breaker, etc.).

Nevertheless, circuit breakers in the substation can still be grouped in three main types with respect to their construction philosophy:

- Live Tank circuit breaker;
- Dead Tank circuit breaker; and
- Gas Insulated Switchgear (GIS).

Figure 4.17 illustrates the Live and dead tank circuit breaker.

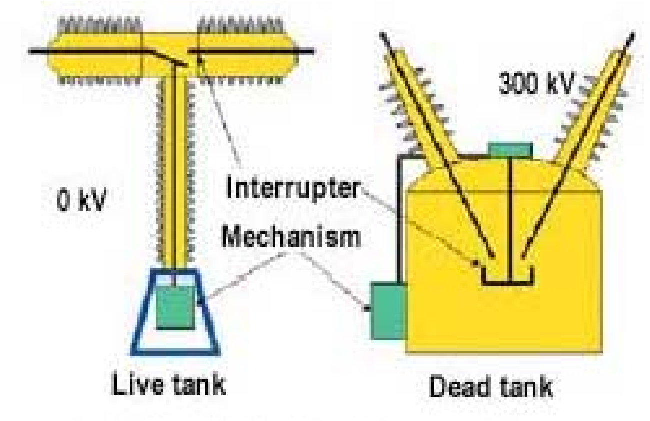
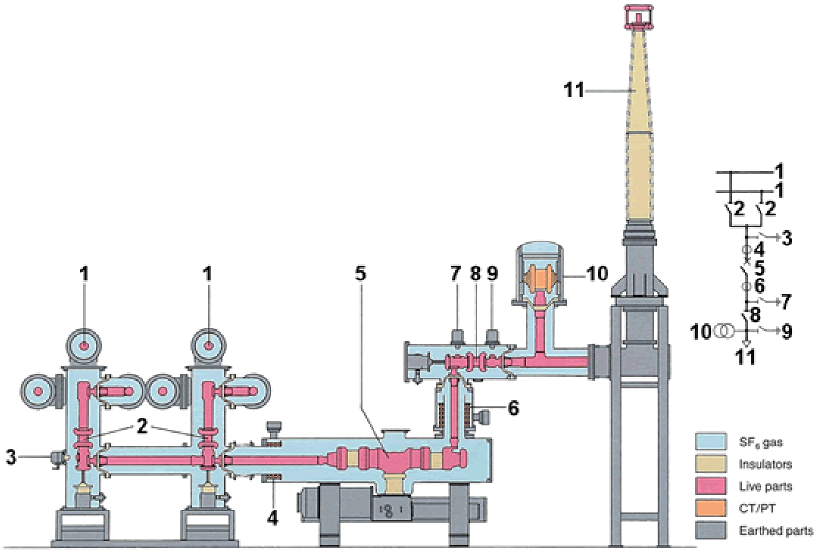


Figure 4.17: Live tank versus Dead tank
(Nasrallah et al., 2007)

Figure 4.18 illustrates a compact Gas Insulated Switchgear.



1. Busbar;
2.Disconnector;
3.Maintenance Earthing Switch;
4.Current Transformer;
5.Circuit Breaker;
6.Current Transformer;
7.Maintenance Earthing Switch;
8.Diconnector;
9.Earthing Switch;
10.Voltage Transformer;
11.Bushing

Figure 4.18:Gas Insulated Switchgear (GIS)

(TOSHIBA, 2012)

There is need to research the advantages and disadvantages of each type of circuit breaker and their variants in the substation. The results of this study could then be used to make a suitable choice for the extension of the current project to a specific substation circuit breaker. Relative adjustment of the data model and the software developed here is necessary so that this can suit the chosen power plant target perfectly.

Figure 4.19 shows only the tripping mechanism for an ABB vacuum circuit breaker controller in principle (GIS type).

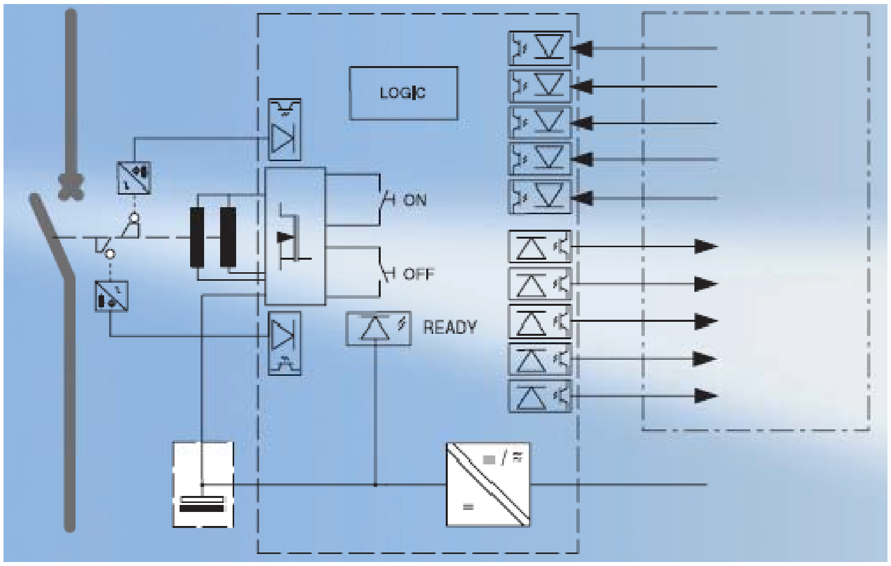


Figure 4.19: Trip module for a compact vacuum circuit breaker system
(ABB AG, 2005)

Only now can the development of a Data model for circuit breaker be started. For this project only mandatory data is included in the data model. The software (process application) design for the circuit breaker controller and other implementation issues are dealt with in Chapter five. The ICD Designer software is used to define the component of the Logical device view and communication view (Section 4.3.4).

The next section presents the steps to be taken to develop a circuit breaker data model.

4.4.2 Definition of an Actuator

An actuator in the strict sense of the IEC 61850 standard is an IED that controls a circuit breaker or switch element at the process bus level. Before the advent of the IEC 61850 standard, the common designation of an actuator for a circuit breaker was simply a circuit breaker controller. These actuators are primarily microcontroller-based. They are connected to protection relays at the bay level using hardwired cables (trip, close and status hardwired lines) as illustrated in Figure 4.20

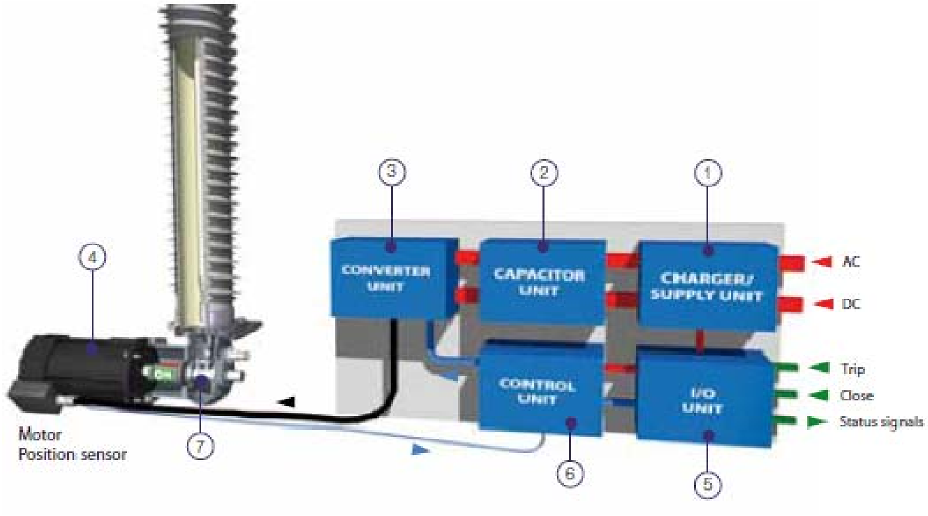


Figure 4.20: Block diagram of conventionnal Circuit Breaker Actuator (Live Tank)
(ABB AB, 2010)

The components for the circuit breaker actuator in Figure 4.20 are as follows:

1. Charger unit that converts supply voltage and feeds the capacitors and internal electronics.
2. Capacitor unit stores the energy for operation.
3. The converter unit transforms DC from the capacitors to AC for the motor.
4. Servomotor that delivers the force to move the contacts, integrated position sensor gives information to the control unit.
5. The I/O unit takes care of signalling between the station control system and the operating mechanism.
6. Control unit that controls and regulates the motion of the contacts. The interlock functions are also handled by this module.
7. Link gear that transforms rotary movement to linear movement.

There are three main parts in a substation circuit breaker. The first one is the "Circuit Breaker Controller" which acts as an actuator. The second is the actuating equipment (Circuit breaker Motor). The third part is the circuit breaker body itself which contains the contact element and a mechanism to extinguish the electrical arc.

As stated earlier the actuator similar to the one presented in Figure 4.20 is connected to the protection relay via hardwired cable. Figure 4.21 provides an illustration of the common connection between the protection relay and the primary equipment (diagram on the left-hand side). The same figure also shows the evolution introduced by the IEC 61850 standard based on process bus connectivity (diagram on the right-hand side). The process bus ensures less wiring is required to connect to the primary equipment and to reduce the global substation expenditure costs due to the high price of copper wire (assuming fibre optic is used).

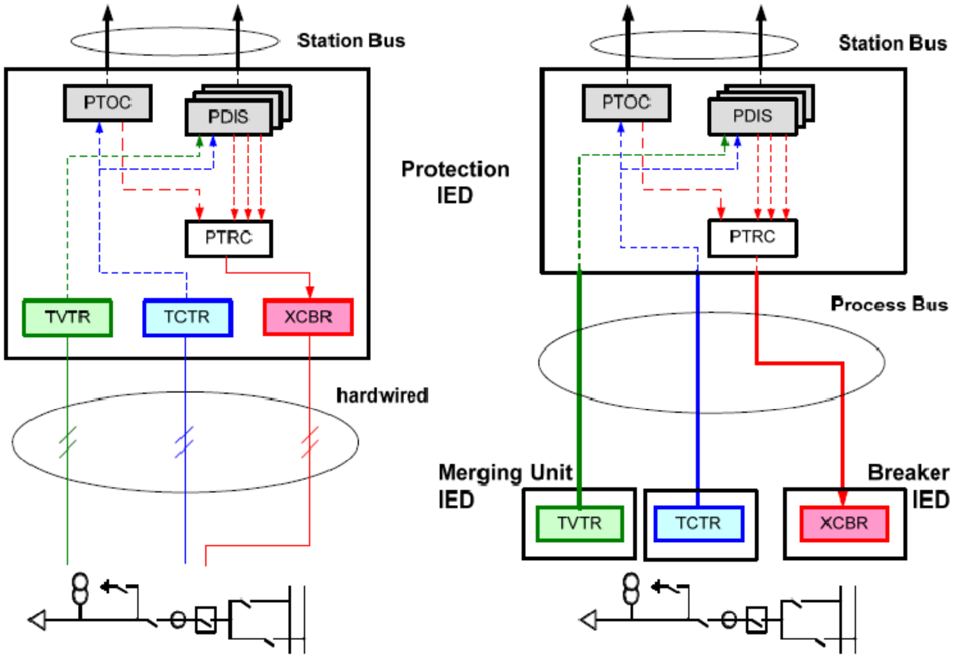


Figure 4.21: Bay protection without process bus (left) and with process bus (right)
(IEC 61850-7-500-DTR_Ed1, 2011)

With the technology evolution and the introduction of the IEC 61850 process bus, the conventional circuit breaker controller (or actuator) presented in Figure 4.20 is now provided with IEC 61850 communication capabilities. This has changed its designation from a simple circuit breaker controller to an intelligent circuit breaker controller (or breaker IED).

The definition of the *actuator node* in this particular research project refers to the logical structure that controls a substation switch element. This structure can be localised in a process level- IED (assuming IEC 61850 process bus implementation) or within an IEC 61850 compliant protection relay at the bay unit level (assuming IEC 61850 station bus implementation only). As stated earlier, the considered switch can be a circuit breaker (XCBR/ with short circuit breaking capabilities) or a simple power switch element (XSWI/ without short circuit breaking capabilities).

For this research work, it is assumed that the actuator node structure is contained within the breaker IED which is localised at the process bus. Therefore, the present IED is referred to as an Intelligent Circuit Breaker Controller or breaker IED.

The protection test-case is illustrated in Figure 4.22. The SIEMENS protection IED used for this project does not support subscription or publishing of sampled value messages. A conventional hardwired structure for a CT element is then considered.

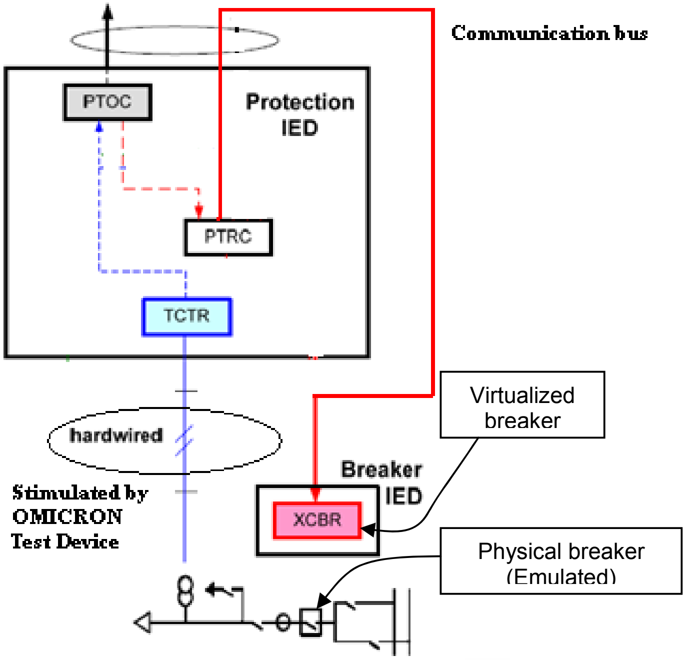


Figure 4.22: Protection scheme test case

The SIEMENS protection IED does not have dual ports that are active at the same time to handle exchange of information on two distinct communication buses. Therefore, a ring topology is then assumed- this means that the Ethernet traffic generated by the protection IED (normally at station bus level for a conventional protection IED) is assumed to be available also at the process bus.

For additional information please refer to the SIEMENS user guide “Ethernet and IEC61850 Start up” at:
http://siemens.siprotec.de/download_neu/.../IEC61850_STARTUP_A3_EN.pdf

The absence of two distinct and isolated communication buses has no impact on the data model and software development of the circuit breaker controller. This assumption enables one to focus on the development of the actuator only since the IEC 61850 standard allows for GOOSE communication at process bus level and that GOOSE communication is not based on IP address, but rather on the MAC address.

As the scope of this research work is reduced to the demonstration of GOOSE message operation, the Select-Before-Operate (SBO) method as required by the IEC 61850 standard is not implemented. Instead of the SBO method, the “Direct with

Normal Security" method is used to control the circuit breaker through a MMS command.

Process level- IEDs are developed following on some principles of the IEC 61850 standard. In general, IEC 61850 compliant circuit breaker IEDs are directly incorporated in the circuit breaker control unit box and needs to fulfil three main requirements which are:

- Backward compatibility with respect to the old hardwired techniques used to link the breaker to conventional protection relays.
- Support for IEC 61850 communication based (MMS and GOOSE). If the merging unit is also implemented within the same control unit of the circuit breaker, the Sampled Value messages will generally use a distinct Ethernet interface. When GOOSE and SV messages are on the same process bus, they will each have a distinct V-LAN ID (VID). This is due to the fact that Sampled Values need to have its own bandwidth allocation (IEC 61850-2, 2004, p.14).
- The circuit breaker IED also allows for manual operation.

These IEDs are able to switch between hardwired signals (as input control), remote control signals via the Local Area Network (LAN) based on the IEC 61850 specification and manual operation.

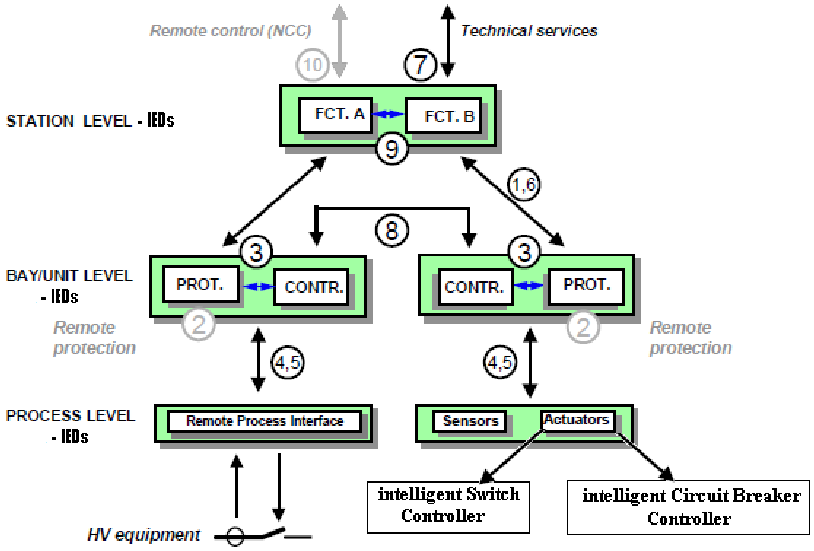


Figure 4.23: Principal type of Actuators
Adopted from (IEC 61850-1, 2003, p.13)

Figure 4.23 is an extension of Figure 3.18 which now shows the principal type of actuators used in the substation diagram. The actuator node prototype for the circuit breaker developed in this research project is a process bus IED type with limited functionality. For the developed IED there is no support for hardwired connectivity and no mechanism to support manual operation mode. The developed work therefore focused on the communication aspects only.

Based on the above discussion it is clear that the developed actuator also known as a breaker IED can also be called an intelligent breaker controller. Therefore, every reference in this document to the term “breaker IED” or “intelligent circuit breaker controller” refers to the “actuator” concept.

Details on circuit breaker technology are presented in Section 4.4.1. The logical modelling of the prototype actuator is dealt with in Section 4.4.2.

4.4.3 Component hierarchy of different IEC 61850 views

The development of an IEC 61850 application or device starts with the process modelling. The concept of modelling includes all steps that contribute to the project definition and to its detailed specifications. This implies that the electromechanical characteristics including the behavioural response of the physical device to be modelled are well known in advance by the protection engineer. The IEC 61850 modelling for this research project can be segmented into three views which are:

- Application functional view related to the substation automation application, the application program and its subroutines (beyond the scope of the IEC 61850 series);
- Application data modelling or Device view (IEC 61850 series);
- Communication view (IEC 61850 series).

Figure 4.24 illustrates those views:

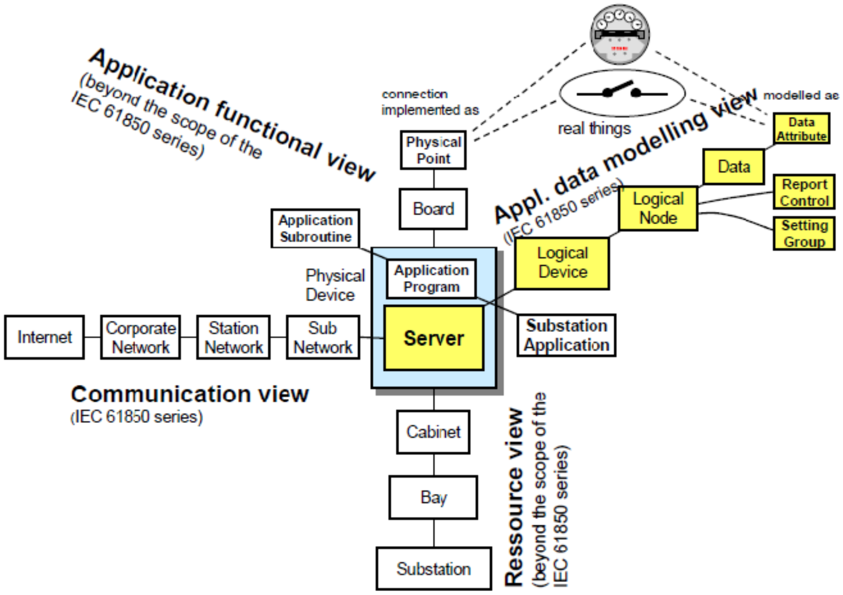


Figure 4.24: Component hierarchy of different views
(IEC 61850-7-1, 2003, p.58)

The data model and the communication parameters of an IEC 61850 application can be expressed without knowledge of the final hardware target. However, this is not the case when it comes to the software development of the application process. An IEC 61850 developer will need to know in the early stages of the project what the hardware platform of the application being developed is.

Section 4.4 provides a brief operational description of a circuit breaker. Section 4.4.4 presents the data model (modelling step 1) and the communication service supported (modelling step 2) by the intelligent circuit breaker node. Section 5.2.2addresses the software modelling component with the SC143 of the DK60 board as the final target.

The following section presents the particulars of each view[20] discussed above.

4.4.3.1 Application functional view

The application functional view in this work is limited to the modelling of different software functions for the intelligent circuit breaker node (section 5.2.2). The application functional view is related to the software application subroutines, application programme and to some extended to domain-specific substation application sub-routine function (e.g. Select-Before-Operate) (see Figure 4.24).

[20] The Resource view will not be examined because this falls out of the scope of the current project.

The application functional view is also related to the interaction that a program on a device A can have with another program on device B (IEC 61850, 2010), to achieve any substation function (e.g. distributed protection scheme).

The software modelling of an Intelligent Electronic Device to be used in the substation is not defined in the IEC 61850 standard. The data model and some state machine applicable to the communication stack are the only items defined in this view. Developers are at liberty to choose whichever method or technology is required. There is however a need to ensure that the data is communicated in the proper manner and that the devices provide sufficient functions and network services to ensure interoperability. Thus, the software modelling of an IEC 61850 device can be implemented using any method that the developer deems suitable for the project.

When the developer considers the detailed software modelling, particular attention needs to be paid to the hardware platform where the IEC 61850 application is installed. For instance, the developer will need to have a clear idea of the microprocessor in use. There is a requirement to determine how the physical input/output (I/O) of the plant is accessed and how to manage the communication on the data networking side.

Note that the Configured IED Description (CID) file will always be used to configure the communication on the server side and other parameters for the application programme.

Mapping of data to different OSI layers is the responsibility of the communication stack linked to the application program. The binding of data to physical input/output of the device is done via the CID file based on memory address location techniques (other implementations are possible).

4.4.3.2 Application data modelling view

The data model of an IEC 61850 device is obtained using a process known as virtualisation from the real world device into a ‘virtual’ environment where some hardware constraints are transformed into data objects (mandatory or optional data) and others are simply ignored.

There are two main steps to follow when building a data model of a device. The first step is the identification of all logical nodes, their data and data attributes. The second step is the construction of a logical device with all related controls blocks. The logical device including its logical node and data should be visible to the other devices on the network.

The data model is an integral part of the XML file called the IED Capability Descriptor (ICD). Figure 3.1 on page 44 illustrates the virtualisation process.

When performing the data model of a functional element in the substation, this functional element will not operate in isolation. The role of a system integration engineer is to complete the data model with an oriented view either in terms of Protection, Automation or Control model etc.

4.4.3.3 Communication view

The communication view defines the network structure in which the device is placed together with the other communication services. These definitions should be visible to the rest of the device connected to the network. Communication parameters are also available from the ICD file.

For a better understanding of the modelling process of the intelligent circuit breaker node, please review the operational description of a circuit breaker in section 4.4. For the current project, the focus has been placed on the GOOSE message communication aspects in the laboratory. A compatible 5 Volt latching relay is used as a circuit breaker to emulate different status position of the real substation circuit breaker. Figure 4.25 shows an abstract view of the communication flow in this research work.

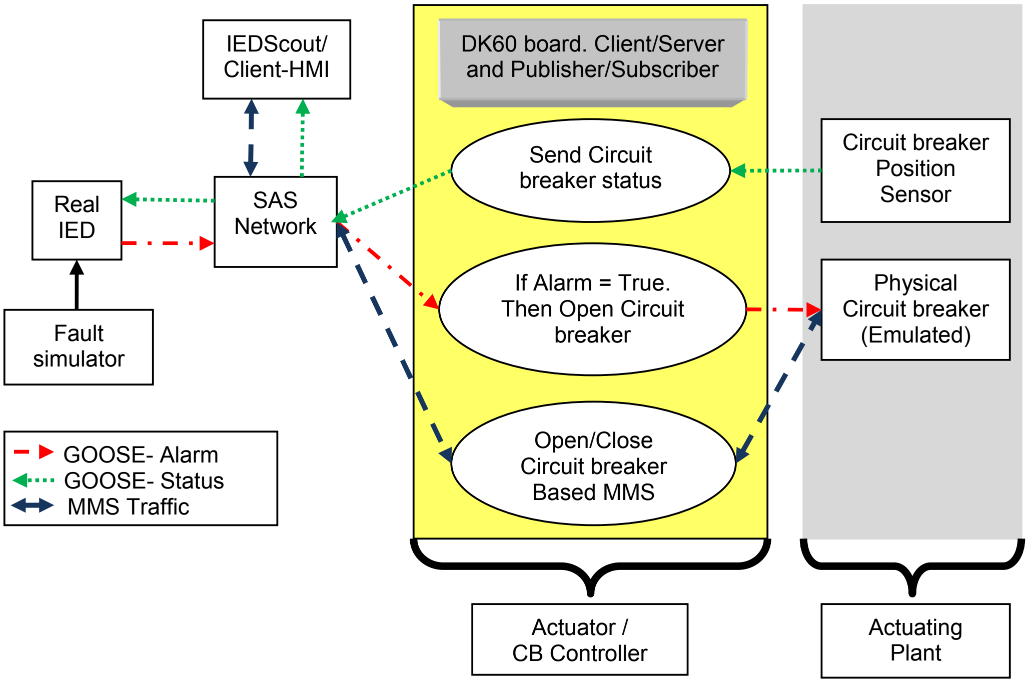

IEDScout/ Client-HMI
DK60 board. Client/Server and Publisher/Subscriber
Real IED
SAS Network
Send Circuit breaker status
Circuit breaker Position Sensor
If Alarm = True. Then Open Circuit breaker
Physical Circuit breaker (Emulated)
Fault simulator
Open/Close Circuit breaker Based MMS
GOOSE- Alarm
GOOSE- Status
MMS Traffic
Actuator / CB Controller
Actuating Plant

Figure 4.25: Communication scenario for thesis project

To develop a data model or to perform the detailed software design of a circuit breaker controller, it is important to first understand the operating mechanism of a power substation circuit breaker.

4.4.4 Intelligent circuit breaker node data modelling

With reference to Figure 4.22 (on page 99), the data model of an intelligent circuit breaker node is implemented using to the IEC 61850 methodology. Due to the fact that a real substation circuit breaker will not being used, the virtualisation process as defined in the IEC 61850 standard takes only into account mandatory data in order to realize an intelligent circuit breaker controller model. The reader should be aware that the 61850 modelling process does not refer to mathematical modelling of the circuit breaker nor software modelling.

To start the data model of any functional element in the substation automation system, one needs to verify that the functional element has not been modelled already in terms a logical node. If the considered functional element has not been modelled already in the IEC 61850 series, then there are appropriate procedures specified within the standard to develop a new logical node model. The definition of new model is independent from the hardware platform, but attention has to be paid so that the model is consistent with the data templates currently in use in the ICD file (this procedure is out of the scope of the current research work).

For the case of the current thesis, there is already a logical node that represents the circuit breaker function. This logical node is known as XCBR and is defined in IEC 61850-7-4. There is no therefore no need to develop a new model from scratch. However, for the considered circuit breaker, one needs to include all mandatory data and pay particular attention to other optional data that will increase the correctness of the XCBR LN model. The complete control model is discussed later.

As indicated in Figure 4.26, the data modelling of the intelligent circuit breaker node is implemented in a reverse manner to that presented in sections 7-4 to 7-1 of the IEC 61850 standard.

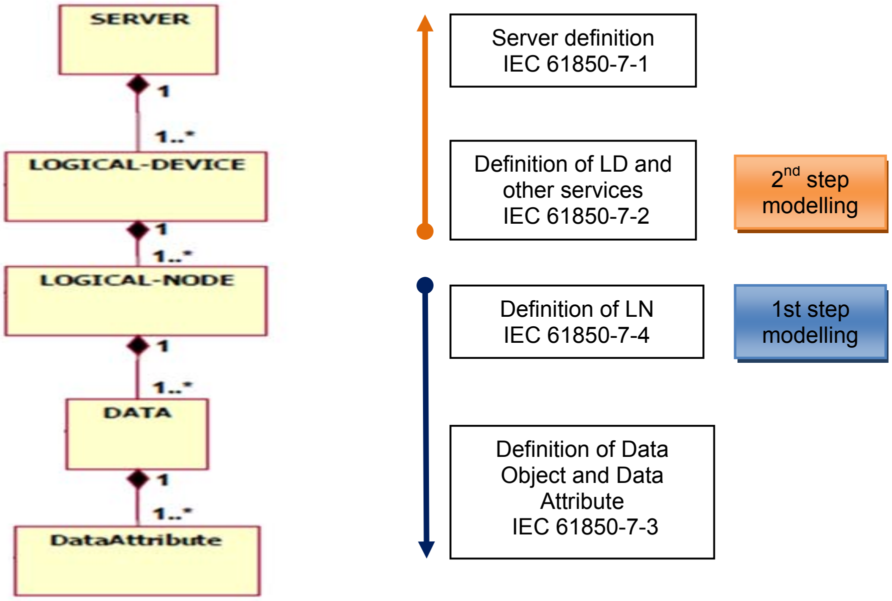

SERVER
1
1..*
LOGICAL-DEVICE
1
1..*
LOGICAL-NODE
1
1..*
DATA
1
1..*
DataAttribute
Server definition IEC 61850-7-1
Definition of LD and other services IEC 61850-7-2
2nd step modelling
Definition of LN IEC 61850-7-4
1st step modelling
Definition of Data Object and Data Attribute IEC 61850-7-3

Figure 4.26: Data Model steps

The following lines are an adaptation of the circuit breaker data modelling implemented in part 7 of the IEC 61850 standard.

4.4.4.1 First modelling step: Identification of logical node and Data

Figure 4.27 represents the logical node relationship. The logical node XCBR for the circuit breaker is the starting point of the modelling process if we ignore logical node zero (LLN0) and logical node physical device (LPHD) which are part of the second modelling step.

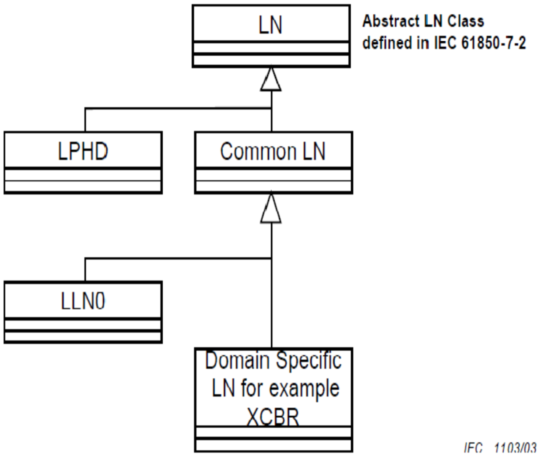


Figure 4.27: Logical Node relationships
(IEC 61850-7-4, 2003, p.17)

The basic building blocks of a logical node are depicted in Figure 4.28 below:

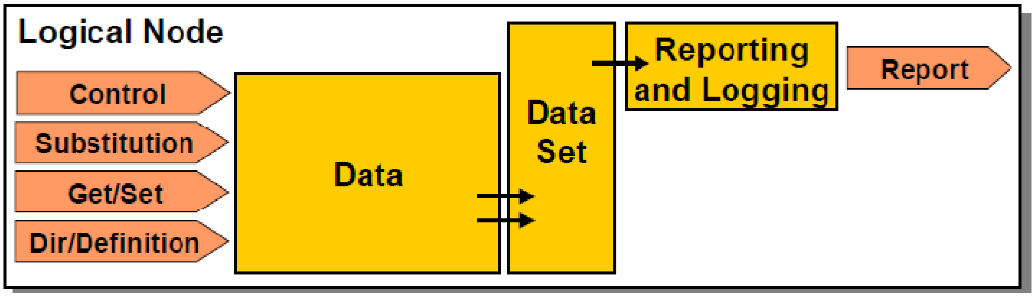
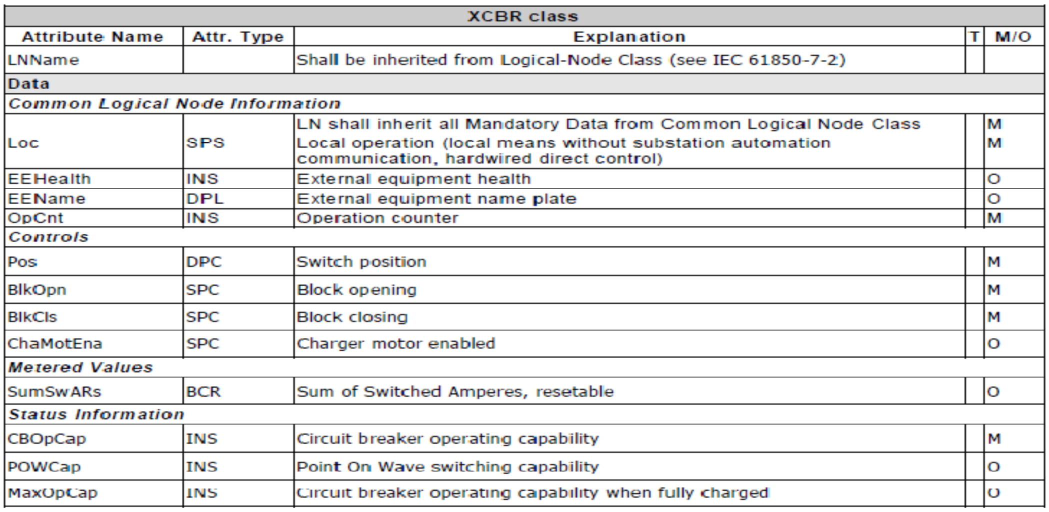


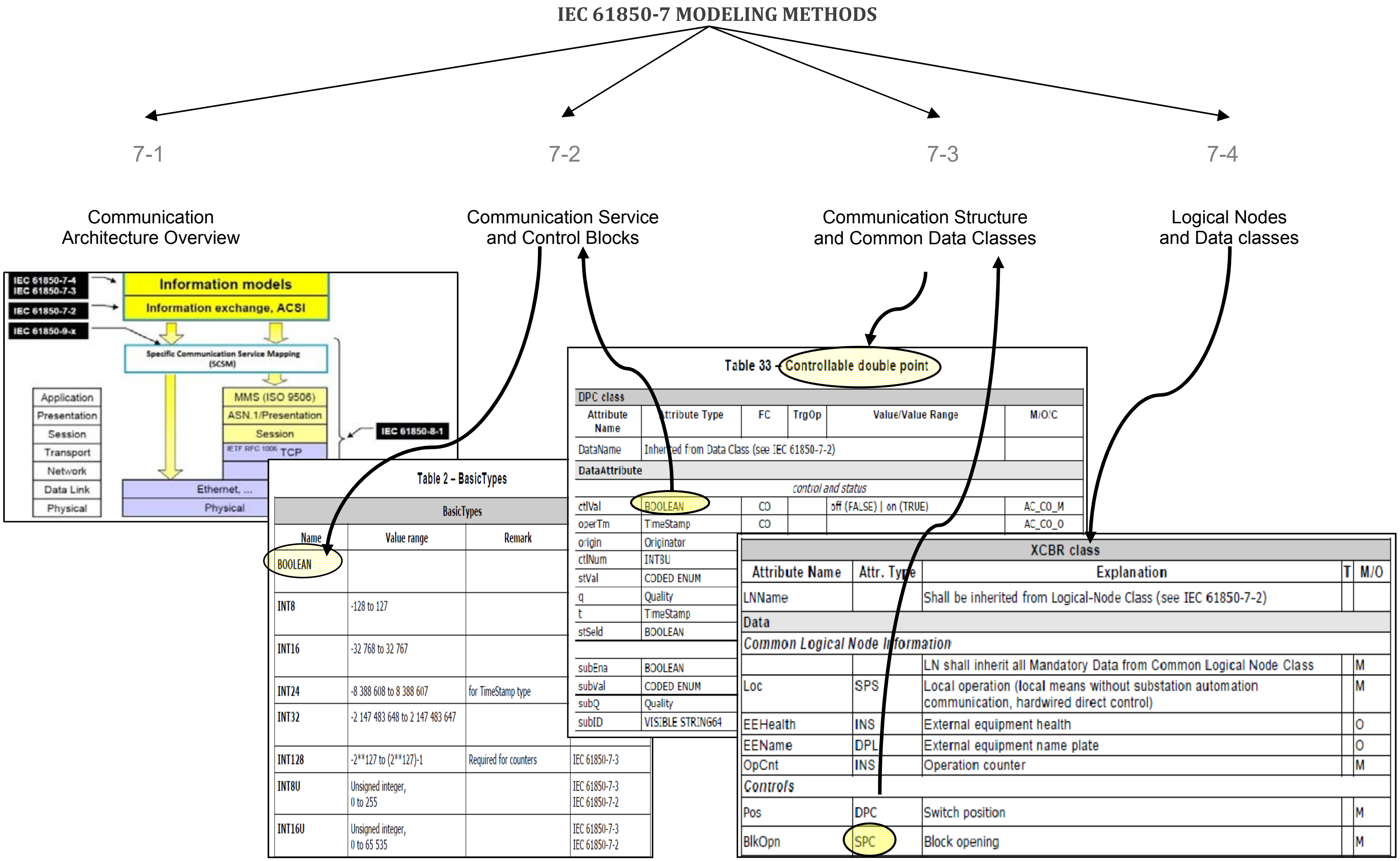
Figure 4.28: Basic building blocks for LN
(IEC 61850-7-1, 2003, p.45)

From the definition of the circuit breaker logical node XCBR in IEC 61850 section 7-4, Table 4.8 is presented:

Table 4.8: XCBR Class definition



XCBR class				
Attribute Name	**Attr. Type**	**Explanation**	**T**	**M/O**
LNName		Shall be inherited from Logical-Node Class (see IEC 61850-7-2)		
Data				
Common Logical Node Information				
		LN shall inherit all Mandatory Data from Common Logical Node Class		M
Loc	SPS	Local operation (local means without substation automation communication, hardwired direct control)		M
EEHealth	INS	External equipment health		O
EEName	DPL	External equipment name plate		O
OpCnt	INS	Operation counter		M
Controls				
Pos	DPC	Switch position		M
BlkOpn	SPC	Block opening		M
BlkCls	SPC	Block closing		M
ChaMotEna	SPC	Charger motor enabled		O
Metered Values				
SumSwARs	BCR	Sum of Switched Amperes, resetable		O
Status Information				
CBOpCap	INS	Circuit breaker operating capability		M
POWCap	INS	Point On Wave switching capability		O
MaxOpCap	INS	Circuit breaker operating capability when fully charged		O



IEC 61850-7 MODELING METHODS

7-1

Communication
Architecture Overview

7-2

Communication Service
and Control Blocks

7-3

Communication Structure
and Common Data Classes

7-4

Logical Nodes
and Data classes

Table 2 – BasicTypes

BasicTypes			
Name	Value range	Remark	
BOOLEAN			
INT8	-128 to 127		
INT16	-32 768 to 32 767		
INT24	-8 388 608 to 8 388 607	for TimeStamp type	
INT32	-2 147 483 648 to 2 147 483 647		
INT128	-2**127 to (2**127)-1	Required for counters	IEC 61850-7-3
INT8U	Unsigned integer, 0 to 255		IEC 61850-7-3 IEC 61850-7-2
INT16U	Unsigned integer, 0 to 65 535		IEC 61850-7-3 IEC 61850-7-2

Table 33 – Controllable double point

DPC class					
Attribute Name	Attribute Type	FC	TrgOp	Value/Value Range	M/O/C
DataName	Inherited from Data Class (see IEC 61850-7-2)				
DataAttribute					
				control and status	
ctlVal	BOOLEAN	CO		off (FALSE) | on (TRUE)	AC_CO_M
operTm	TimeStamp	CO			AC_CO_O
origin	Originator				
ctlNum	INT8U				
stVal	CODED ENUM				
q	Quality				
t	TimeStamp				
stSeld	BOOLEAN				
subEna	BOOLEAN				
subVal	CODED ENUM				
subQ	Quality				
subID	VISIBLE STRING64				

XCBR class				
Attribute Name	Attr. Type	Explanation	T	M/O
LNName		Shall be inherited from Logical-Node Class (see IEC 61850-7-2)		
Data				
Common Logical Node Information				
		LN shall inherit all Mandatory Data from Common Logical Node Class		M
Loc	SPS	Local operation (local means without substation automation communication, hardwired direct control)		M
EEHealth	INS	External equipment health		O
EEName	DPL	External equipment name plate		O
OpCnt	INS	Operation counter		M
Controls				
Pos	DPC	Switch position		M
BlkOpn	SPC	Block opening		M

Please find the complete IED Capability Description file for the current project in APPENDIX C .

4.4.4.2 Second modelling step: definition of the logical device model

The logical device is a composition of logical nodes and other services (GOOSE, Setting Group …). The grouping of logical nodes within logical devices is based on common features of these logical nodes as related to substation functions (IEC 61850, 2010). Therefore, in ICDs of multifunctional IEDs there can be a logical device for protection, another for measurement, another for control, etc. Figure 4.29 represents the building blocks for the logical device.

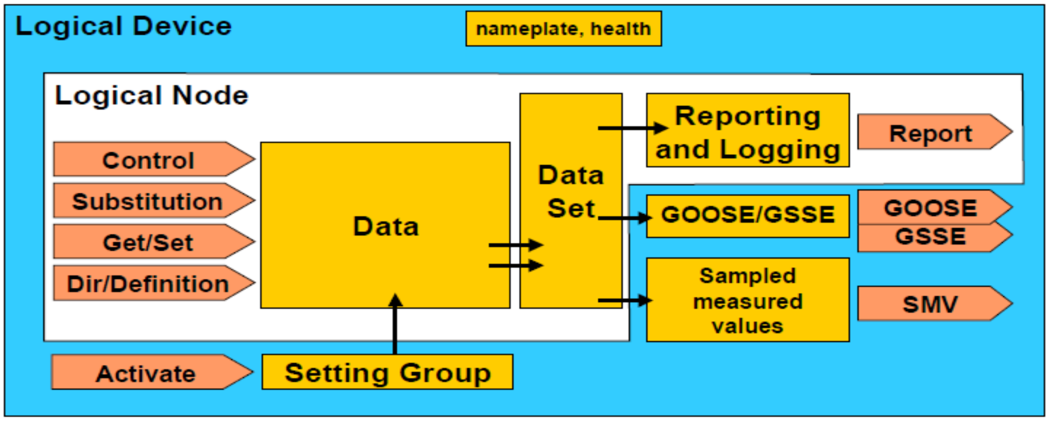


Figure 4.29 : Logical Device buiding block
(IEC 61850-7-1, 2003, p.47)

The logical node zero (LLN0) and the logical physical node (LPHD) are deployed within the logical device itself. It is through the logical node zero that communication services such as GOOSE are enabled. The logical node zero represents the common data of the logical device and the logical node physical device represents the common data of the physical device hosting the logical device (IEC 61850, 2010).

As illustrated in Figure 4.30, the logical device is then encapsulated in a server structure together with an additional function block (File transfer, time synchronisation and association).

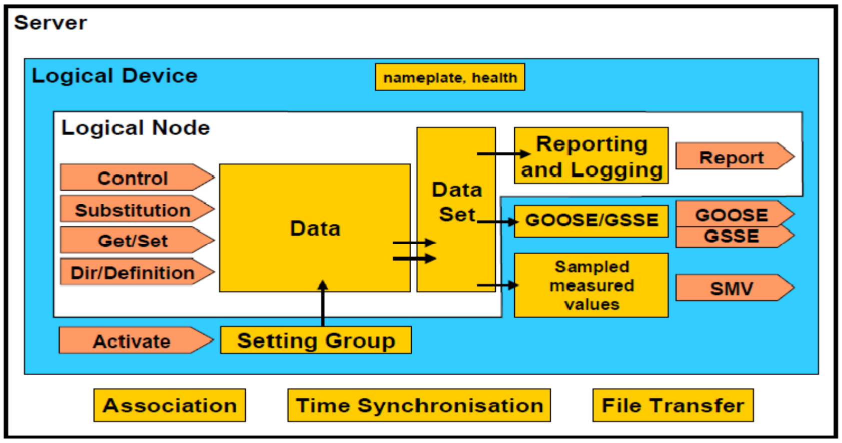


Figure 4.30: Server Logical Block
(IEC 61850-7-1, 2003, p.54)

In general, the basic data model is extended by the control model that will include additional logical nodes. In the case of a circuit breaker, the control model needs to include two additional logical nodes which are the switch control LN (CSWI) and the interlocking LN (CLIO). Figure 4.31 identifies additional logical nodes required for the control model to have a fully implemented circuit breaker control function according to common protection practice.

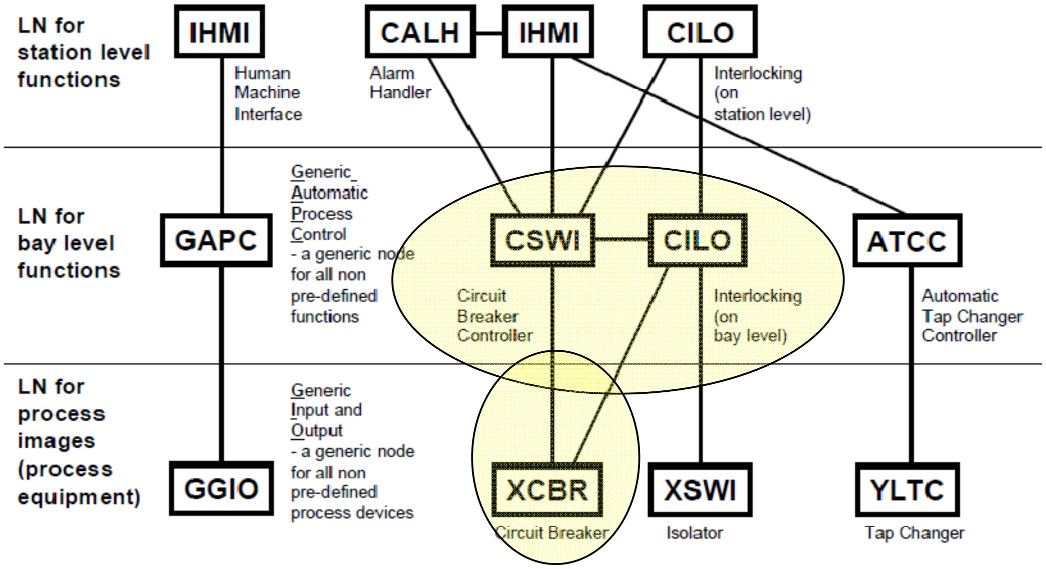

LN for station level functions
IHMI
Human Machine Interface
CALH
Alarm Handler
IHMI
CILO
Interlocking (on station level)
LN for bay level functions
GAPC
Generic Automatic Process Control - a generic node for all non pre-defined functions
CSWI
Circuit Breaker Controller
CILO
Interlocking (on bay level)
ATCC
Automatic Tap Changer Controller
LN for process images (process equipment)
GGIO
Generic Input and Output - a generic node for all non pre-defined process devices
XCBR
Circuit Breaker
XSWI
Isolator
YLTC
Tap Changer

Figure 4.31: Full Breaker control functions
(IEC 61850-5, 2003, p.38)

Assuming a control function that is not distributed, the logical device for a circuit breaker control model localised at the station bus level is composed of the following logical nodes:

- LLN0 (Logical node zero);
- LPHD1 (Physical device information);
- XCBR1 (Circuit breaker);
- CSWI1 (Switch controller); and
- CILO1 (Interlocking).

An optional Generic Input/ Output (GGIO) LN can be part of the model. The opening/closing control command of the XCBR LN shall be subscribed to from the CSWI LN as stated in the IEC61850 standard. The control of the circuit breaker from a remote location is based on client/server architecture and uses associated services based on MMS communication.

Since the current work is limited to the GOOSE communication aspects, the circuit breaker control model is reduced to its minimum model for demonstration of GOOSE communication. The XCBR LN is directly controlled from a remote Desktop without passing through the CSWI or CILO LN. Thus, the Select-Before-Operate control method will not be used. Using IEDScout software as a client HMI, the control method used in this project is known as the Direct with Normal Security method.

For a minimal implementation the intelligent circuit breaker controller consisting of the following Logical Nodes are part of a single control logical device:

- LLN0 (Logical node zero);
- LPHD1 (Physical device information);
- XCBR1 (Circuit breaker); and
- GGIO (Generic process I/O).

This minimal logical device composition has been acknowledged as sufficient for demonstration of the GOOSE message publishing mechanism and direct control to the circuit breaker position. In the following section, this logical device model is part of the DK60SV1.CID that is used for the server configuration. Figure 4.32 illustrates what is known in object-oriented terminology as “encapsulation” or the “container” principle. A review of some important IEC 61850 concepts might be necessary when it comes to modelling in section 3.3.

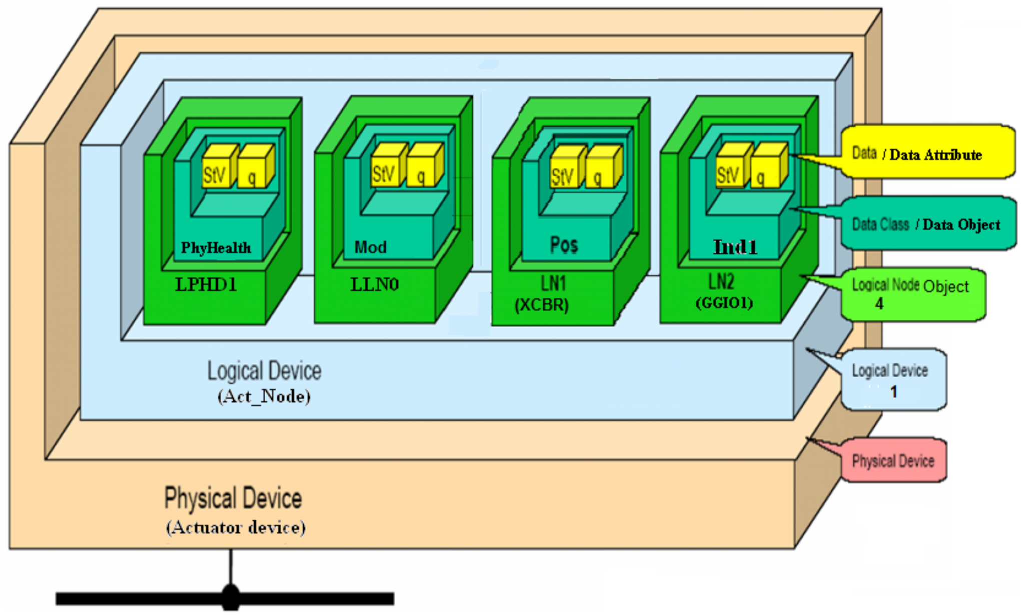


Figure 4.32: Encapsulation perspective for DK60SV1.CID

As stated in section 4.3.7.2, the subscription mechanism is associated with a client deployment from within the IED together with an imported and adapted Substation Configuration Description (SCD) file. Additional details with regards to the subscription model (DK60CL1.SCD) and its mechanism is discussed in section 5.2.2.

Figure 4.33 provides an overview of the correlation of block data inside a CID file.

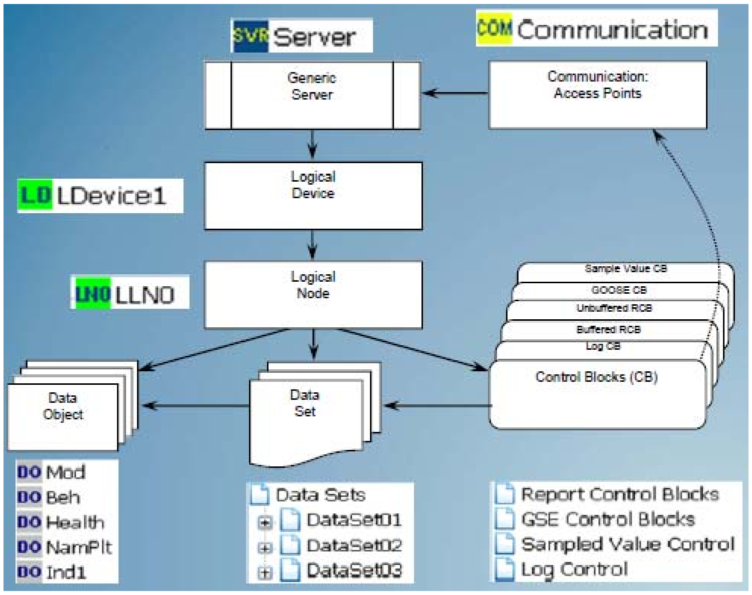


Figure 4.33: Overview of Block Data in a CID file
(SystemCorp_a, 2011)

4.4.4.3 DK60SV1.CID file creation for Server configuration

Figure 4.34 is an example of Circuit breaker control function using a direct access with normal security method (IEDScout software from OMICRON can be used as a universal client HMI). The DK60SV1.ICD file has first to be designed on paper and should also include the position of the breaker that is embedded within the GOOSE message and published on the network.

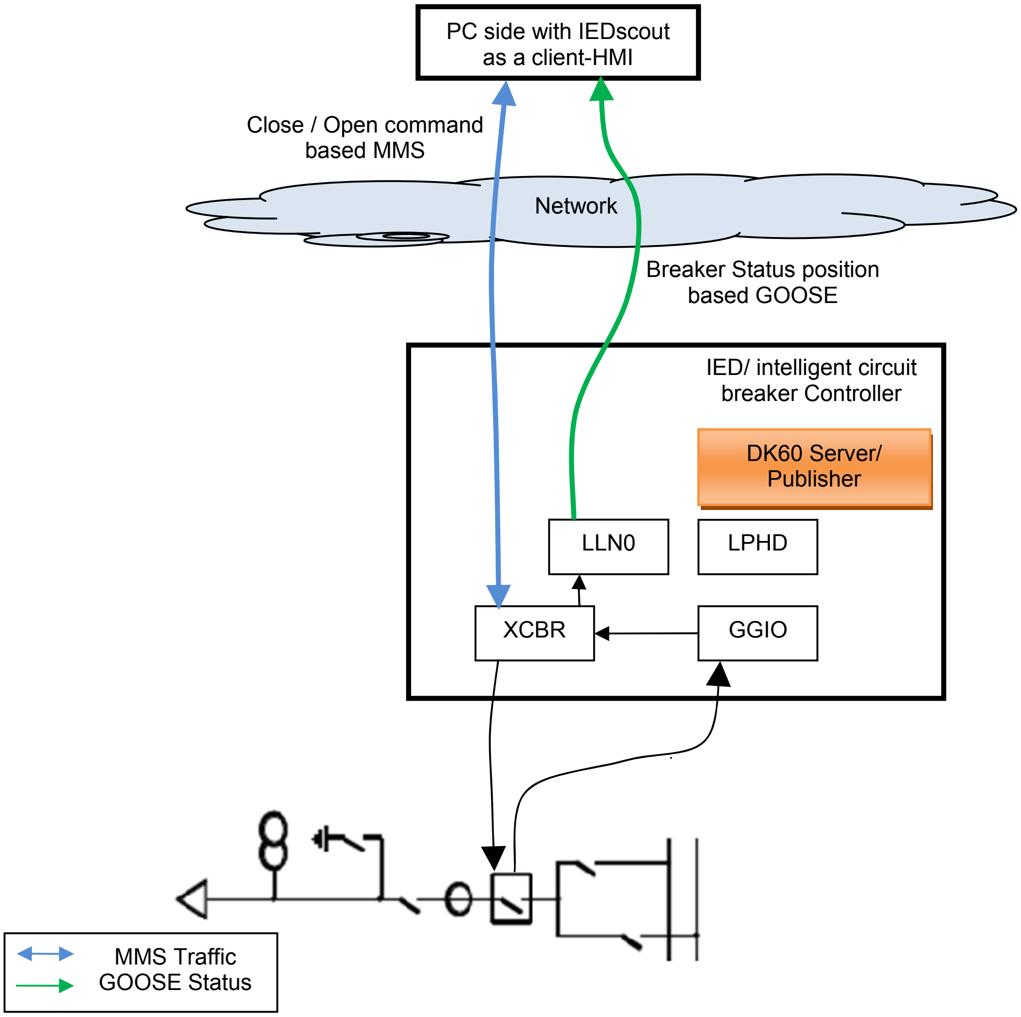

PC side with IEDscout as a client-HMI
Close / Open command based MMS
Network
Breaker Status position based GOOSE
IED/ intelligent circuit breaker Controller
DK60 Server/ Publisher
LLN0
LPHD
XCBR
GGIO
MMS Traffic
GOOSE Status

Figure 4.34: DK60SV1.ICD functions overview

Please refer to the ICD Designer manual on how to develop custom ICD files (Section 4.3.4).

Figure 4.35 displays different details of a CID file (Communication Block section and IED block section). The following SCL section composition conforms to that specified by the IEC 61850 standard and reported in Figure 3.23: UML diagram overview of SCL schema, page 67.

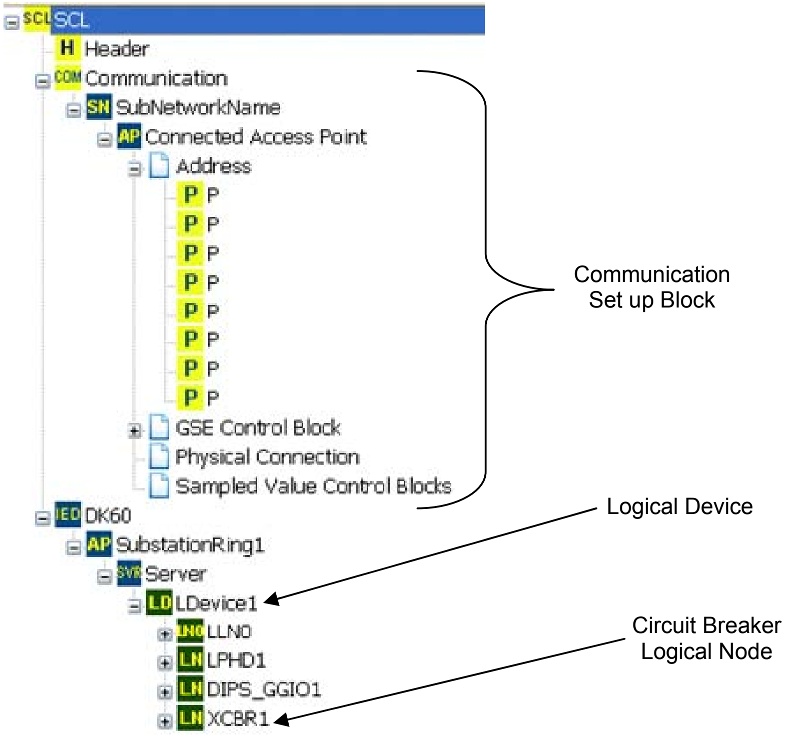
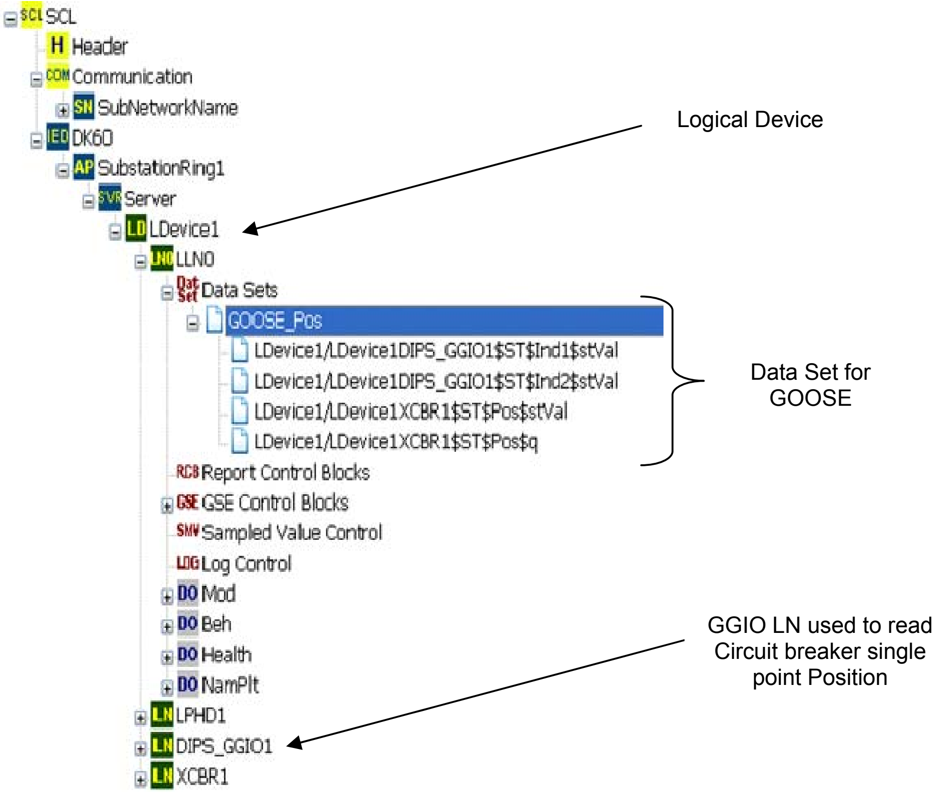


Figure 4.35: Partial view of DK60SV1.CID file

The data template section is not accessible from within ICD Designer. For the development of a new logical node, XML Marker or the Notepad application within Windows may be used to modify the data template according what is required.

Please note that it is advisable to make a backup copy of the data template before introducing any changes. If the data template is inconsistent after changes have been brought about, ICD Designer upon execution displays an error message. The data template used by ICD Designer is located in:
C:\Program Files\ICD Designer\Templates\IEC61850 for the full version.

Figure 4.36 presents details of the composition of the IED section.



Logical Device

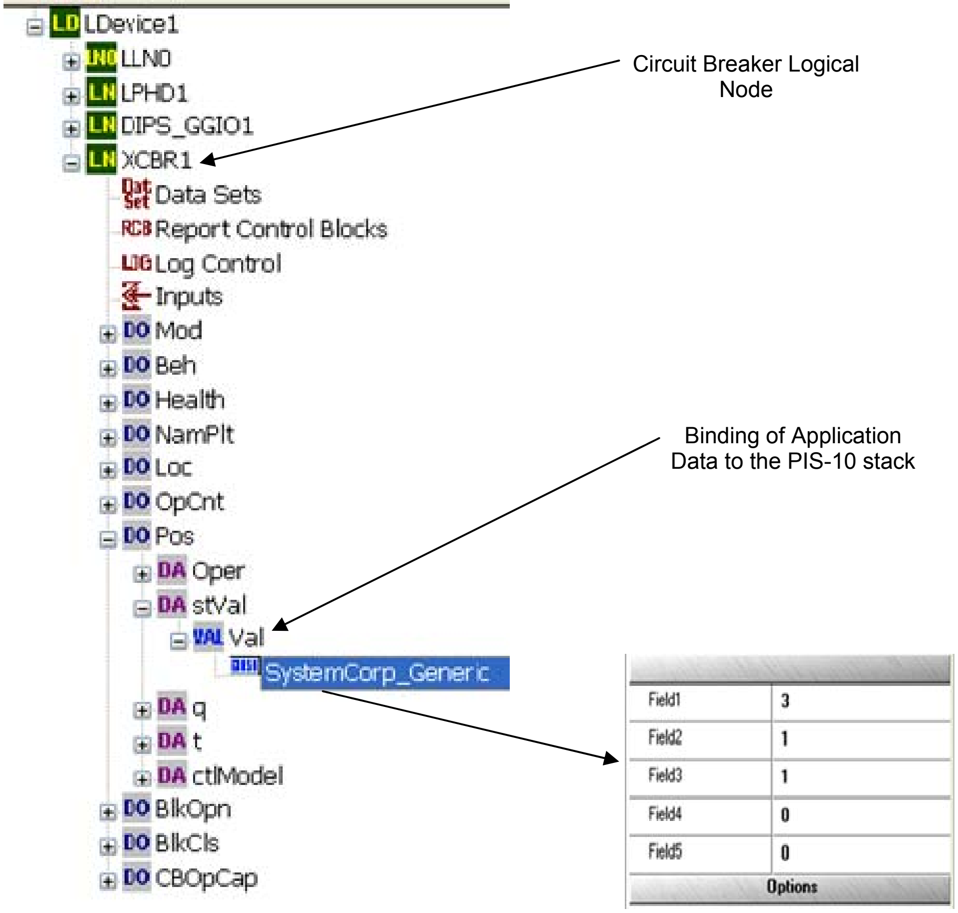
Data Set for GOOSE

GGIO LN used to read Circuit breaker single point Position

Figure 4.36: Details composition of IED section of the DK60SV1.CID

Please refer to Appendix C for a review of the complete DK60SV1.CID file.

Figure 4.37 illustrates the use of Private Generic Value (GPV) as per PIS-010 implementation.



Circuit Breaker Logical Node

Binding of Application Data to the PIS-10 stack

Field1	3
Field2	1
Field3	1
Field4	0
Field5	0

Figure 4.37: Private Generic Value for Data binding to PIS-10 stack

The GPV are used to map user application data to the PIS-010 communication stack. GPVs field are also implicitly used in the user application local database function. The binding of user application data to the PIS-010 and to physical input/output of the DK60 expansion pin header S10 respectively, is discussed in section 5.2.

4.4.4.4 DK60Cl1.SCD file creation for Client configuration

As per the PIS-010 implementation of IEC 61850 functions, the subscription to GOOSE message of a field device is implemented through the deployment of a client module configured with a Substation Configuration Description (SCD) file (DK60CL1.SCD in this document).Please, refer to the ICD Designer manual to discover how to develop a custom SCL file (Section 4.3.4).

Figure 4.38 illustrates the internal organisation and communication flow based on the DK60SV1.CID and DK60CL1.SCD for the current research project.

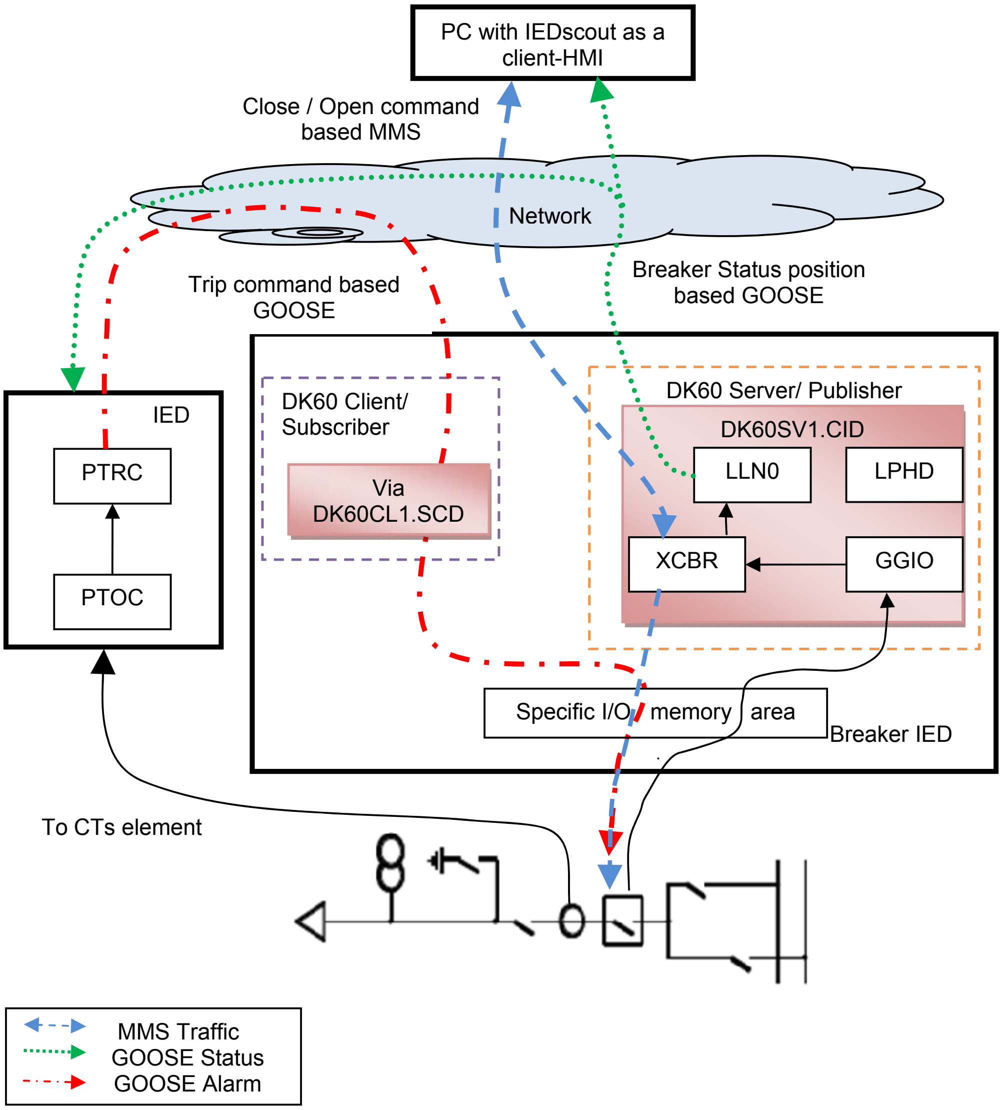


Figure 4.38: DK60CL1SCD and DK60SV1.ICD functions overview

GOOSE message are used by the substation protection function as a fast alarm communication system to generally issue a trip command to a circuit breaker. GOOSE messages can also be used to indicate issues within the communication system. These indications might be circuit breaker status position or any data included in the dataset associated with the GOOSE control block.

Figure 4.39 presents a diagram that shows the steps to be followed in order to develop a client Substation Configuration Description (SCD) file that will work with the PIS-10 stack such as the DK60CL1.scd.

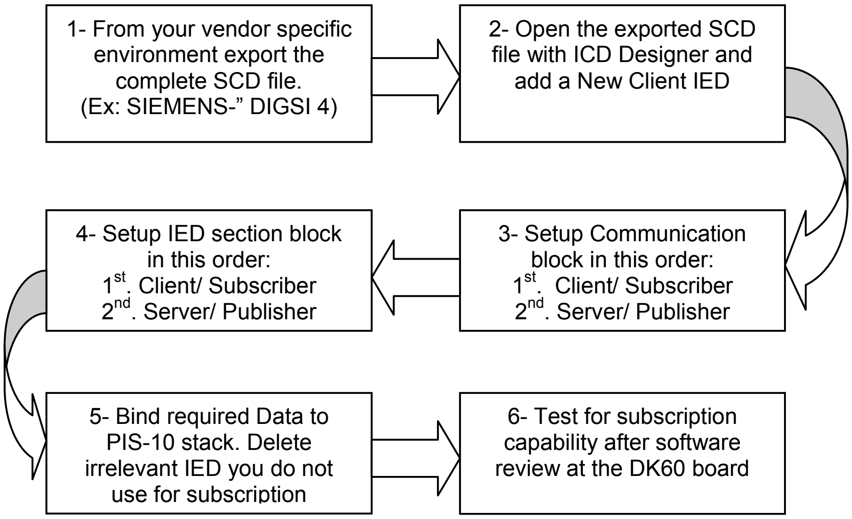
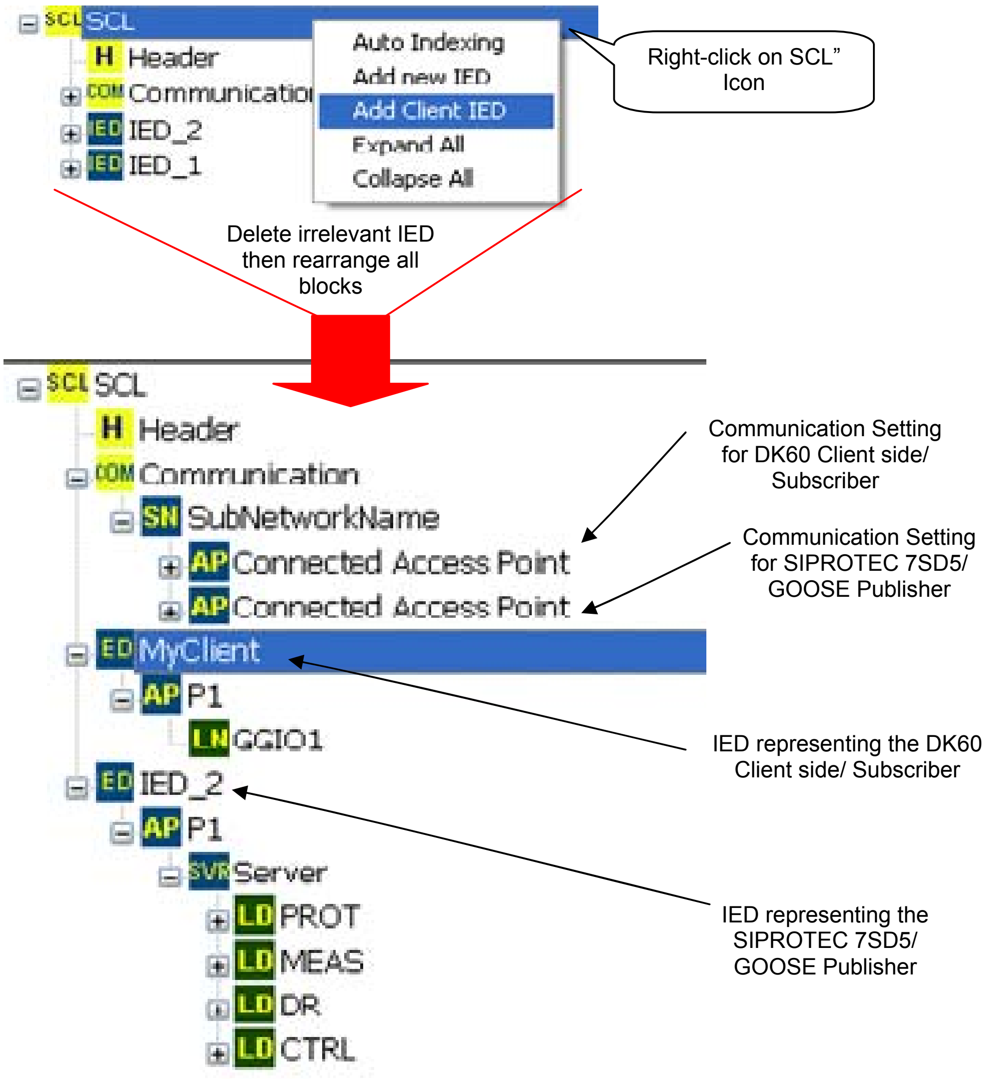

1- From your vendor specific environment export the complete SCD file. (Ex: SIEMENS-" DIGSI 4)
2- Open the exported SCD file with ICD Designer and add a New Client IED
3- Setup Communication block in this order: 1st. Client/ Subscriber 2nd. Server/ Publisher
4- Setup IED section block in this order: 1st. Client/ Subscriber 2nd. Server/ Publisher
5- Bind required Data to PIS-10 stack. Delete irrelevant IED you do not use for subscription
6- Test for subscription capability after software review at the DK60 board

Figure 4.39: Development steps for compatible PIS-10 stack SCD file

Please follow these steps strictly; the SystemCorp team may be contacted for additional support if particular problem are encountered.

Figure 4.40 shows how to add a New Client IED section for use with the DK60 board and the final view of the DK60CL1.SCD file after rearrangement of the communication section block and the IED section block.



Right-click on SCL" Icon

Delete irrelevant IED then rearrange all blocks

Communication Setting for DK60 Client side/ Subscriber

Communication Setting for SIPROTEC 7SD5/ GOOSE Publisher

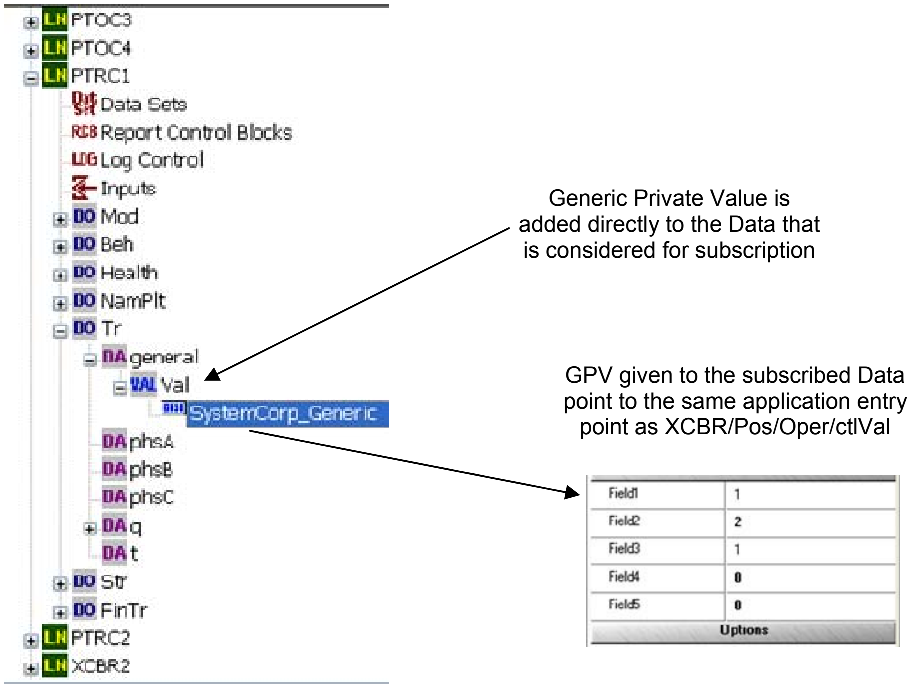
IED representing the DK60 Client side/ Subscriber

IED representing the SIPROTEC 7SD5/ GOOSE Publisher

Figure 4.40: Details composition of the DK60CL1.SCD

Note that any IED that is not required to implement the subscription needs can be deleted from the design. It is recommended to not redefine the SCL "Data Template". This section of the SCL file is normally not visible within ICD-Designer to avoid any structural damage.

Figure 4.41 shows where the data binding for subscription of an incoming GOOSE message has been made for the current project in order to trip the prototype circuit breaker.



Generic Private Value is added directly to the Data that is considered for subscription

GPV given to the subscribed Data point to the same application entry point as XCBR/Pos/Oper/ctlVal

Field1	1
Field2	2
Field3	1
Field4	0
Field5	0

Options

Figure 4.41: Subscription by means of GPV

Note that the current data binding (see Figure 4.41) for the circuit breaker trip is based on the General Trip signal alarm. The trip subscription is directed to the same memory area as the control data for circuit breaker positioning (XCBR/Pos/Oper/ctlVal).

4.5 Conclusion

Chapter 4 presents an IEC61850 solution that is used to develop an IEC61850 compliant device. It also presents preliminary work in terms of the data model according to the IEC 61850 project specifications.

The choice made with respect to the DK61 development environment to continue this research work is justified by the amount of system integration work (especially on the communication stack) that was required by the author before a fully functional IEC61850 application was operational.

Table 4.9 shows where the work done in this research projects fits in, and what studies still need to be performed for future project developments to ensure complete IEC 61850 compliant intelligent circuit breaker controller devices.

Table 4.9: Project complexity delimitation

Moral entity or Institutional body	Task	Required knowledge
SystemCorp	Communication stack development (PIS-10/ GOOSE, MMS …)	Embedded Network programming
	API for End-User software developer in order to add IEC 61850 communication capabilities to an application.	
Beck-IPC	Platform Target development and specific system integration on the SC 143 chip.	Embedded software development and hardware expertise.
IEC 61850 System Engineer	[**This is where the current research work focuses**]. IEC 61850 software application development, homogenous integration and interoperability achievement.	-Embedded software development for Real Time Systems - Understand substation principle for Protection, Control, Metering, Monitoring …
Related to the application of other substation standard	[**This is where the focus of future research work should be**]. Deployment of substation logic control automation and distribution at the software level. Final device build and verification for correctness of execution and compliance to the relevant standard	- Distributed system; - IEC 61499 and IEC61131-3 - Electromechanical - Electromagnetic interference - Heat transfer, etc.

Any further developments on this research work are based on the DK61 development kit and its embedded SC143 chip.

CHAPTER FIVE:

CASE STUDY IMPLEMENTATION: SOFTWARE DEVELOPMENT, INTERFACING AND SYSTEM INTEGRATION

5.1 Introduction

The sections in this chapter detail the following:

Section 5.1 - the Introduction (current section). Section 5.2 - the Software Development steps leading to the creation of the GA61850.exe (process application software) which is IEC 61850 compliant. This is not to be confused with the communication software stack (PIS-10, already provided by SystemCorp). Particular details with regard to the software planning and design are given in Section 5.2.2. Details on Data Object mapping procedure from the IEC 61850 communication stack (PIS-10) to the process application (GA61850.exe), also from the process application to the I/O unit and vice-versa are also described in the same Section. More details with respect to specific Database structure are given in Section 0. Section 5.3 - the system Hardware Integration between the emulated circuit breaker and its inter-action with the process application which was developed on the embedded system platform.

5.2 Software development

The term “software” is used to define a collection of computer programs and related data that provides the instructions in order for a computer to know what to do and how to do it. In other words, software is the realization of mathematical or logic functions as procedures. These functions map a body of input data into body of output data (Edward, 2002).

In addition to the above general description of the term “software”, the term “Embedded software” refers to the realization of mathematical or logic functions as procedural behaviour that uses some input parameter/s to produce output control signal/s. Those output signals affect directly the state of some hardware component. Therefore, the principal role of “Embedded software” is the interaction with the physical world (Edward, 2002).

Embedded systems react in real-time with the surrounding peripherals and environment. For the case of this project, this ability is facilitated through the use of a

Real-Time Operating System (RTOS). In general, embedded systems utilize peripherals interacting around a central processing unit (microprocessor). In the case of the current thesis, the SC143 embedded web microcontroller (from Beck IPC) has its processing unit together with its main peripherals integrated in a single-chip solution.

For the current project it is to be assumed that the software developed falls into the definition of “embedded software” stated above.

[Note: Section 3.6.3 has given a description of a compact approach which may be used for the software and product development life cycle. One can decide to choose another software life cycle or product development methodology.]

With reference to the note above: Figure 5.1 shows the proposed project evolution flow (Software/ Hardware sequence co-development strategy) which starts by requiring a clear identification of the project needs and specification(Step A). It is practical to develop different use-case diagrams (UML based) that focuses on different aspects of the project (Step B). Different IEC 61850 Data and service models (Protection, Control, Monitoring...) have to be clearly understood, decided upon, and set down on paper (Step C); before one can start the development of any type of System Configuration file - SCL file (ICD, CID, SCD…) (Step E). It is only when the Data model construction of all required SCL files has been completed, that you may proceed to the Software Design stage (Step F).

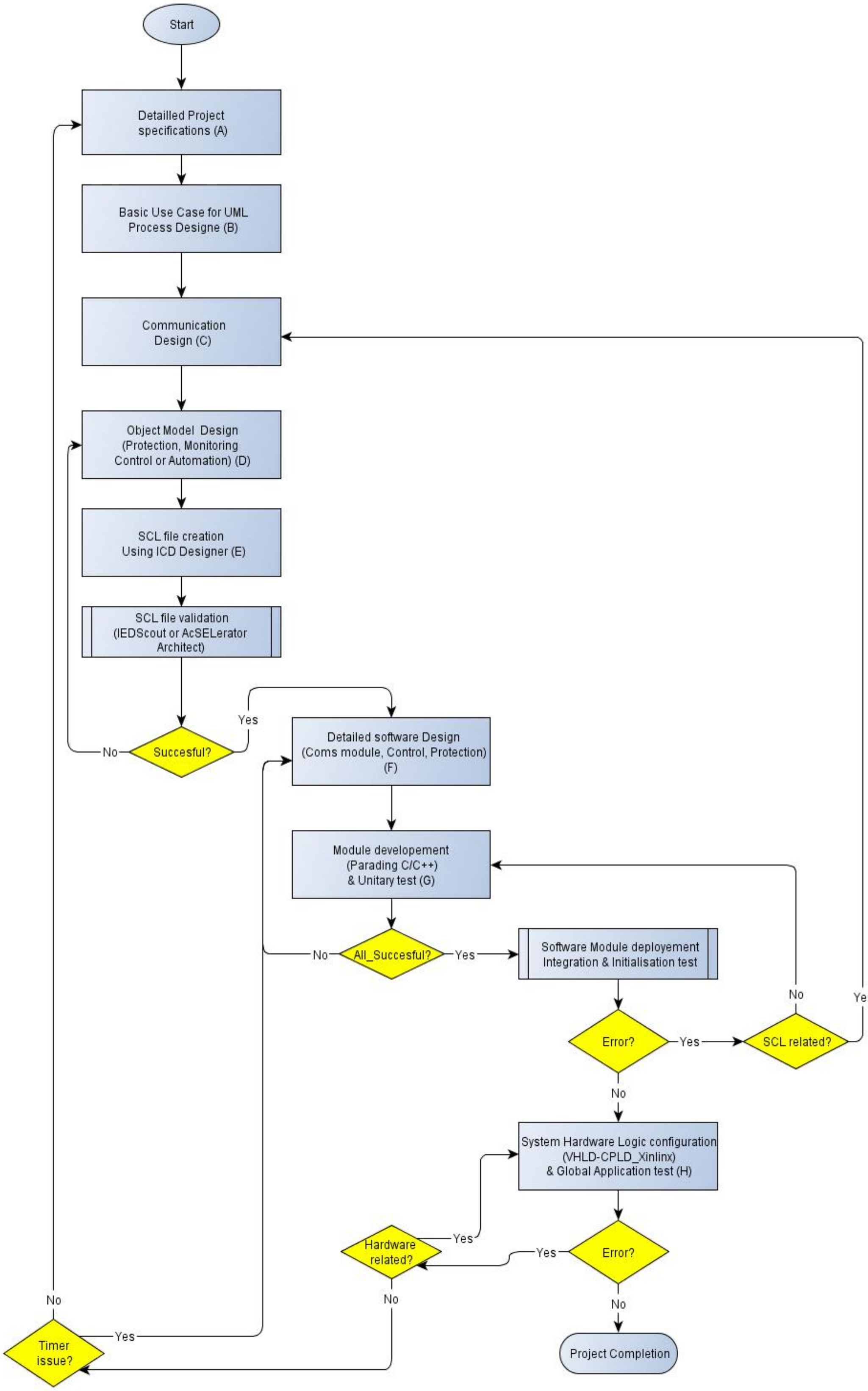

Start
Detailled Project specifications (A)
Basic Use Case for UML Process Designe (B)
Communication Design (C)
Object Model Design (Protection, Monitoring Control or Automation) (D)
SCL file creation Using ICD Designer (E)
SCL file validation (IEDScout or AcSELerator Architect)
Succesful?
No
Yes
Detailed software Design (Coms module, Control, Protection) (F)
Module developement (Parading C/C++) & Unitary test (G)
All_Succesful?
No
Yes
Software Module deployement Integration & Initialisation test
Error?
Yes
SCL related?
No
Yes
No
System Hardware Logic configuration (VHLD-CPLD_Xinlinx) & Global Application test (H)
Error?
Yes
Hardware related?
Yes
No
No
Project Completion
Timer issue?
Yes
No

Figure 5.1: Software/ Hardware sequence co-development strategy

In order to develop properly executing/ functioning, and robust software, different verification steps (not described in this thesis) have to be taken according to software standards (See Section 3.6.3). The software developer needs to commit to incorporating “Error handling” strategies to avoid application crashes or unreliable software. When the quality of the software operation and reliability is regarded as sufficient, one can move forward to the hardware development stage (Step H: Figure 5.1). This stage includes the adaptation of some of the CPLD code in order to establish a hardware link to specific I/O ports. You also need to re-evaluate your timing system (software Timers and hardware delay) more than once before being able to reach sufficient stability in your system.

Figure 5.2 shows the sequence of the development (and/or adaptation) of the software, and the file types and file names that arise during the system integration process.

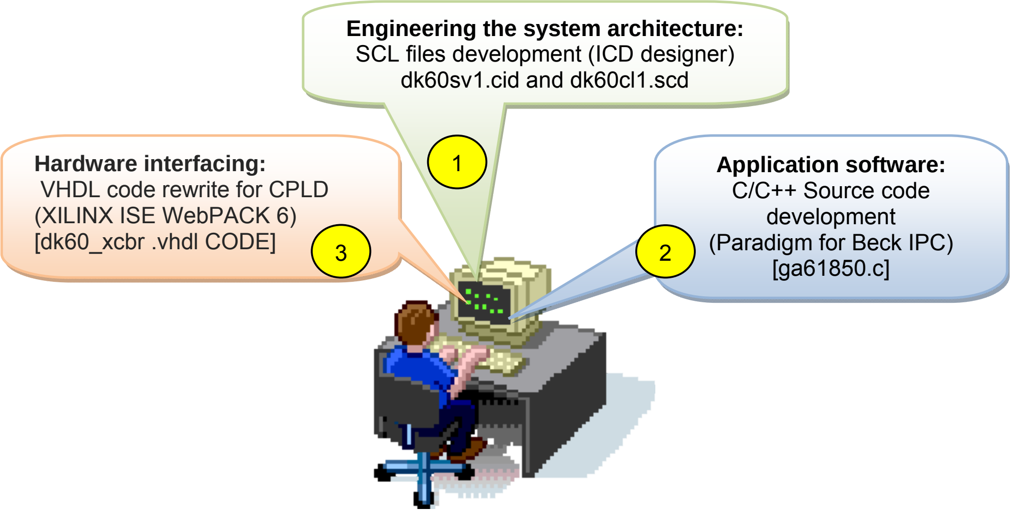

Engineering the system architecture:
SCL files development (ICD designer)
dk60sv1.cid and dk60cl1.scd
1
Hardware interfacing:
VHDL code rewrite for CPLD
(XILINX ISE WebPACK 6)
[dk60_xcbr .vhdl CODE]
3
2
Application software:
C/C++ Source code
development
(Paradigm for Beck IPC)
[ga61850.c]

Figure 5.2: System development workflow

Explanation of, or development of the software is discussed below in the sequence indicated in Figure 5.2 above.

5.2.1 Engineering the system architecture

For this project, the SCL file development has been dealt with in Section 4.4.4.3 for the DK60SV1.ICD (Server configuration) and at Section 4.4.4.4 for the DK60CL1.SCD (Client configuration).

5.2.2 Application software: planning and design

Before the software design stage, one needs to have a minimum knowledge of Beck specific APIs and a good knowledge of the IEC 61850 API from SystemCorp (see section 4.3.6). The developer needs to identify clearly what functionalities are implemented within the domain of the IEC 61850 stack, and what other functionalities he/she would have to write while developing their own source code. Please refer to the online KEMA certificate of the PIS-10 stack at: http://www.systemcorp.com.au/images/stories/manuals/KEMA_Certified_61850_Stack_Functions.pdf for more information about the capabilities and limitations of the PIS-10 IEC 61850 stack.

On examining the internal configuration of the PIS-10 IEC 61850 stack (version 1.36.), the developer realises that the Publisher and Subscriber mechanism for GOOSE is considered to be an added feature to the Client/Server functionality (see section 3.3.7). The Publisher role is executed through utilization of the IEC 61850 Server mechanism, while the Subscriber role is performed by an IEC 61850 Client (see section 4.3.7). Server and Client roles have been implemented independently from one another. This means that both objects can exist on the same hardware platform and also within the same executable programme.

The PIS-010 (v1.36) communication stack is event driven. The PIS-010 interacts with the software application by means of function calls and call backs. It also interacts with the Data-Link layer of the OSI model via the Ethernet Packet driver provided by “Beck IPC”. On reception of a GOOSE message (Ethernet based), a hardware interrupt (Int 0xC2) is issued to the PS-010 stack by the packet driver. In turn, a call-back function (interrupt) is issued by the PIS-010 stack to the software application for it to process the new GOOSE message.

Figure 5.3 below presents the project software design procedure.

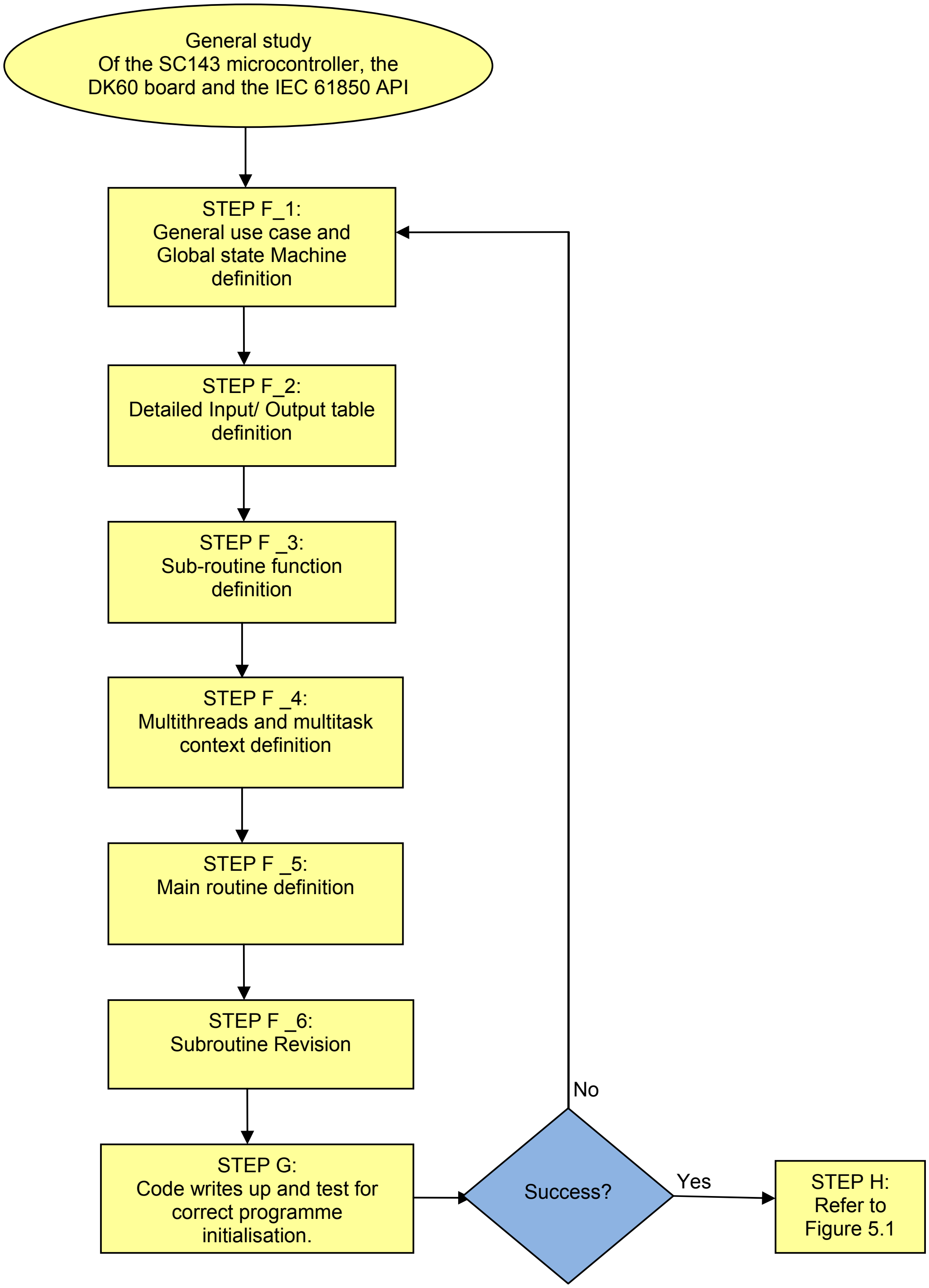


Figure 5.3: Project software design procedure

Note: The Design sub-stages specific to the Software Planning and Design depicted in Step F of Figure 5.1, are outlined in more detail in Figure 5.3 (Steps F_1 to F_6) above, and expanded upon by explanation below.

- **STEP F_1: General use case and Global state Machine definition**

A software architecture design starts by the identification at a high level of abstraction of what task the developed software is intended to accomplish, and what the resources are (e.g. Input\Output interfaces etc) that the software needs to use in order to complete a given task. Figure 4.25 gives the use case definition that summarises what the developed software is intended to do.

Without giving much attention to the communication aspect of this thesis, Figure 5.4 shows the basic state machine for the proposed "Intelligent Circuit Breaker Controller". This state machine has been implemented within the software source code developed in this thesis. The transition (condition) label "Pb_Open" and "Pb_Close" stands respectively for "Push button open command" and "Push button close command" (see APPENDIX F and APPENDIX H).

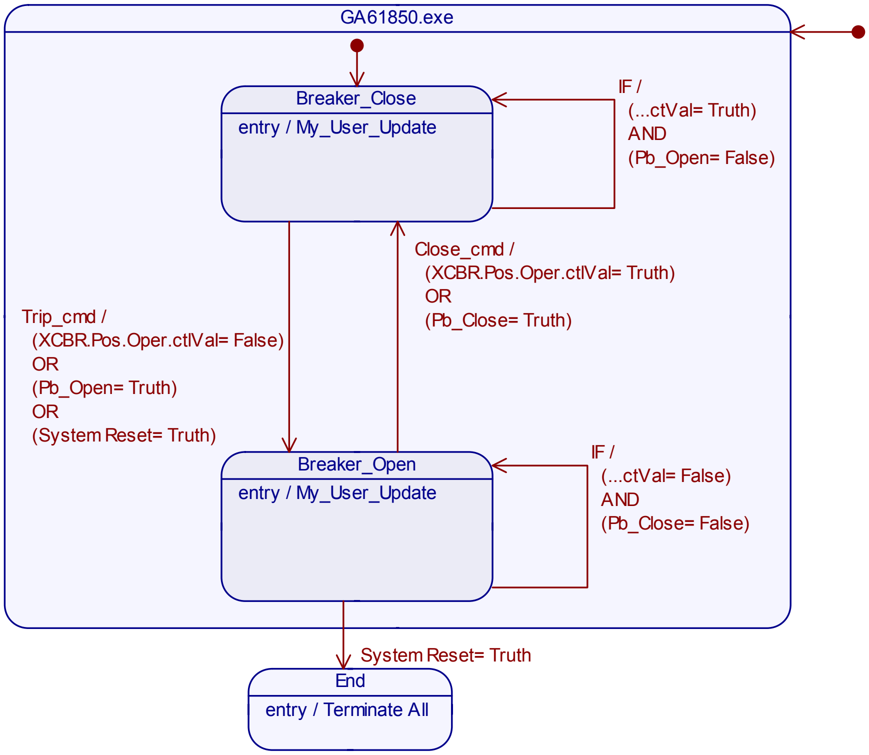


Figure 5.4: Process application state machine for breaker IED

After examination of the IEC 61850 API, it becomes evident that the software application for the current research thesis needs to be a combination of Server functionality as well as Client. In other words, the developed software application is composed of two IEC 61850 object modules (Server and Client). Note that Server and Client have been arranged to operate independently from one another.

Figure 5.5 below details the State Machine for the Server module.

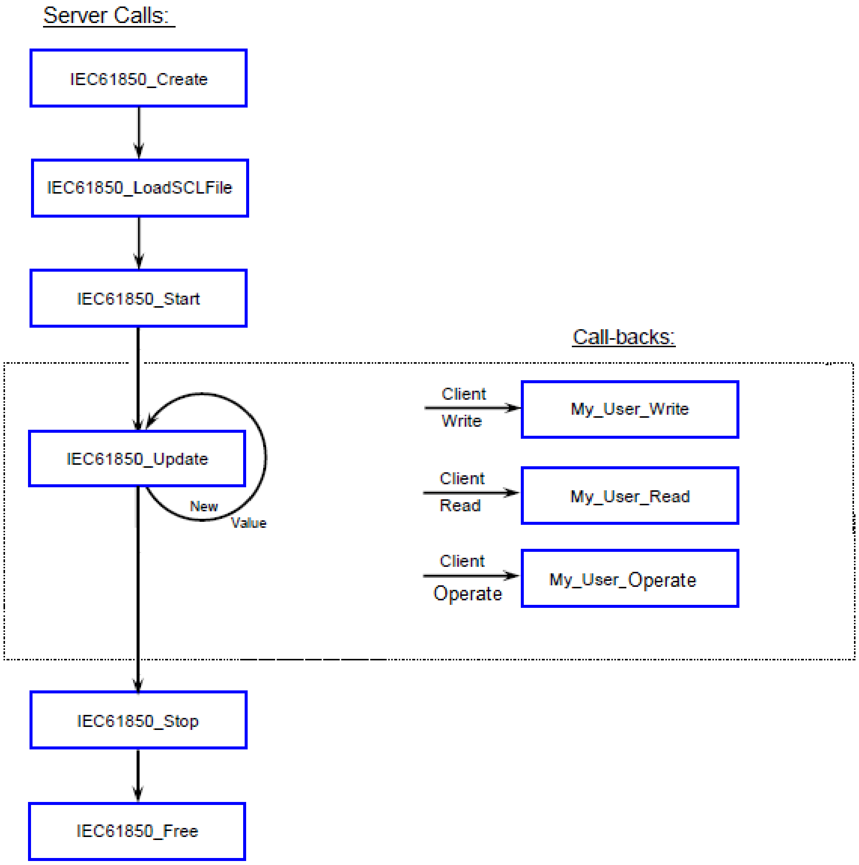


Figure 5.5: Server State Machine
Adapted from (SystemCorp_b, 2010)

The basic Read and Write function for the server is used together with the Operate function.

Figure 5.6 below details the State Machine for the Client module.

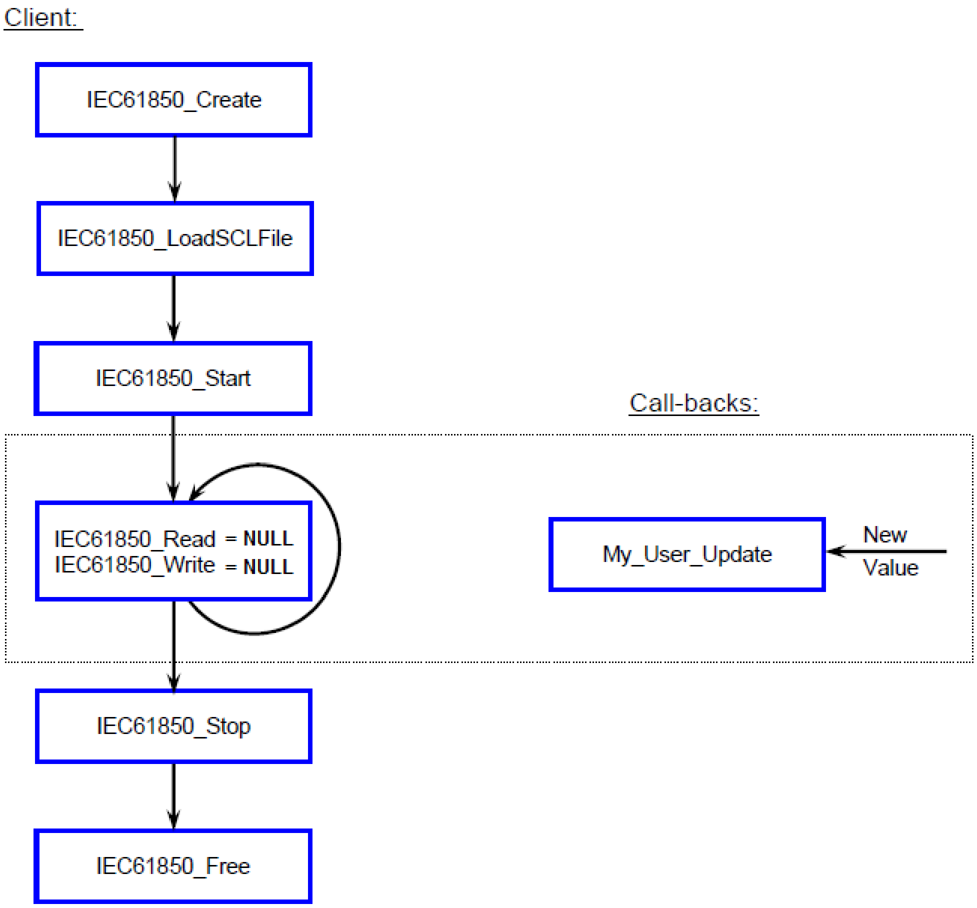


Figure 5.6: Client state machine
Adapted from (SystemCorp_b, 2010)

The basic Client Read and Write function parameters are set to “Null” as only the GOOSE subscription mechanism is required in this case.

- **STEP F _2: Detailed Input/ Output table definition**

In this step, the system engineer needs to define the input/output system. Figure 5.7 shows an overview of the hardware interconnection system. The complete definition of input/output system needs to take into account the Data mapping from the IEC 61850 communication stack to the software application local database. But also from the entry point in the local database to a particular I/O (physical point).

The binary information carried within the payload of the transmitted or received IEC 61850 GOOSE message has to be mapped to the physical pins connecting the DK 60 Development Board to the Emulated Circuit Breaker – as shown in Figure 5.7 below.

The binary information - before being inserted into, or stripped from the Ethernet Frame, resides within a dedicated memory location within the SC143. This memory location is referred to as the GPV

The virtual input/output within the IEC61850 environment needs to be mapped to the real physical input/output via the CPLD. A VHDL file has been adapted for this purpose. (See APPENDIX E)

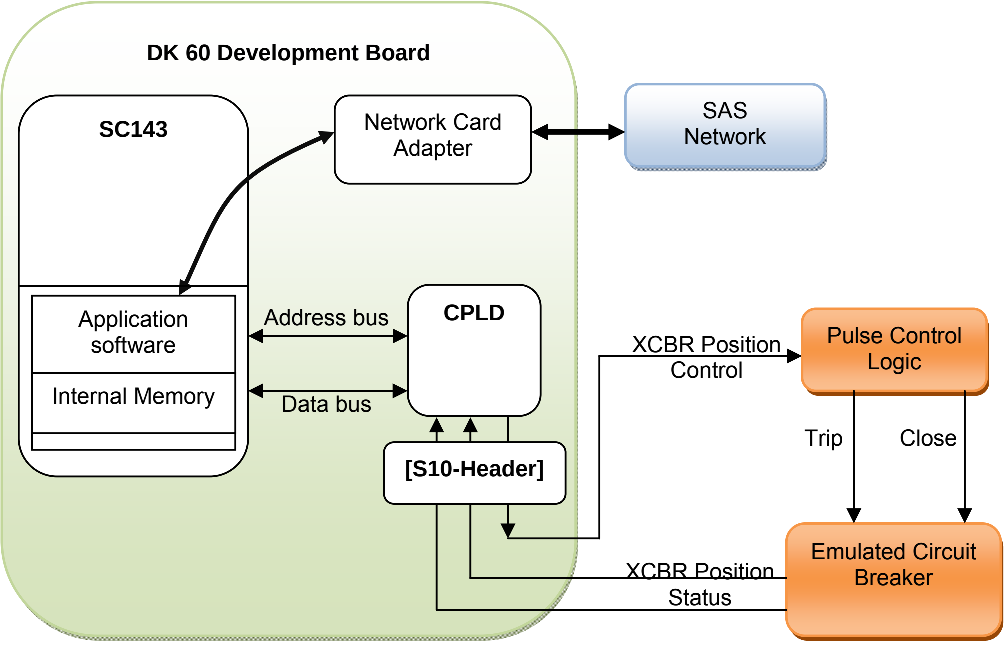


Figure 5.7: hardware connectivity between DK60 board and Emulated Circuit Breaker

The Generic Private Value implemented by “SystemCorp” in their IEC61850 schema allows the System Engineer to assign an address location (Data ID) to a data point in the SCL configuration file. This allows the System Engineer to access that data point via the User application. Data point (or entry point) just defines the specific location in the memory of the CPU. Also one needs to distinguish between the binding of the Application Data Object to the PIS-10 stack and the binding of the physical input to the Application Data Object.

The Local Database is created and initialised with the value {0} from within the software application code. The Local Database is configured with the appropriate value found in the corresponding SCL file (DK60SV1.CID or DK60CL1.SCD). For the current project, the SC143 input/output address Location starts at **0xC00**. The input/output start address refers to a predefined memory location within the SC143 which corresponds to the Dip switches and the LEDs on the DK 60 board.

The PIS-10 stack uses 5 fields to implement the Generic Private Value data binding (Field 4 and Field 5 are not used). Their description is as depicted in Figure 5.8.

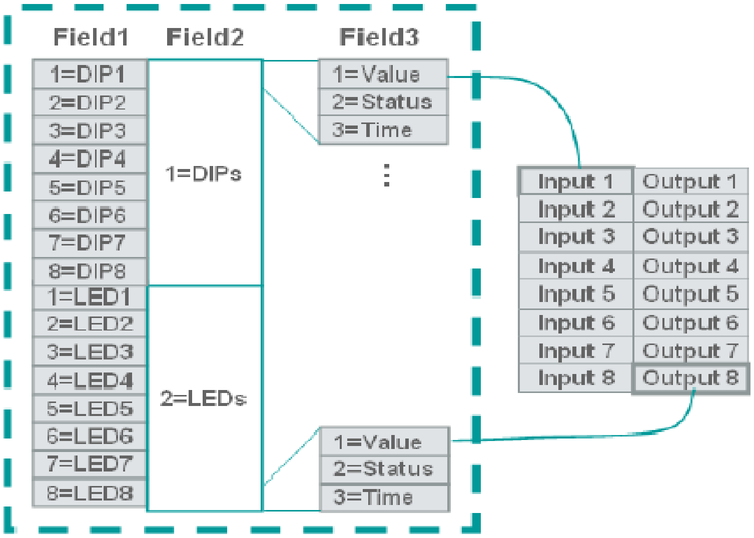


Figure 5.8: GPV field description
(SystemCorp_a, 2011)

As one can see on the picture above, "Field1" is used to identify the Object number. "Field2" identifies the object categories (Input or Output) and "Field3" defines the object parameter to look for (Value; Quality or Time). For more information relating to the PIS-10 Data binding, refer to the "**Getting Started IPC@CHIP Embedded Web Controller Family IEC 61850 Basics**" document. This document is available at: http://www.systemcorp.com.au/images/stories/manuals/Getting_Started_DK61_IEC61850.pdf

Table 5.1 presents the input/output table matrix. The Generic Private Value (GPV) as per “SystemCorp” mapping is shown for each data set. As stated previously, the GPV fields 4 and 5 are not used in this thesis. They are assumed to be equal to zero {0}. Table 5.1 serves as a guide-line for data binding between application data and the IEC 61850 communication stack (PIS-10). The table also serves as a reference for the implementation of the logical binding between the Complex Programmable logic Device (CPLD) extension pin header [S110] on the DK60 board and the corresponding entry point in the SC143 internal memory.

Table 5.1: Input/Output matrix

System Input From Circuit Breaker Prototype	Unused_CPLD Pin header	System Output to Local Circuit control (Close, Trip)	Unused_CPLD Pin header
IN_1: 1st position bit of XCBR/Pos/stVal. IN_GGIO1/Ind1/stVal **GPV** { Field1 = 1 Field2 = 1 Field3 = [1 to 3]	CPLD_1 [Real Input]	Active in the current project OUT_1: XCBR/Pos/Oper/ctlVal **GPV** { Field1 = 1 Field2 = 2 Field3 = 1	CPLD_3 [Real Output]
IN_2: 2nd position bit of XCBR/Pos/stVal. IN_GGIO1/Ind2/stVal **GPV** { Field1 = 2 Field2 = 1 Field3 = [1 to 3]	CPLD_2 [Real Input]	Not used. Reserved for future development Ex: to block ‘Open operation’ OUT_2: XCBR/**BlkOpn**/Oper/ctlVal **GPV** { Field1 = 2 Field2 = 2 Field3 = 1	Not used CPLD_4 [Real Output]

IN_3 (XCBR/Pos/stVal) as a double point position status. Container is built in software.
GPV { Field1 = 3
Field2 = 1
Field3 = [1 to 3]
	NA
	[Recombined Input]
		Not used. Reserved for future development.
		Ex: to block ‘Close operation’
		OUT_3:
		XCBR/**BlkCls**/Oper/ctlVal
		GPV { Field1 = 3
		Field2 = 2
		Field3 = 1
			Not used
			CPLD_5
			[Real Output]

For this project, interlocking Data-Object “BlkOpn” and “BlkCls” (highlighted in bold font on Table 5.2) are part of the modelling as they are specified as mandatory Data-Objects within IEC61850 - but, they are not routed to the “Logic Control” stage. They shall be routed to the “Logic Control” stage in future projects where there might be a particular interest on interlocking functions. The “local database” implemented within the software application, is structured to accommodate this Look-Up Table. The next section gives more details on how the local database is structured and how the communication flow sequence arises.

- **STEP F _3: Function description**

Development of an IEC61850 application requires the integration of key functions into the user/developer application. On the DK61 Development Kit some of the functions reside within the PIS-010 stack, and the user/developer is allowed access to them through the function interfaces – some of which have been opened to the developer. Even given that the functions are available for use, identification or determination of which sub-routines to include in the code might be difficult at first if the developer is not familiar with IEC 61850 application development.

Table 5.2 below presents the list of functions/methods used in this project - with a short description of their role.

Table 5.2: Function list identification

Function/ method name	**Description**	**Use for Server module**	**Use for Client module**
PIS-010 Object Management- From SystemCorp			
IEC61850_Create	create a client/server object	☑	☑
IEC61850_LoadSCLfile	Configures the client/server from an SCL file	☑	☑
IEC61850_Start	Starts the services (e.g. GOOSE, MMS etc) within the client/server object.	☑	☑
IEC61850_Stop	Stop the services (e.g. GOOSE, MMS etc) within the client/server Object.	☑	☑
IEC61850_Free	Delete a client/server object	☑	☑
PIS-10 Data Attribute Access.- Adapted from SystemCorp			
IEC61850_Update	Server Function called from the user application indicating a change in the hardware to the IEC 61850 stack.	☑	□
My_User_Write [MyWriteFunction]	Function calls backs by the PIS-10 to handle a Write command coming from a client on the Network. Changes a value on the Server	☑	□
My_User_Read [MyReadFunction]	Function calls backs by the PIS-10 to handle a Read command coming from a Client on the network. Requests a value FROM the Server	☑	□
My_User_Operate [MyOperateFunction]	Function calls backs by the PIS-10 to handle an Operate command on selected control/s with specified value	☑	□

Function/ method name	Description	Use for Server module	Use for Client module
My_User_Update [MyUpdateFunction]	Function calls backs by the PIS-10 to inform the user application of a change of a value on a published GOOSE issue by a Server on the network	□	☑
PIS-10 Time Function From SystemCorp			
getTime	get the NTP time	NA	NA
PIS-10 Support functions From SystemCorp			
IEC61850_GetLibraryVersion	Returns the version of the PIS-10 library	NA	NA
IEC61850_GetLibraryBuildTime	Returns build time for PIS-10 Library		
Other important function or command (Non exhaustive) – From Beck IPC and SystemCorp			
CreateLocalDatabase	Create the local data base that holds the Application Object	NA	NA
huge InputHandler	This is the function that is executed continuously. Other function are event driven	NA	NA
RTX_Create_Task	This function is used to create an executable task (RTOS task).	NA	NA
RTX_Sleep_Time	This function is used to put the execution of the current program on hold. This is very important in multitasking environment.	NA	NA
outportb	Use to Set the Output port	NA	NA
inportb	Use to read the Input port	NA	NA

For more information relating to the arguments passed to, and returned from the PIS-10 functions, please refer to the online API manual at:
http://www.systemcorp.com.au/PIS10API/index.html

Any other PIS-10 related functions/ methods that apply to Server or Client that are not listed in this table, are to be considered as "Null" for the current project- (i.e. IEC61850_Read = NULL, for a Client) (See APPENDIX F: GA61850 Application source code at page 273).

- **STEP F _4: Multitasking and Multithreading context definition**

According to BECK-IPC (2011), "A task provides a thread of execution. Each task has its own context, including an instruction pointer and program stack. Each task in the @Chip-RTOS has a unique priority of execution, providing a pre-emptive form of multitasking".

Considering that the communication stack used to accommodate the Server/publisher and the Client/Subscriber (PIS-10) is event driven and no protection algorithm or monitoring function (etc) have been implemented as part of the actuator node itself.

There is no multitasking operation required for normal execution of the Server and Client module - but if one would like to create an application that executes both the Server and Client module in a concurrent manner, it is recommended to read the BECK-IPC online documentation for multitasking available at:
http://www.beck-ipc.com/files/api/scxxx/multitasking.htm

[Note: Development of a concurrent system in the case of the current project is not critical]. One can develop a multitasking/multithreading application using the Beck RTOS @ Chip if doing so increases the stability, reliability and overall performance of the system.

Thus the developed software (GA61850.C) executes only a single task (InHandlerTask). The RTOS command used to create and start the task is as follow:

```
/* Create and Start IO Handler Task */
/* with tasked = -1 */
error = RTX_Create_Task(&taskID , &InHandlerTask);
if(error != 0)
{
printf("\r\n IO Handler task Create Failed : %i", error);
}
```

For more detail please consult the software code in APPENDIX F].

- **STEP F _5: Main routine definition**

The function “main” is used to create and initialise the IEC 61850 object (Server and Client). It is also from the function “main” that the “**huge InputHandler**” function is put to action as a RTOS task. Figure 5.9 below was developed to describe (a) the software internal initialisation stage (executed once), (b) the active stage and (c) the termination stage. Adapted call-back functions such as “My_User_Write” are **interrupt-driven** (based on specific communication event). Some of the important functions have been previously described (Cf. Table 5.2: Function list identification).

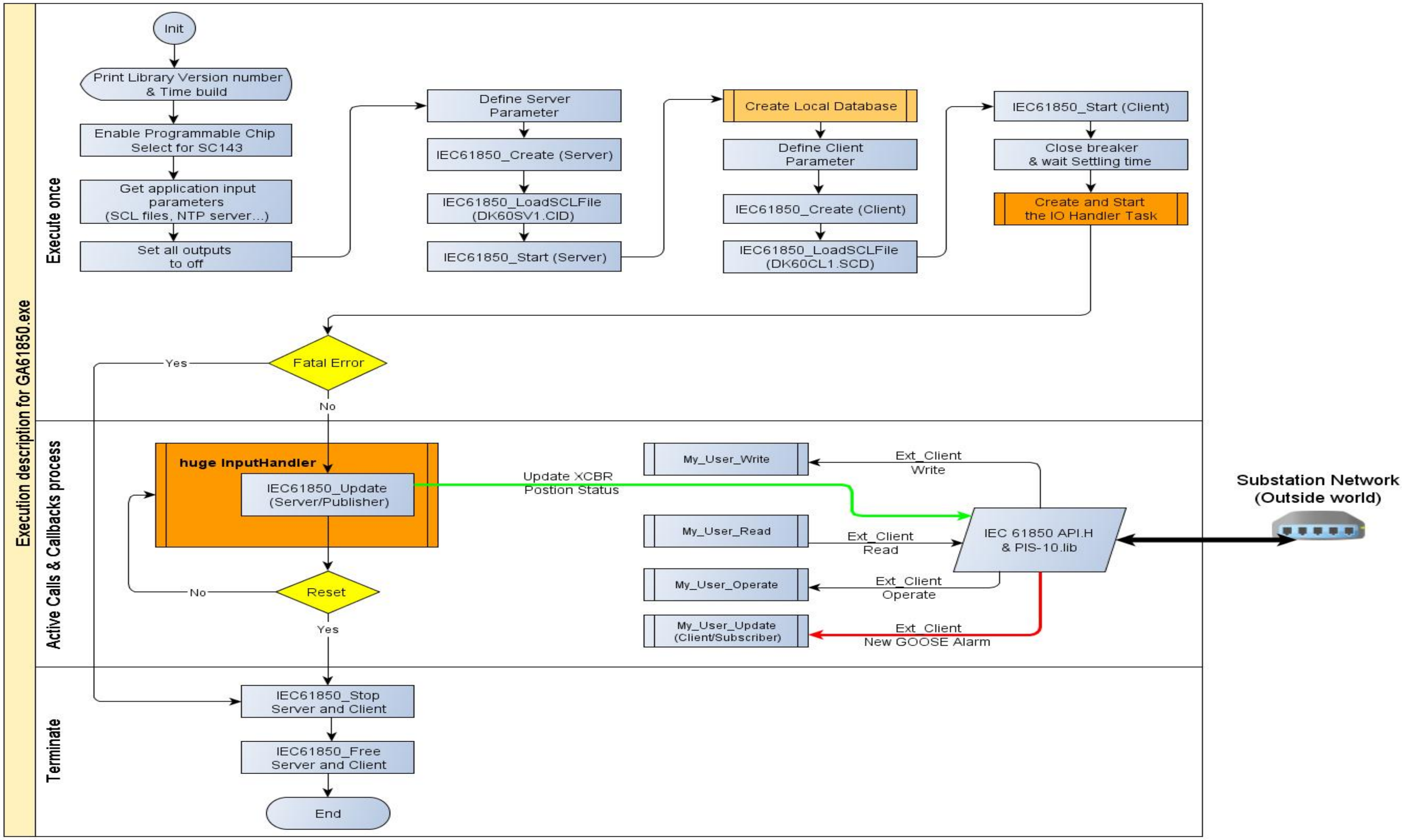

Execution description for GA61850.exe
Execute once
Init
Print Library Version number & Time build
Enable Programmable Chip Select for SC143
Get application input parameters (SCL files, NTP server...)
Set all outputs to off
Define Server Parameter
IEC61850_Create (Server)
IEC61850_LoadSCLFile (DK60SV1.CID)
IEC61850_Start (Server)
Create Local Database
Define Client Parameter
IEC61850_Create (Client)
IEC61850_LoadSCLFile (DK60CL1.SCD)
IEC61850_Start (Client)
Close breaker & wait Settling time
Create and Start the IO Handler Task
Fatal Error
Yes
No
Active Calls & Callbacks process
huge InputHandler
IEC61850_Update (Server/Publisher)
Update XCBR Postion Status
Reset
No
Yes
My_User_Write
My_User_Read
My_User_Operate
My_User_Update (Client/Subscriber)
Ext_Client Write
Ext_Client Read
Ext_Client Operate
Ext_Client New GOOSE Alarm
IEC 61850 API.H & PIS-10.lib
Substation Network (Outside world)
Terminate
IEC61850_Stop Server and Client
IEC61850_Free Server and Client
End

Figure 5.9: Software application internal description

For details on **Step F_6** (Subroutine revision) and **Step G** (Code write-up) (Cf. Figure 5.3), please refer directly to the software application code (see APPENDIX F, page 273).

The next section gives more details on how the local database is structured and how the communication flow sequence happened.

Data communication exchange between PIS-10 stack and process application

The “local database” implemented inside the software application is built up in a way to accommodate Table 5.1 presented earlier on page 133.

The E in GOOSE message represents the occurrence of an Event. The question therefore becomes - how is this “Event” detected/recognised by the system? The manner in which it is achieved in the DK60 environment is that a variable is monitored periodically and updated and stored in a database. When there is a deviation detected from this reference value, an Event is signalled and responded to accordingly.

The details of the interaction between the PIS-10 communication stack and the local database via specific IEC 61850 API functions are shown in Figure 5.10 which represents an elaboration of Figure 5.9. Also note that the general communication scenario has been given at “Figure 4.25: Communication scenario for thesis project “on page 105.

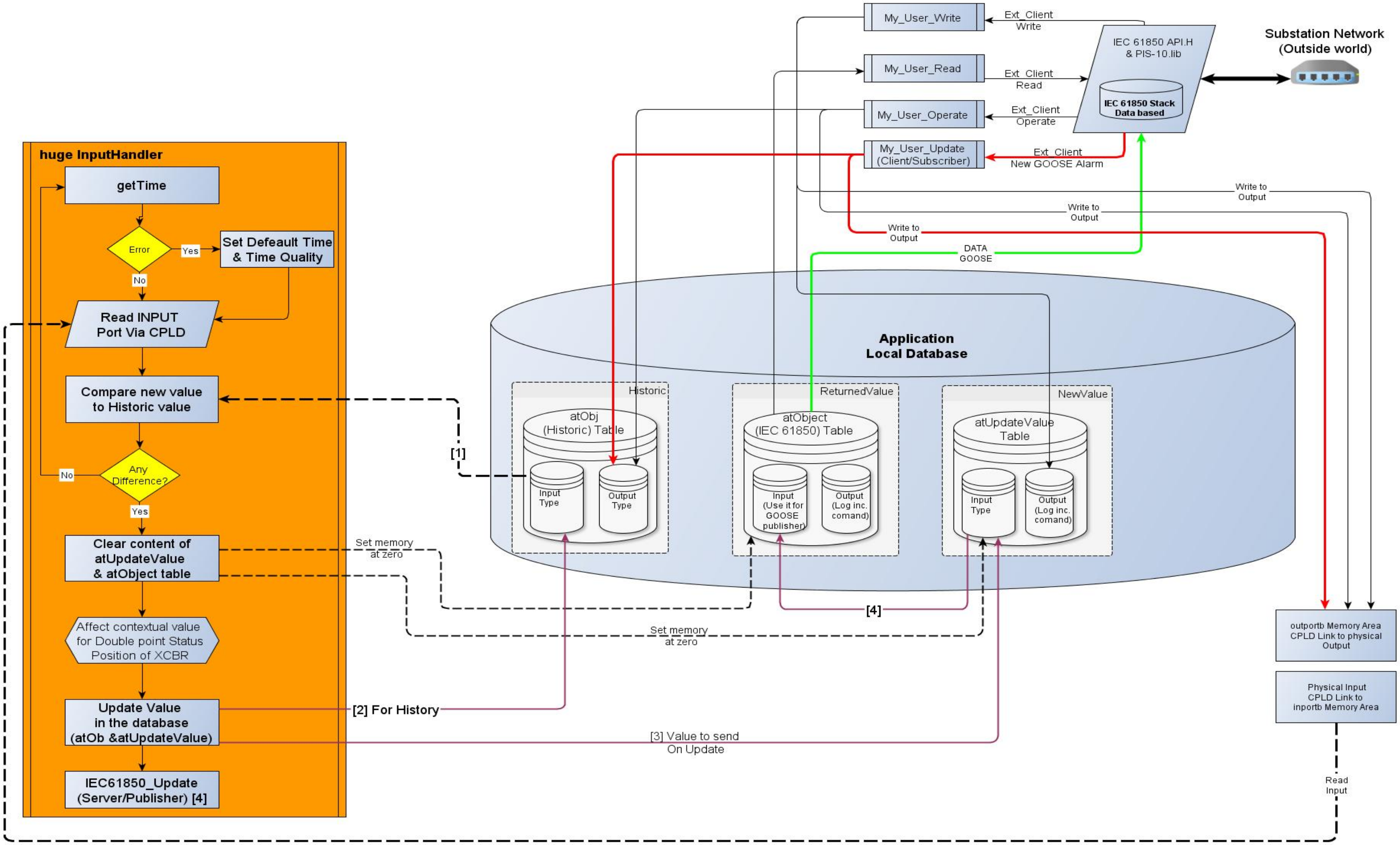

huge InputHandler
getTime
Error
Yes
Set Defeault Time & Time Quality
No
Read INPUT Port Via CPLD
Compare new value to Historic value
No
Any Difference?
Yes
Clear content of atUpdateValue & atObject table
Affect contextual value for Double point Status Position of XCBR
Update Value in the database (atOb &atUpdateValue)
IEC61850_Update (Server/Publisher) [4]
[1]
Set memory at zero
Set memory at zero
[2] For History
[3] Value to send On Update
My_User_Write
My_User_Read
My_User_Operate
My_User_Update (Client/Subscriber)
Ext_Client Write
Ext_Client Read
Ext_Client Operate
Ext_Client New GOOSE Alarm
IEC 61850 API.H & PIS-10.lib
IEC 61850 Stack Data based
Substation Network (Outside world)
Write to Output
Write to Output
Write to Output
DATA GOOSE
Application Local Database
Historic
atObj (Historic) Table
Input Type
Output Type
ReturnedValue
atObject (IEC 61850) Table
Input (Use it for GOOSE publisher)
Output (Log inc. comand)
NewValue
atUpdateValue Table
Input Type
Output (Log inc. comand)
[4]
outportb Memory Area CPLD Link to physical Output
Physical Input CPLD Link to inportb Memory Area
Read Input

Figure 5.10: Local Database relationship with the PIS-10 stack

For the scope of the current thesis which focuses on GOOSE application, the structural arrangement (multiple relational tables) of the local database is sufficient. But the management of the local database can be more refined and more complex especially if one considers the possibility of communication with an external client to the system using an MMS command.

The application local database has a dynamic structure that links local application variables to corresponding variables of the IEC 61850 layer structure. Relevant Data of the information model (Cf. Figure 3.14 of Section 3.4) are logically linked to an entry point in the local database. Also note that, the local database has been arranged in such way to detect change of Data value. This is achieved using an historical table holding back previous Data value.

Specific function such as **MY_User_Operate** and **My_User_Update** are directly translating the value of Data they have received from the network into the local database and at the same time they are used to set (based on additional software routines) the corresponding physical output to the adequate state.
[Note: Prior to the PIS-10 version 1.36, the basic write function (My_User_Write) was used to write data to the local database and at the same time issue an operate command to the corresponding output. With PIS-10 version 1.36, the basic write function (My_User_Write) is no longer used to issue an "operate" command to the output. There is now a dedicated function (My_User_Operate) that handles this task. This has changed the way IEC 61850 programmers have to manage the internal local database structure and the Update sequence.]

With respect to the Circuit Breaker case study, any change (representing an Event) in the local data base output section (see Table 5.1) is going to be automatically reflected at the extension pin header S10 (Unused CPLD pin). The extension pin header S10 is the point from where the sub-system B2 that corresponds to the logic control (connected to the Circuit Breaker emulator) is connected to the DK60 board (See Figure 5.18 and Figure 5.18 on page 145 and 146).

In the same manner the Data value to be passed into the PIS-10 stack (from the Circuit Breaker emulator) is directly translated from the physical point to its logical location in the "local data base". This is done inside the "**huge InputHandler**" function. The Circuit Breaker emulator returns a binary two-bit word as its status which is to be passed through the IEC 61850 Stack (e.g. Double point position status).

The binding of the physical input to the application Data Object has been catered for as a point-to-point implementation. The default Data that results from such a connection is a Single bit binary (generically referred to as “Boolean”). The problem now is that there is more than one type of data in use in the substation. The current project involves four (4) basic Data types which are: Boolean, Double point position status (two-bit binary), quality and time.

Unfortunately within the development environment, only the single-bit binary, quality, and time are catered for, and the two-bit binary has to specially be accommodated (no array processing available). To create a data-type such as the “Double point status” of a Circuit breaker, you may use what is known as a container structure. In this thesis the container structure that holds the position status of the prototyped circuit breaker is a variable of type “Double point position status” (Dbpos).

The link between the container and the individual input point that represents the position of the circuit breaker (See Figure 5.18) is made by creating a software loop that reads the individual position bit signals sequentially from the circuit breaker emulator (System C: Figure 5.18) then, based on those values, link a predefined value to the container structure. The values to load in the container based on single point position entries are as depicted in Table 5.3 below:

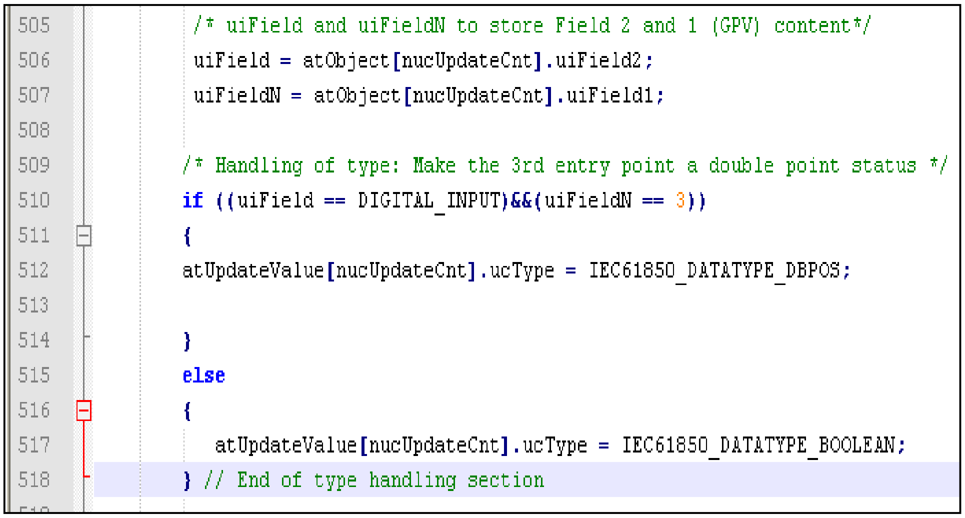
Table 5.3: Container Value for Circuit breaker position

Position Bit_2	**Position Bit_1**	**Container value as per PIS-10 v-1.36 (XCBR.Pos.stVal)**	**Interpretation**
0	0	IEC61850_DB_POS_OFF {0x00}	“Intermediate state”
0	1	IEC61850_DB_POS_FALSE {0x40}	“Off/ Open state”
1	0	IEC61850_DB_POS_TRUE {0x80}	“On/ Close state”
1	1	IEC61850_DB_POS_INVALID {0xC0}	“Invalid state”

After this step has been completed, a call is performed to the IEC61850_Update with the corresponding parameters (see APPENDIX F).

The following gives some code samples that illustrate what has previously been mentioned above.

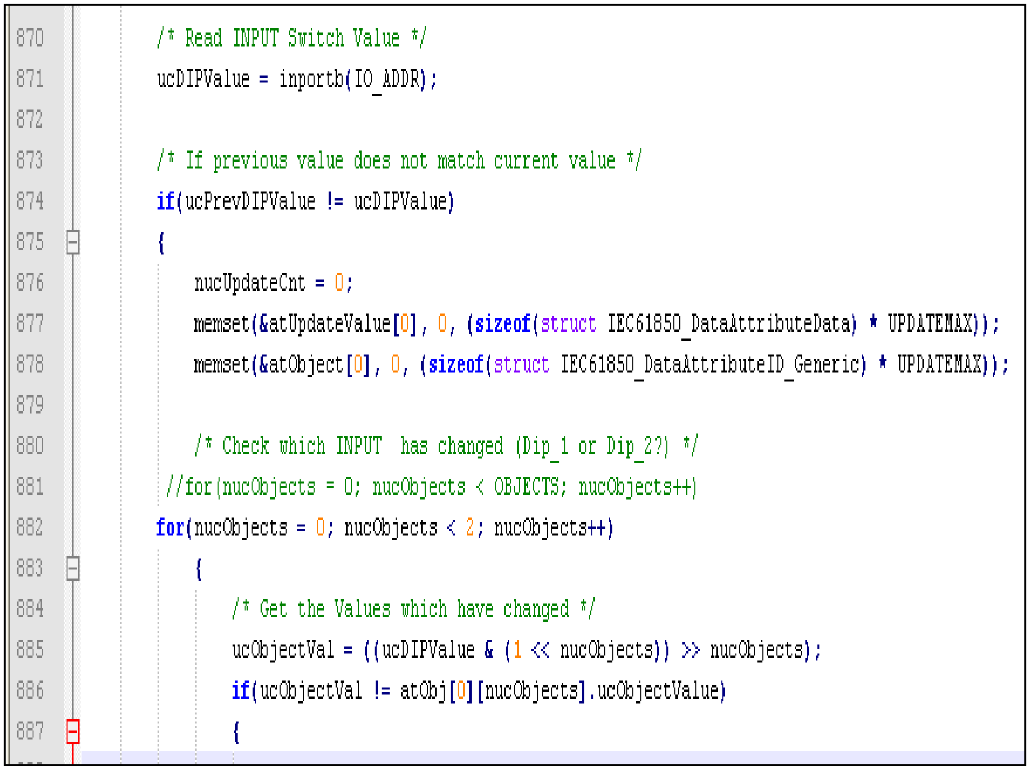
A code sample for adapting the link between the database and the GOOSE message to accommodate two data types instead of one has been developed and is presented below;



```
      /* uiField and uiFieldN to store Field 2 and 1 (GPV) content*/
      uiField = atObject[nucUpdateCnt].uiField2;
      uiFieldN = atObject[nucUpdateCnt].uiField1;

     /* Handling of type: Make the 3rd entry point a double point status */
     if ((uiField == DIGITAL_INPUT)&&(uiFieldN == 3))
     {
     atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_DBPOS;

     }
     else
     {
        atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_BOOLEAN;
     } // End of type handling section
```

Figure 5.11: Data type managment for local database

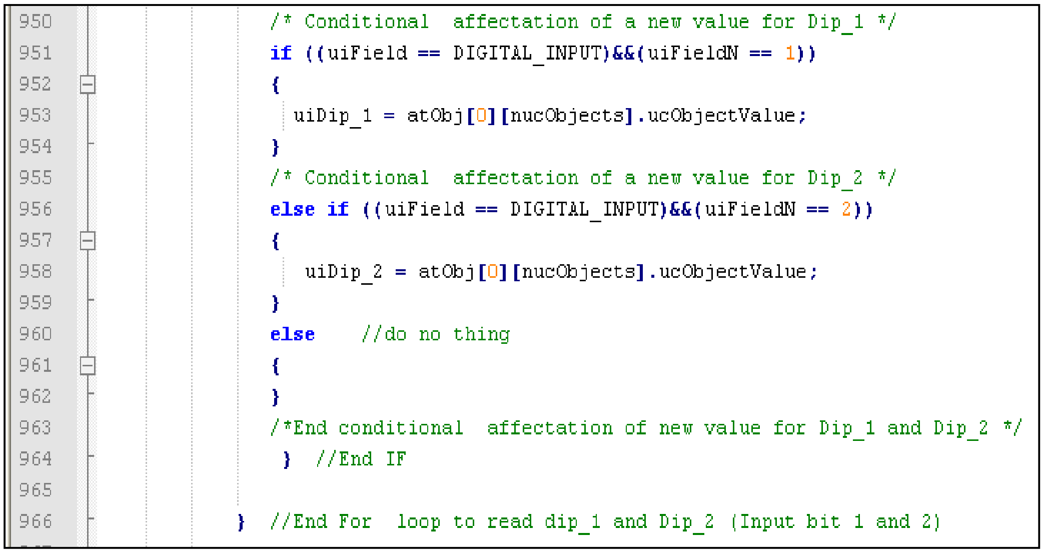
The real-time updating of two data types instead of one - to track GOOSE events on the DK60 platform; a typical scenario is the following - Digital I/O points continuously monitored, status of breaker changes; status change detected and compared to value in historical field in database (see Figure 5.12)



```
    /* Read INPUT Switch Value */
    ucDIPValue = inportb(IO_ADDR);

    /* If previous value does not match current value */
    if(ucPrevDIPValue != ucDIPValue)
    {
        nucUpdateCnt = 0;
        memset(&atUpdateValue[0], 0, (sizeof(struct IEC61850_DataAttributeData) * UPDATEMAX));
        memset(&atObject[0], 0, (sizeof(struct IEC61850_DataAttributeID_Generic) * UPDATEMAX));

        /* Check which INPUT  has changed (Dip_1 or Dip_2?) */
     //for(nucObjects = 0; nucObjects < OBJECTS; nucObjects++)
    for(nucObjects = 0; nucObjects < 2; nucObjects++)
        {
            /* Get the Values which have changed */
            ucObjectVal = ((ucDIPValue & (1 << nucObjects)) >> nucObjects);
            if(ucObjectVal != atObj[0][nucObjects].ucObjectValue)
            {
```

Figure 5.12: Monitoring of Input for a change of status

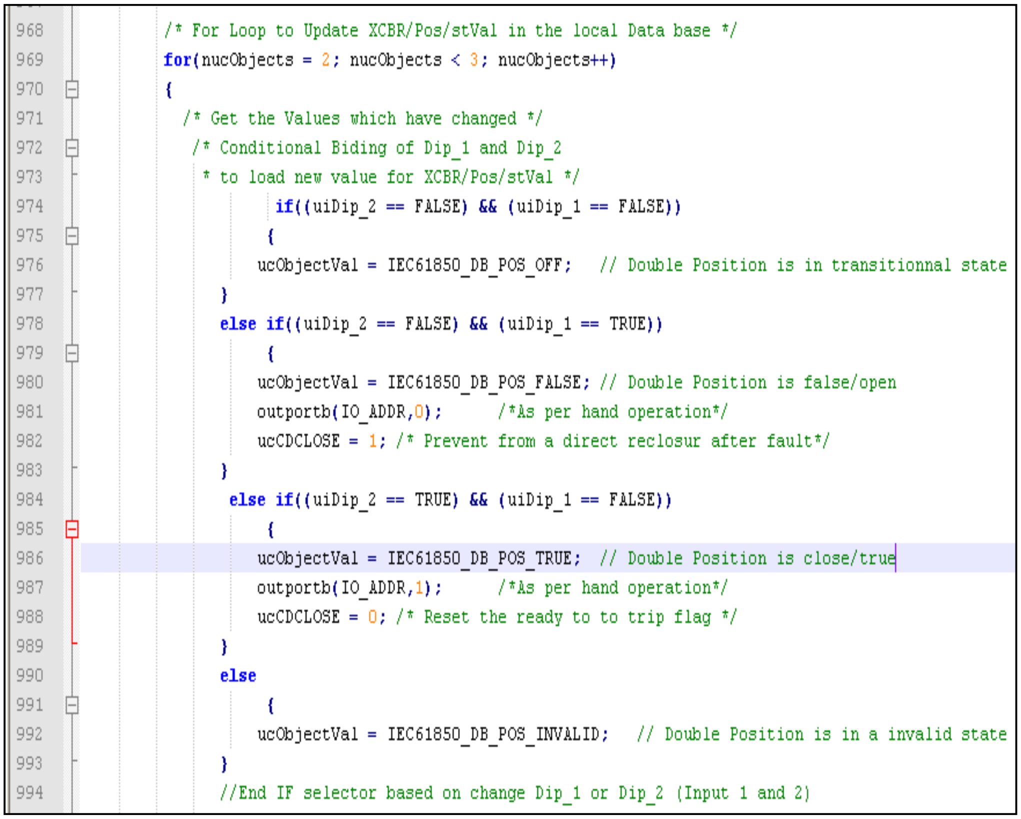
Assignment of value pointed to by database pointer to a variable: Computation for new value for double point status – using a loop, store individual bits in separate variables then use logic test conditions (see Figure 5.13)



```
            /* Conditional  affectation of a new value for Dip_1 */
            if ((uiField == DIGITAL_INPUT)&&(uiFieldN == 1))
            {
              uiDip_1 = atObj[0][nucObjects].ucObjectValue;
            }
            /* Conditional  affectation of a new value for Dip_2 */
            else if ((uiField == DIGITAL_INPUT)&&(uiFieldN == 2))
            {
               uiDip_2 = atObj[0][nucObjects].ucObjectValue;
            }
            else    //do no thing
            {
            }
            /*End conditional  affectation of new value for Dip_1 and Dip_2 */
             }  //End IF

         }  //End For  loop to read dip_1 and Dip_2 (Input bit 1 and 2)
```

Figure 5.13: Saving Status of Input into global variabale

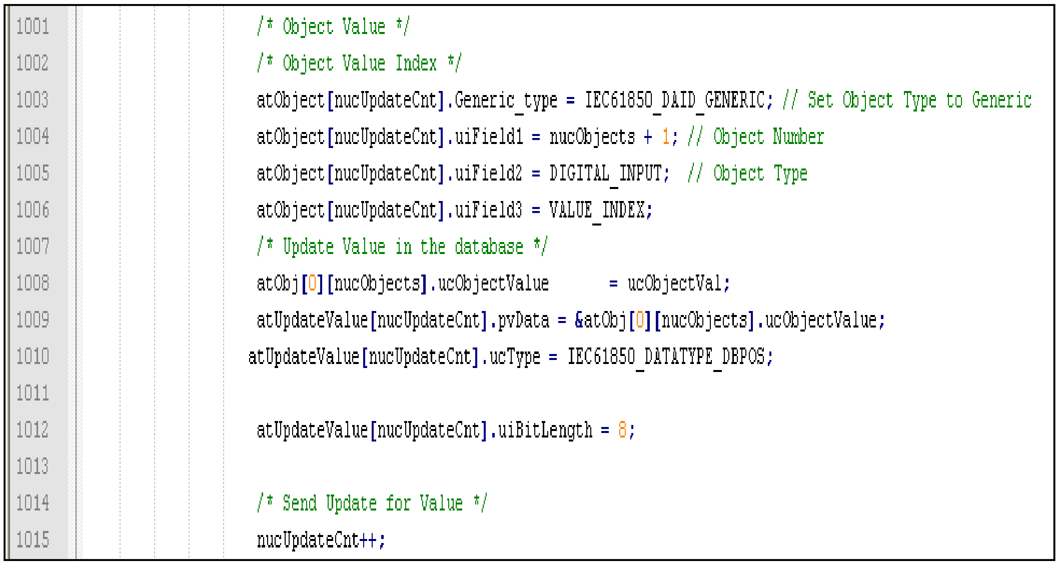
Determine a value to assign to a structure used to hold the double-point-status value (see Figure 5.14).



```
    /* For Loop to Update XCBR/Pos/stVal in the local Data base */
    for(nucObjects = 2; nucObjects < 3; nucObjects++)
    {
      /* Get the Values which have changed */
       /* Conditional Biding of Dip_1 and Dip_2
        * to load new value for XCBR/Pos/stVal */
              if((uiDip_2 == FALSE) && (uiDip_1 == FALSE))
              {
             ucObjectVal = IEC61850_DB_POS_OFF;   // Double Position is in transitionnal state
        }
        else if((uiDip_2 == FALSE) && (uiDip_1 == TRUE))
              {
             ucObjectVal = IEC61850_DB_POS_FALSE; // Double Position is false/open
             outportb(IO_ADDR,0);      /*As per hand operation*/
             ucCDCLOSE = 1; /* Prevent from a direct reclosur after fault*/
        }
         else if((uiDip_2 == TRUE) && (uiDip_1 == FALSE))
              {
             ucObjectVal = IEC61850_DB_POS_TRUE;  // Double Position is close/true
             outportb(IO_ADDR,1);      /*As per hand operation*/
             ucCDCLOSE = 0; /* Reset the ready to to trip flag */
        }
        else
              {
             ucObjectVal = IEC61850_DB_POS_INVALID;   // Double Position is in a invalid state
        }
        //End IF selector based on change Dip_1 or Dip_2 (Input 1 and 2)
```

Figure 5.14: Conditional affection of container value for circuit breaker position

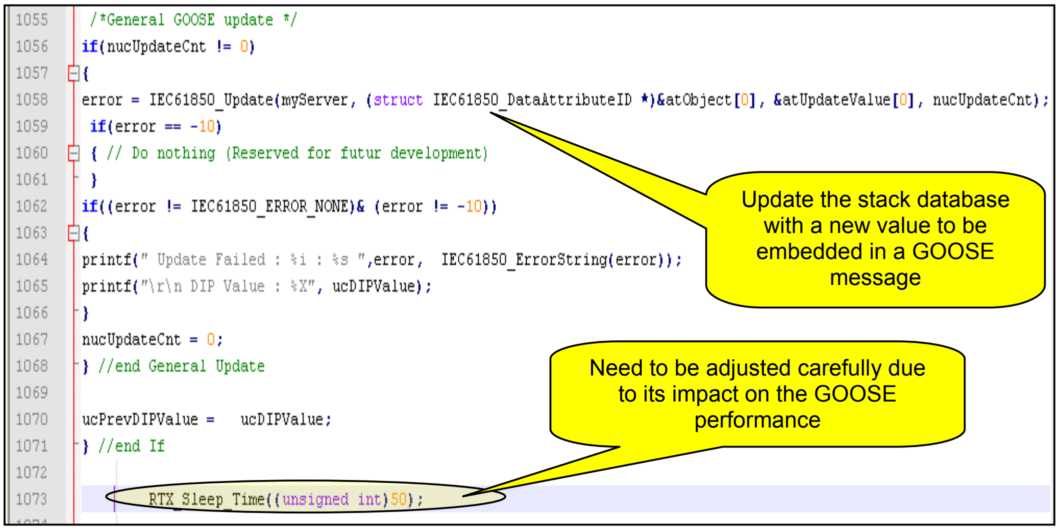
Figure 5.15 shows the update of the individual bit values in the historical table within the database is done;



```c
/* Object Value */
/* Object Value Index */
atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
atObject[nucUpdateCnt].uiField3 = VALUE_INDEX;
/* Update Value in the database */
atObj[0][nucObjects].ucObjectValue      = ucObjectVal;
atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].ucObjectValue;
atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_DBPOS;

atUpdateValue[nucUpdateCnt].uiBitLength = 8;

/* Send Update for Value */
nucUpdateCnt++;
```

Figure 5.15: Update of local database

In Figure 5.16, a call to the “IEC61850_Update” function of the PIS010 stack is performed in order to update the GOOSE value; (GOOSE performance affected by pause in this polling loop);



```c
 /*General GOOSE update */
if(nucUpdateCnt != 0)
{
error = IEC61850_Update(myServer, (struct IEC61850_DataAttributeID *)&atObject[0], &atUpdateValue[0], nucUpdateCnt);
 if(error == -10)
 { // Do nothing (Reserved for futur development)
 }
if((error != IEC61850_ERROR_NONE)& (error != -10))
{
printf(" Update Failed : %i : %s ",error,  IEC61850_ErrorString(error));
printf("\r\n DIP Value : %X", ucDIPValue);
}
nucUpdateCnt = 0;
} //end General Update

ucPrevDIPValue =   ucDIPValue;
} //end If

    RTX_Sleep_Time((unsigned int)50);
```

Update the stack database with a new value to be embedded in a GOOSE message

Need to be adjusted carefully due to its impact on the GOOSE performance

Figure 5.16: Update of IEC 61850 database

The procedure above refers to the case when the event is generated from within the DK60 environment (GOOSE transmitted). A different mechanism is to be considered when the event is generated externally (GOOSE received).

5.3 Interfacing and System integration

Re-direction of binary i/o's

The basic hardware connectivity between the DK60 board and the circuit breaker emulator is presented in Figure 5.17. The Complex Programmable Logic Device (CPLD) is used to send the breaker control value signal to the "Pulse Control Logic" stage via the extension pin header [S10] of the Dk60 board (Cf. Section 4.3). The CPLD also passes position information of the emulated circuit breaker to the software application (GA61850.exe).

The DK60 board originally came with the binary I/O routed to switches and LEDs. For the application developed redirection was required of the binary I/O to different pins. The VHDL code required to re-programme the CPLD to achieve the goal stated above, is listed in the file dk60_xcbr.vhdl (in Appendix E)

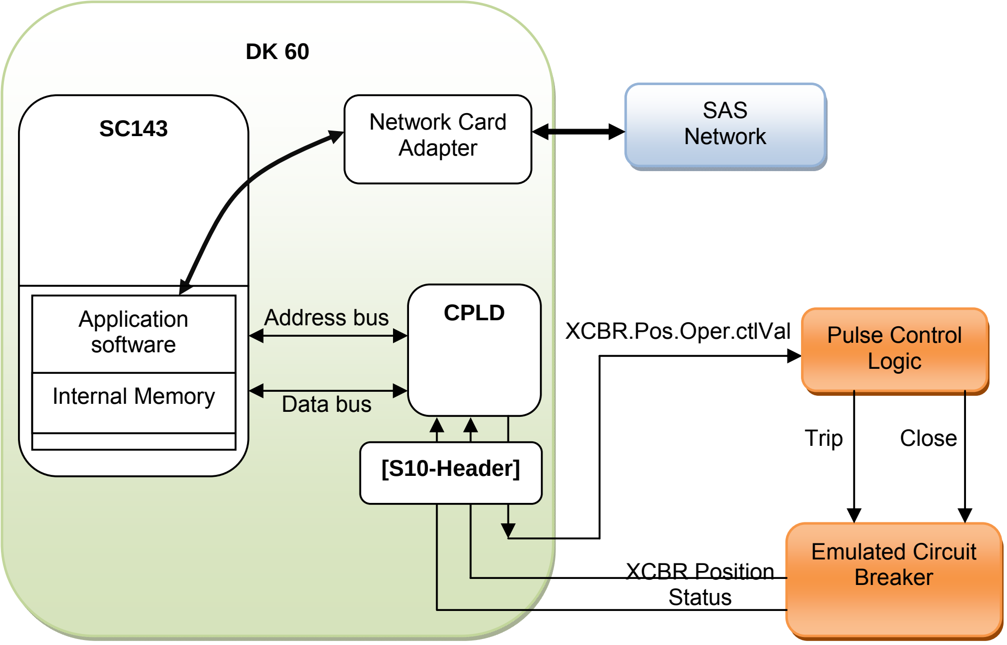


Figure 5.17: Basic hardware connectivity

Figure 5.18 presents a view of the system integration that takes into account the software modularity. The system B is the Actuator Node. It is composed of two (2) subsystems. The Subsystem B_1 is the Dk60 board (already presented in section 4.3). The subsystem B_2 is made of logic gate ICs and RC circuit. The subsystem B_2 is used as a pulse control circuit.

The pulse control circuit (Subsystem B_2) takes as input the internal memory state of "XCB.Pos.Oper.ctlVal" from which it generates two (2) outputs (SW_Trip and SW_Close). With respect to a change of state in the data attribute "ctlVal", the pulse control circuit is going to generate a pulse signal at the suitable output to either close or open the breaker.

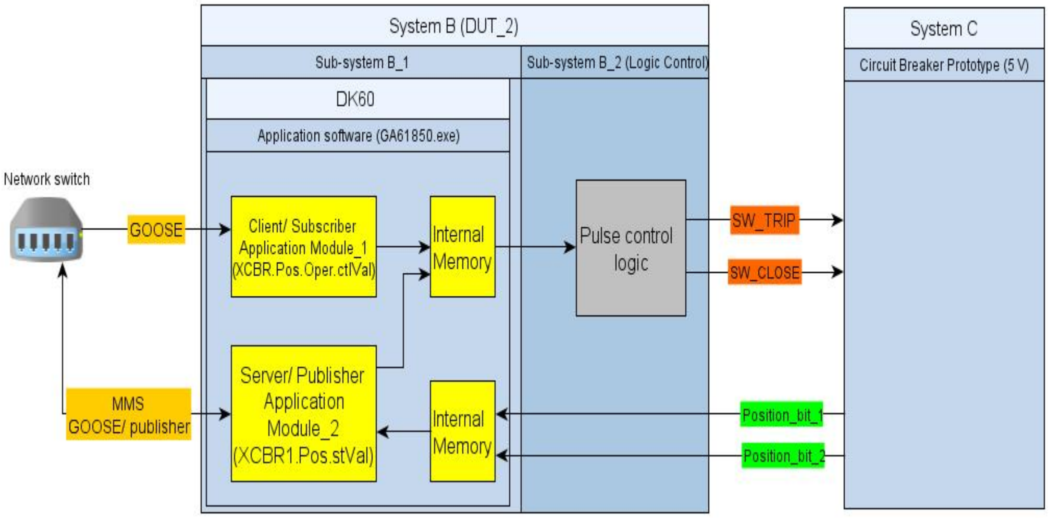
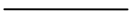
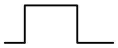
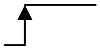
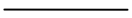
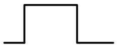
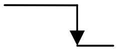
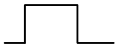
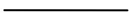

System B (DUT_2)
Sub-system B_1
Sub-system B_2 (Logic Control)
DK60
Application software (GA61850.exe)
Network switch
GOOSE
Client/ Subscriber Application Module_1 (XCBR.Pos.Oper.ctlVal)
Internal Memory
Pulse control logic
SW_TRIP
SW_CLOSE
System C
Circuit Breaker Prototype (5 V)
MMS GOOSE/ publisher
Server/ Publisher Application Module_2 (XCBR1.Pos.stVal)
Internal Memory
Position_bit_1
Position_bit_2

Figure 5.18: System integration for Actuator Node

With respect to the current thesis, the "Actuator" functionality is then contained within the "System B" that constitutes the "Intelligent Circuit Breaker Controller". Also note there is no power equipment (see Figure 4.20 page 97) to serve as actuating module (Circuit breaker Motor) as it has been decided to use a compatible 5 volts latching relay that is going to serve as an emulated breaker (System C) for laboratory experimentation.

Table 5.4 presents the Sub-system B_2 output signal variation based on its input (OUT_1) taken from the Sub-system B_1 (DK60 board). The Output signals (SW_TRIP and SW_CLOSE) are then connected to the emulated circuit breaker plant which is the System C.

Table 5.4: Output logic control

OUT_1 **[XCBR$Pos$Oper$ctlVal** **Sub-system B_1]**	**SW_TRIP** **[Pulsed/ Output]**	**SW_CLOSE** **[Pulsed/ Output]**
Initialization of Client/Server application		
0	1 0 	1 0 
On Value change		
1 0 	1 0 	1 0 
1 0 	1 0 	1 0 

The Sub-system B_2 (Pulse control logic) is built up by means of individual Integrated Circuits (ICs) that implement logic gates. In order to generate the pulse control strategy presented in Table 5.4. The circuit diagram shown in Figure 5.19 has been adopted.

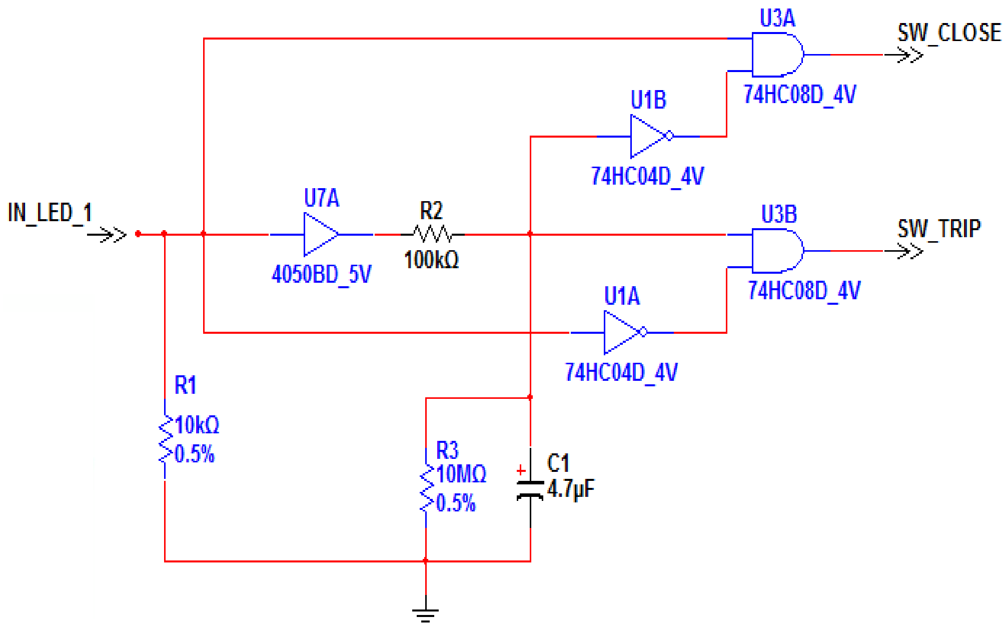


Figure 5.19: Pulse Control schematic

The driving input signal of the Pulse control circuit is taken from the memory location that corresponds to the first Output LED on the DK60 board (LED_1). The resulting output signals (SW_CLOSE and SW_TRIP) are the input signals of the System C (Emulated circuit breaker).

The pulse duration is achieved by the RC circuit composed of **R2, R3** and **C1**. Based on the characteristics of the type of component used as physical circuit breaker, the duration of the pulse to close the breaker is about 50 mS and the duration of the pulse to trip the breaker is 100 mS. Note that in commercial IEDs a normal pulse has duration of about 1second (user configurable).

The System C represents a circuit breaker emulator prototype (5v compatible) to be used for laboratory purposes. It is composed of a two (2) way latching relay (DSP2a/ see APPENDIX I). In the case one wants to connect the System B to a circuit breaker prototype (System C) using a voltage higher than 5 volt, optical isolation of both systems should be considered. The schematic of the System C is shown Figure 5.20.

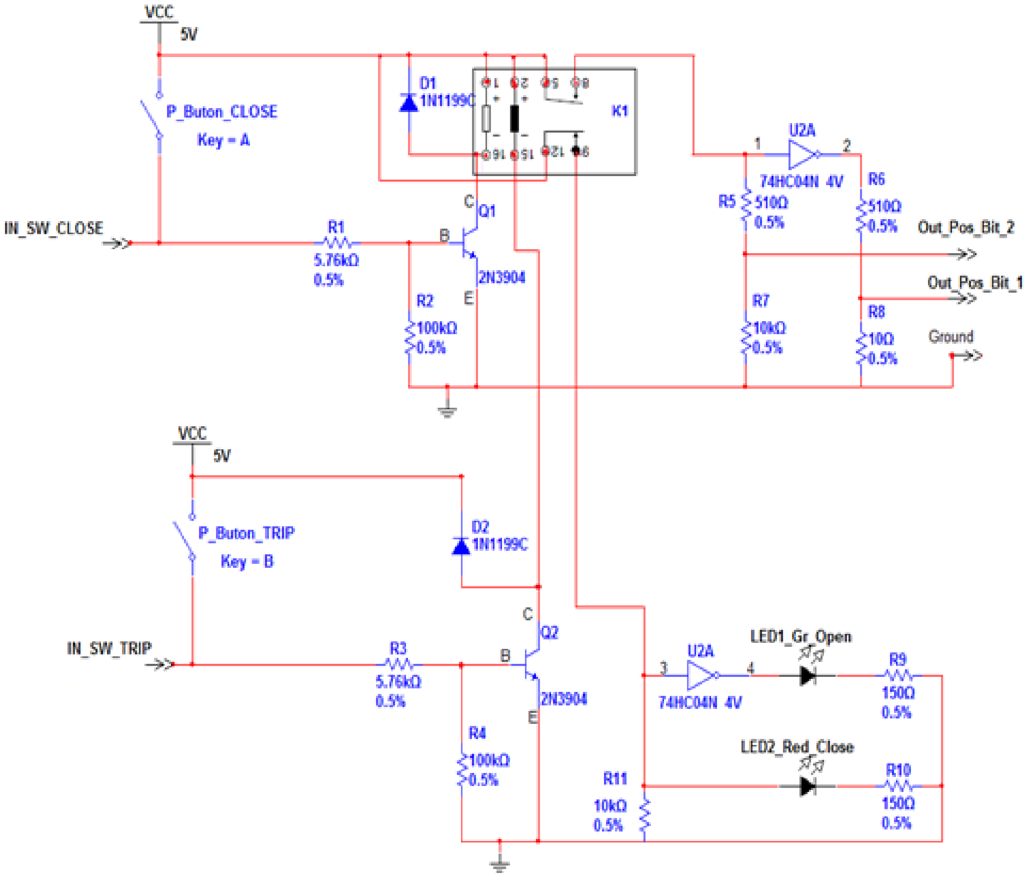


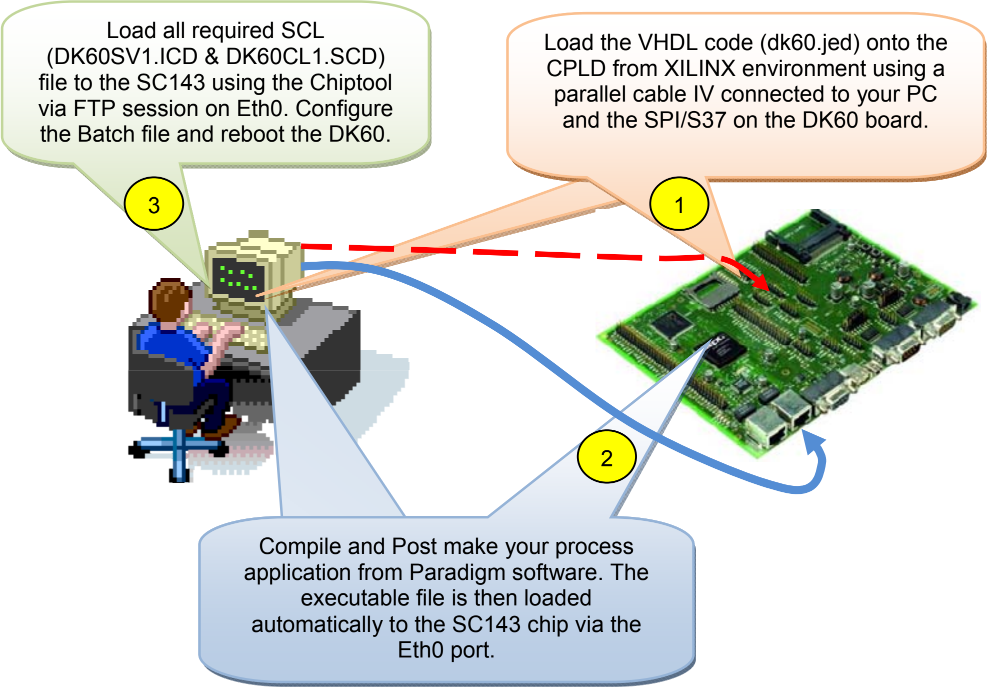
Figure 5.20: Prototype Circuit breaker schematic

Please find the complete diagram for both the Pulse control logic and the prototype Circuit breaker in appendices G and H (pp.303-305).

The resulting file for each process can be found in Appendices as listed below:

- Appendix C: Engineering the system architecture: filename dk60sv1.icd and Appendix D: dk60cl1.scd, for SCL file development – XML files;
- Appendix F: application code software, for “C” source code of the process application: filename ga61850.c ; and
- Appendix E: Hardware interfacing (re-direction of binary i/o’s) for the CPLD on the DK60 board: filename dk60_xcbr.vhdl VHDL code.

Note: Although the software was considered and developed in the sequence shown in Figure 5.2, Figure 5.21 below shows the practical sequence order for DK60 file upload. This is done in the reverse order as compared to the logical development flow steps described in Figure 5.2.



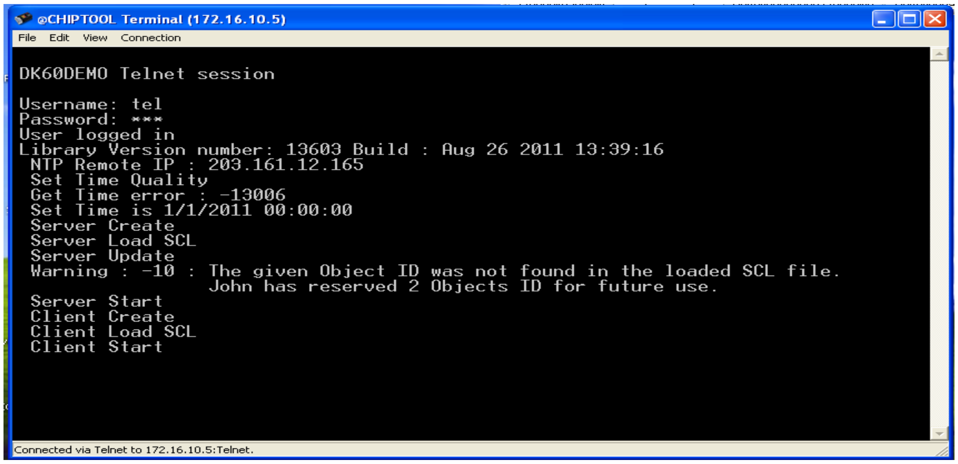
Load all required SCL (DK60SV1.ICD & DK60CL1.SCD) file to the SC143 using the Chiptool via FTP session on Eth0. Configure the Batch file and reboot the DK60.

Load the VHDL code (dk60.jed) onto the CPLD from XILINX environment using a parallel cable IV connected to your PC and the SPI/S37 on the DK60 board.

Compile and Post make your process application from Paradigm software. The executable file is then loaded automatically to the SC143 chip via the Eth0 port.

Figure 5.21: Prefered Sequence for DK 60 file upload

Figure 5.22 presents the screen result obtained via a Telnet connection to the DK60 board at start up (Program to be started has been predefined in the Batch file on the SC143 Chip).



```
DK60DEMO Telnet session

Username: tel
Password: ***
User logged in
Library Version number: 13603 Build : Aug 26 2011 13:39:16
 NTP Remote IP : 203.161.12.165
 Set Time Quality
 Get Time error : -13006
 Set Time is 1/1/2011 00:00:00
 Server Create
 Server Load SCL
 Server Update
 Warning : -10 : The given Object ID was not found in the loaded SCL file.
                 John has reserved 2 Objects ID for future use.
 Server Start
 Client Create
 Client Load SCL
 Client Start
```

Connected via Telnet to 172.16.10.5:Telnet.

Figure 5.22: Telnet program feedback at start up

The “Error: 13006” (Figure 5.22) indicates that no Network Time Protocol (NTP-Server) was present or reachable.

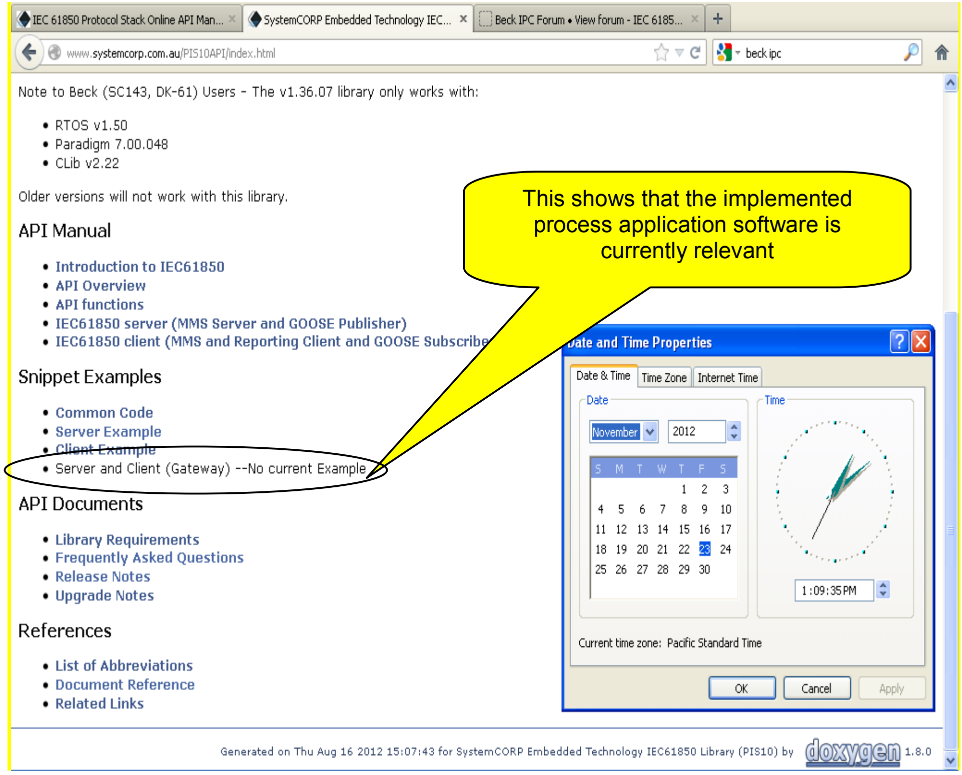
5.4 Reported issues

Some of the issues experienced (non exhaustive) during development can be stated as follows:

- Documentation lacking in accuracy or detail on many aspects of the Protocol Integration Stack (PIS-10) and associated sample code delivered with it (e.g. the target model used for code compilation is of type "huge" and not "large" as stated in the original manual delivered with the DK61 Kit).
- The Protocol Integration Stack (PIS-10) used (early version), did not have the level of maturity (of the vastly improved current version) when the research project was started.
- The MAC address of a device publishing GOOSE message (commercial IED) should normally be specified in the "DK60CL1.SCD" file (see DK60CL1.SCD). Unfortunately, due to some changes in the SC143 Operating System structure of the Beck IPC; the use of the subscription functionality of the PIS-010 stack (from SystemCorp) was problematic (at the time). The Error reported was as follows: <Int 0xC2 AX=0x060A not supported at task 1640!> - meaning that the process application software developed could not access the content of the GOOSE packet due to some incompatibility between the PIS-010 and the interrupt service routine of the Beck IPC packet driver's. The solution that worked in the case of the current research project was to delete the MAC address of the commercial IED from the SCL file. This then allowed the PIS-010 stack to retrieve and process the Ethernet frame. Although this was a solution for this project, it is not strictly IEC 61850 compliant, as the MAC address is mandatory in the SCL file.

  There is ongoing collaborative work being done between SystemCorp and Beck IPC in order to resolve outstanding issues such as those described above. The fact that the (proprietary) Operating System of Beck IPC is changing from time to time as well as periodic changes in the PIS-010 stack allow for compatibility issues to take place (developers need to be aware of this possibility of mismatch). This has made the development work of the process application software very difficult.

  As a matter of fact, there is no current demo sample of source code available at the SystemCorp website that corresponds to the type of implementation described and implemented in this research (Server/Publisher and Client/Subscriber example) Figure 5.23.


IEC 61850 Protocol Stack Online API Man...
SystemCORP Embedded Technology IEC...
Beck IPC Forum • View forum - IEC 6185...
www.systemcorp.com.au/PIS10API/index.html
beck ipc

Note to Beck (SC143, DK-61) Users - The v1.36.07 library only works with:

- RTOS v1.50
- Paradigm 7.00.048
- CLib v2.22

Older versions will not work with this library.

API Manual

- Introduction to IEC61850
- API Overview
- API functions
- IEC61850 server (MMS Server and GOOSE Publisher)
- IEC61850 client (MMS and Reporting Client and GOOSE Subscribe

Snippet Examples

- Common Code
- Server Example
- Client Example
- Server and Client (Gateway) --No current Example

API Documents

- Library Requirements
- Frequently Asked Questions
- Release Notes
- Upgrade Notes

References

- List of Abbreviations
- Document Reference
- Related Links

Generated on Thu Aug 16 2012 15:07:43 for SystemCORP Embedded Technology IEC61850 Library (PIS10) by doxygen 1.8.0

This shows that the implemented process application software is currently relevant

Current time zone: Pacific Standard Time

Figure 5.23: Developer's page at SystemCorp website

5.5 Conclusion

This chapter has first presented the software development of an IEC 61850 application based on the PIS-010 stack library from SystemCorp. Secondly, the Chapter has presented the system Hardware integration and inter-action of the PIS-010 stack with the physical input/output (I/O) points on the board. It has shown also how to interconnect the DK60 board to a compatible 5v latching relay implemented as part of a circuit breaker emulator.

The software code for the prototyped actuator node has been done in ANSI C using Paradigm C\C++ for a Beck single chip solution. The VHDL "configware" code for the CPLD on the DK 60 board has been re-written to accommodate the particular thesis project design using XILINX ISE WebPACK 6. Other software tools have also played an important role for the completion of the present thesis. Table 5.5 below details the complete list of software tools and the filenames for the code developed and used during this project.

Table 5.5: Project software tools

Software Tools	Purpose
Management of the DK60 board	
Chiptool (V 6.1.3.6)	Chiptool for the management of the SC143 chip at the Dk60 board in order to configure basic setting of the board and much more http://beck-ipc.com/en/download/index.asp
Programming	
Parading C\C++ for Beck IPC (v7.00.048)	Application (GA61850) software code development in C. http://beck-ipc.com/en/download/index.asp
XILINX ISE WebPACK (V.6)	Application Hardware (dk60_xcbr vhdl) code programming in VHDL http://www.xilinx.com/webpack/classics/wpclassic/
Simulation and Testing	
IEDScout (V 2.11)	A multipurpose IED Universal client used for GOOSE subscription and simulation. Remote control of the actuator node and as a Data model browser
Test Universe (QuickCMC and GOOSE) (V 2.40)	Compact suite of diverse tools that enable testing of protective relays. http://www.omicron.at/en/products/pro/secondary-testing-calibration/sw/
Network analyser	
Wireshark (v 1.8.2)	To capture Ethernet packet (GOOSE…) http://www.wireshark.org/
MMS Ethereal (V 1.1.0)	To capture MMS packet http://www.ethereal.com/
Packet builder (v 1.0)	For advance cyber security investigation http://www.colasoft.com/packet_builder/
Network Switch configuration	
RuggedExplorer (v 1)	Specific tools for Rugged switches. Help to discover and configure the switch online. http://www.ruggedcom.com/products/explorer/
File comparison and management	
WinMerge (V 2.12.4.0)	This is power full windows based tools to compare quickly files between then and pick up were differences are. http://winmerge.org/
Notepad++ (V 6.1.6)	Help in exporting source code from paradigm to MS word 2007. You can use this tool if you want to keep a certain format of your code especially the colour difference between line code and comments. http://notepad-plus-plus.org/
SCL/ XML file design	
ICD Designer (v 1.1.0.12)	Proprietary tools from SystemCorp for the development of SCL files (dk60sv1.icd and dk60cl1.scd) http://www.systemcorp.com.au/component/content/article/23-software/46-icd-designer.html
XML Marker (V 1.1)	Free tools for the development of XML based type file http://symbolclick.com/
Commercial IED Tools used in the project	
SEL Architect	SEL tools for IEC61850 communication configuration between devices https://www.selinc.com/SEL-5032/
DIGSI 4 (v 4.82.16)	SIEMENS tools for device configuration and IEC 61850 communication configuration between devices http://www.energy.siemens.com/co/en/automation/power-transmission-distribution/protection/software/digsi-4.htm
Modelling and Algorithm diagrams (Graphical)	
Eclipse 3.7 (Indigo)/ TOPCASED project	UML 2 modelling platform specially designed for embedded system hardware and software development http://www.topcased.org/index.php/content/view/40/54/
yEd (v 3.9.2)	Simple tools, easy to use for algorithm development and other graphical representation http://www.yworks.com/en/products_yed_about.html

The next Chapter presents the test-scenario for the system developed, and analyses the results obtained from a GOOSE message structure point of view as well as from a GOOSE performance point of view.

CHAPTER SIX:

CASE STUDY IMPLEMENTATION: TESTING AND RESULTS ANALYSIS

6.1 Introduction

One of the messages associated with the GSE services, is the Generic Object Oriented Substation Event (GOOSE) message at the station “**Station Bus**” and also at “**Process Bus**” (Cf. Section 3.4.2 .Definition of a GOOSE **message**, page 59).

Logical interfaces have been defined in the standard. For the simulation and analysis of an IEC 61850 based GOOSE message, the interface 8 at the Bay unit level is of primary interest (direct data exchange between the bays especially for fast response e.g. protection functions). GOOSE messages can also be used at the process bus as explained in Section 4.2 (IF5: Control-data exchange between process and bay level). Please refer to section 3.4.4 for a short description of each interface.

To gain insight into the structure of GOOSE messages, a general case study is first presented in Section 6.2, in which a GOOSE message is simulated by means of OMICRON IEDScout software, captured via network protocol analyzer software - Wireshark, and analyzed relative to the structure as defined in the IEC61850 Standard. Further confirmation of the structure of a GOOSE message and the encoding of its data, is obtained through experimentation utilizing commercially available IEDs.

In Section 6.3 the experimentation and testing for the case-study for the research project utilizing the embedded-system-based controller is outlined. It consists of the demonstration of the deployment of an Overcurrent scheme distributed between a commercial protective relay (SIEMENS_SIPROTEC 7SD5) and the embedded DK60 experimental board. Figure 6.1 below illustrates the topology that is used for that purpose. A step by step description of the experiment is also provided in Section 6.3.

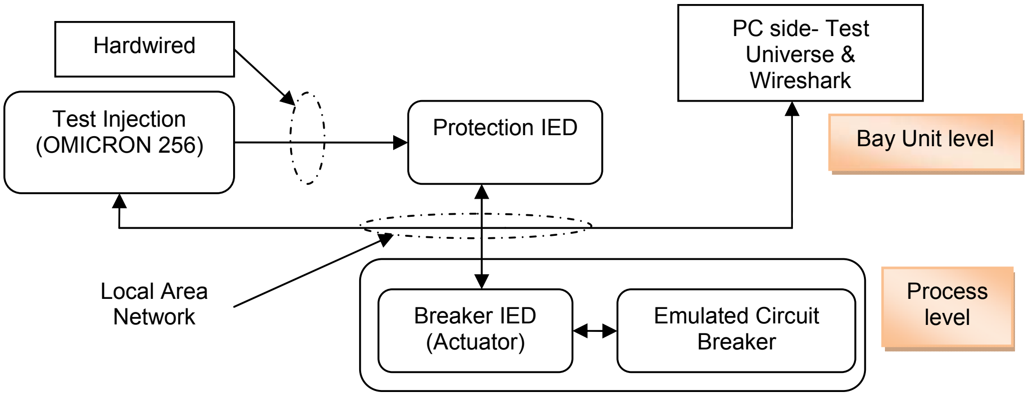
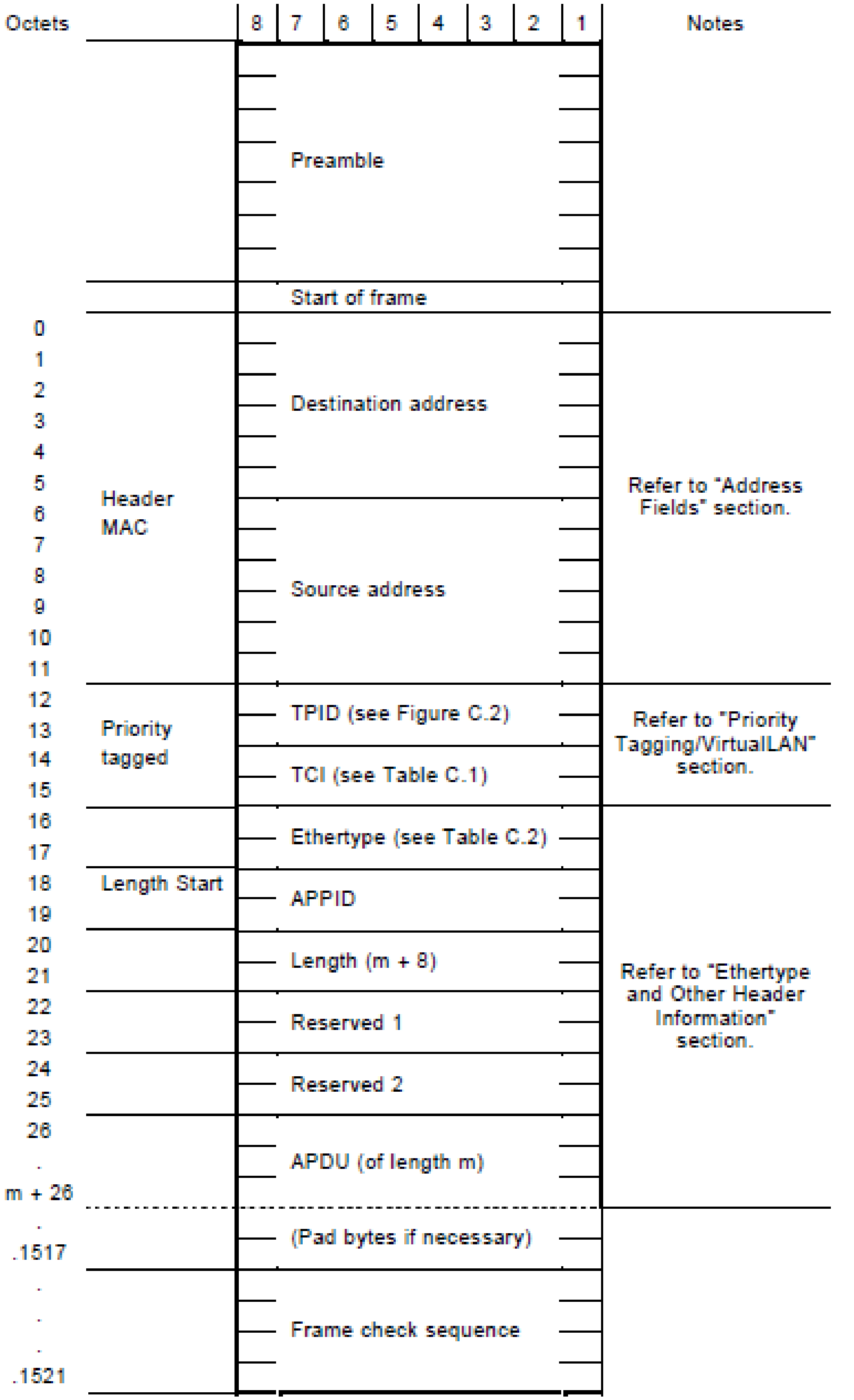


Figure 6.1: Simplified topology for case study verification

In Section 6.4, the frame structure of the GOOSE messages transmitted and received between the DK60 embedded platform and vendor IED in the "Overcurrent" case-study are analysed. The GOOSE messages are captured with the network analyser software, Wireshark. The experiment results-analysis is further extended to the GOOSE performance evaluation; device interoperability assessment; and an aspect of Cyber security. A brief overview of the results obtained for Circuit Breaker control using an MMS command sent through the TCP/IP stack is also presented as part of the results analysis.

6.2 GOOSE message structure investigation

Figure 6.2 details the frame structure of a GOOSE message (ISO/IEC 8802-3) as specified in the IEC 61850-8-page 114.



Octets		8 7 6 5 4 3 2 1	Notes
		Preamble	
		Start of frame	
0 1 2 3 4 5	Header MAC	Destination address	Refer to "Address Fields" section.
6 7 8 9 10 11		Source address	
12 13	Priority tagged	TPID (see Figure C.2)	Refer to "Priority Tagging/VirtualLAN" section.
14 15		TCI (see Table C.1)	
16 17		Ethertype (see Table C.2)	Refer to "Ethertype and Other Header Information" section.
18 19	Length Start	APPID	
20 21		Length (m + 8)	
22 23		Reserved 1	
24 25		Reserved 2	
26 . m + 26		APDU (of length m)	
. .1517		(Pad bytes if necessary)	
. . . .1521		Frame check sequence	

Figure 6.2: ISO/IEC 8802-3 frame format

(IEC 61850-8-1, 2004, p.114)

Detailed knowledge of the structure of GOOSE messages could be of importance in certain instances requiring fault diagnosis to determine the cause of mal–operation or interoperability problems; but this knowledge of message structure may also be a prerequisite to develop an IEC 61850 compliant communication stack such as the “PIS-10” from SystemCorp.

Figure 6.3 shows derivatives of the Ethernet frame. The original Ethernet frame for instance does not have a “start of frame” delimiter or an 802.1Q header.

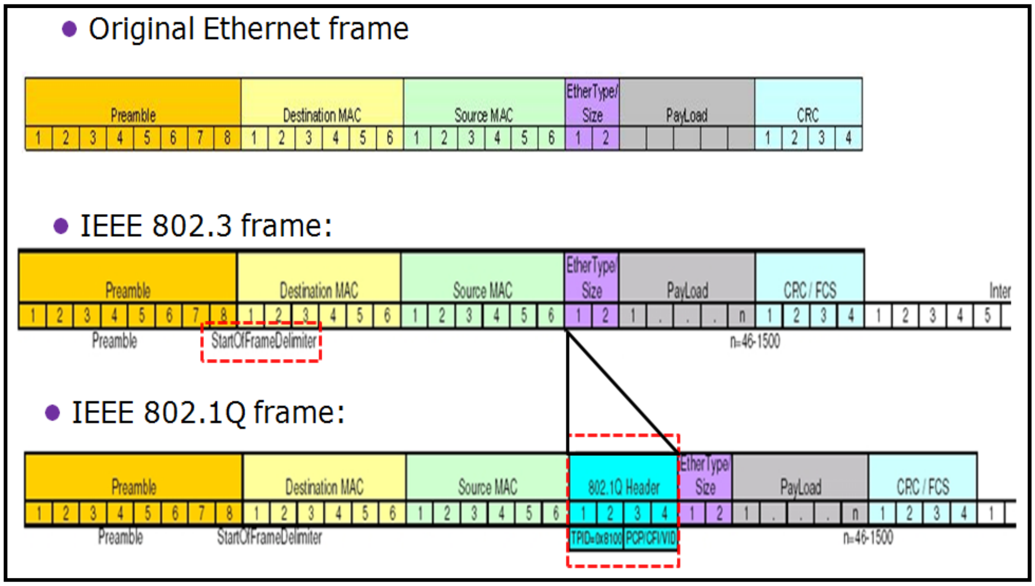


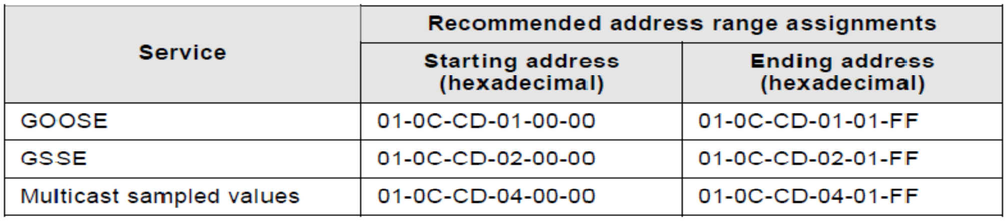
Figure 6.3: Ethernet frames specification

The IEC 61850 standard states the use of the 802.1Q Ethernet frame for VLAN-tagged messages (IEC 61850-8-1, p114), but most IEC 61850 equipment responds appropriately to non VLAN-tagged frames e.g. IEEE 802.3 frames.

[Note: for the purpose of this thesis, the term “Ethernet”, where it appears in the text - refers generically to the IEEE 802.1Q frame]

They are field within the GOOSE message structure that are standardized. Those particular fields are presented in Table 6.1 to Table 6.3 and can be found in: [IEC 61850-8-1, p 113 to p116.]

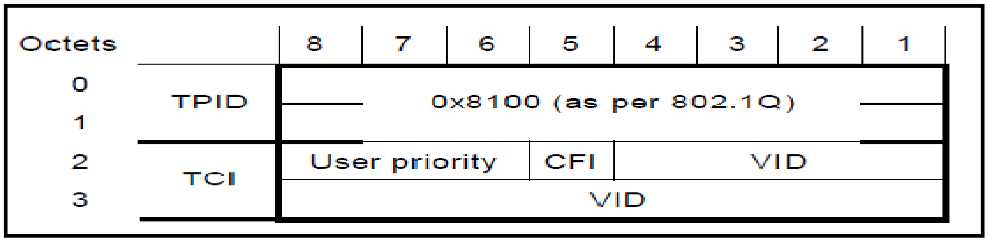
Table 6.1: Recommended multicast address range



Service	Recommended address range assignments	
	Starting address (hexadecimal)	Ending address (hexadecimal)
GOOSE	01-0C-CD-01-00-00	01-0C-CD-01-01-FF
GSSE	01-0C-CD-02-00-00	01-0C-CD-02-01-FF
Multicast sampled values	01-0C-CD-04-00-00	01-0C-CD-04-01-FF

(IEC 61850-8-1, 2004, p.113)

According to IEEE 802.1Q, **Priority tagging/Virtual LAN** is used to separate time critical and high priority bus traffic for protection-relevant applications from low priority bus load. The structure of the VLAN tag header is defined in Figure 6.4 [IEC 61850-8, Figure C.2].

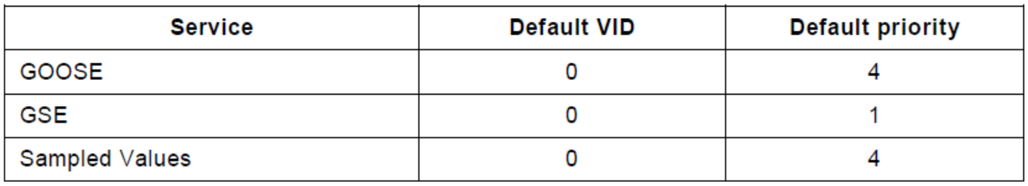


Octets		8	7	6	5	4	3	2	1
0	TPID	0x8100 (as per 802.1Q)							
1									
2	TCI	User priority			CFI	VID			
3		VID							

Figure 6.4: Virtual LAN tag
(IEC 61850-8-1, 2004, p.115)

The default virtual LAN ID and the Ethertype are presented respectively in Table 6.2 [IEC 61850-8, table C.1] and Table 6.3 [IEC 61850-8, table C.2]:

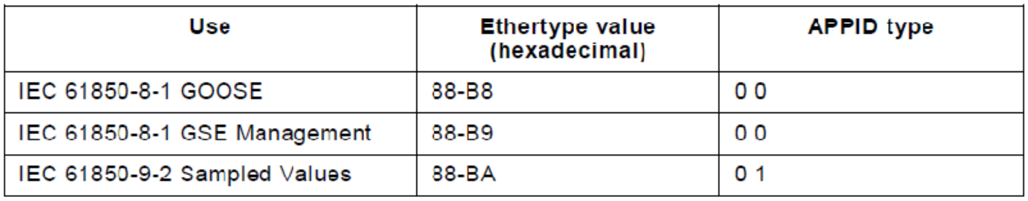
Table 6.2: Default virtual LAN IDs and priorities



Service	Default VID	Default priority
GOOSE	0	4
GSE	0	1
Sampled Values	0	4

(IEC 61850-8-1, 2004, p.115)

Table 6.3: Assigned Ethertype values

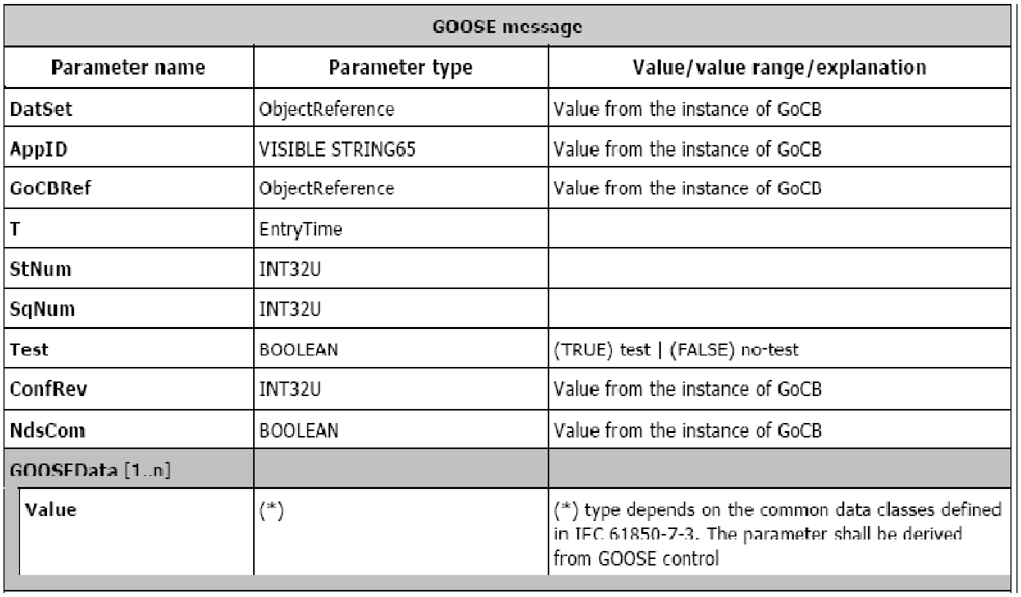


Use	Ethertype value (hexadecimal)	APPID type
IEC 61850-8-1 GOOSE	88-B8	0 0
IEC 61850-8-1 GSE Management	88-B9	0 0
IEC 61850-9-2 Sampled Values	88-BA	0 1

(IEC 61850-8-1, 2004, p.116)

The GOOSE frame section that is variable in length is known as the Application Protocol Data Unit (**APDU**) referenced from IEC 61850-7-2-p116, and defined in Table 6.4 below.

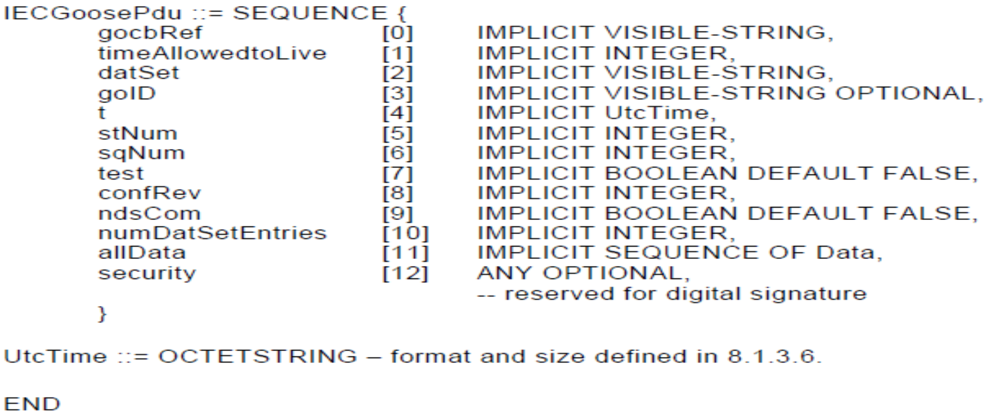
Table 6.4: GOOSE message definition (APDU)



GOOSE message		
Parameter name	**Parameter type**	**Value/value range/explanation**
DatSet	ObjectReference	Value from the instance of GoCB
AppID	VISIBLE STRING65	Value from the instance of GoCB
GoCBRef	ObjectReference	Value from the instance of GoCB
T	EntryTime	
StNum	INT32U	
SqNum	INT32U	
Test	BOOLEAN	(TRUE) test | (FALSE) no-test
ConfRev	INT32U	Value from the instance of GoCB
NdsCom	BOOLEAN	Value from the instance of GoCB
GOOSEData [1..n]		
Value	(*)	(*) type depends on the common data classes defined in IEC 61850-7-3. The parameter shall be derived from GOOSE control

(IEC 61850-7-2, 2003, p.116)

The understanding of the GOOSE message structure reported in Table 6.4 above requires a detailed knowledge of the Abstract Syntax Notation ONE (**ASN. 1**) standard for "**Data networks and open system communications**". The ASN.1 is an international standard used to define a method for encoding Data. As previously mentioned in chapter 2, information on ASN.1 with respect to the IEC 61850 standard can be found at IEC 61850-8-1, page111. The GOOSE protocol is defined using the **ASN.1 /BER** (**X.690-0207).** The following (Figure 6.5) is a part of the ASN.1 encoding rule-base for GOOSE messages (Full code in APPENDIX B) that shows the order in which each element must appear within the GOOSE.PDU (Protocol Data Unit).



```
IECGoosePdu ::= SEQUENCE {
        gocbRef                 [0]     IMPLICIT VISIBLE-STRING,
        timeAllowedtoLive       [1]     IMPLICIT INTEGER,
        datSet                  [2]     IMPLICIT VISIBLE-STRING,
        goID                    [3]     IMPLICIT VISIBLE-STRING OPTIONAL,
        t                       [4]     IMPLICIT UtcTime,
        stNum                   [5]     IMPLICIT INTEGER,
        sqNum                   [6]     IMPLICIT INTEGER,
        test                    [7]     IMPLICIT BOOLEAN DEFAULT FALSE,
        confRev                 [8]     IMPLICIT INTEGER,
        ndsCom                  [9]     IMPLICIT BOOLEAN DEFAULT FALSE,
        numDatSetEntries        [10]    IMPLICIT INTEGER,
        allData                 [11]    IMPLICIT SEQUENCE OF Data,
        security                [12]    ANY OPTIONAL,
                                        -- reserved for digital signature
        }

UtcTime ::= OCTETSTRING – format and size defined in 8.1.3.6.

END
```

Figure 6.5: GOOSE PDU definition
(IEC 61850-8-1, 2004, p.111)

[Note: In the field of Software Engineering, there are tools such as ASN1C[21] used to create an ASN.1 encode/decode function that can be called from within an application to encode or decode the data for/within a message.]

The Ethernet frame is separated into two (2) sections, which are:

- **Pre-defined fields**: These fields carry fixed data for the application for which it was designed for (represented directly by their value within the Ethernet frame – no encoding needed), and,
- **Application** Protocol Data Unit (APDU) fields: These fields carry data of variable length and the data is encoded using the ASN.1 encoding rule for GOOSE.

In the section to follow, further details with respect to ASN.1 are presented.

6.2.1 ASN.1 Encoding rule for GOOSE Protocol Data Unit (goosePDU)

The structure for a goose.PDU [APDU= Application Protocol Data Unit] is defined using the ASN.1 /BER encoding rule. Table 6.5 shows the field encoding structure in BER (ITU-T, 2002).

Table 6.5: Structure encoding in ASN.1 /BER

Tag [Identifier Octet] (1 byte)	Length (1 byte)	CONTENT	End of content (optional)

- The first byte corresponds to the **Tag (**Identifier Octet**)** and has the following syntax:

  {Class of Tag}, {Primitive or Constructed Flag} and {Tag Number}
- The second byte corresponds to the **Length**: {**length}** of the encapsulated content.
- The length is followed by the {**content}**. The content can represent data of various types (e.g. Boolean, Integer etc), or it can be another encapsulated field for APDU (Application Data Unit Protocol).

The information to follow examines in greater detail the field encoding structure.

[21] ASN1C is a full-featured ASN.1 compiler capable of generating BER/DER/CER, PER (aligned/unaligned), and XML encoders and decoders in C, C++, C#, and Java.

- **TAG definition**

As previously stated, fields contained in the “goosePDU” always start with a Tag (Identifier octet) containing the two-bit Class; the Primitive or Constructed Flag, and the Tag number. Figure 6.6 illustrates more clearly the situation.

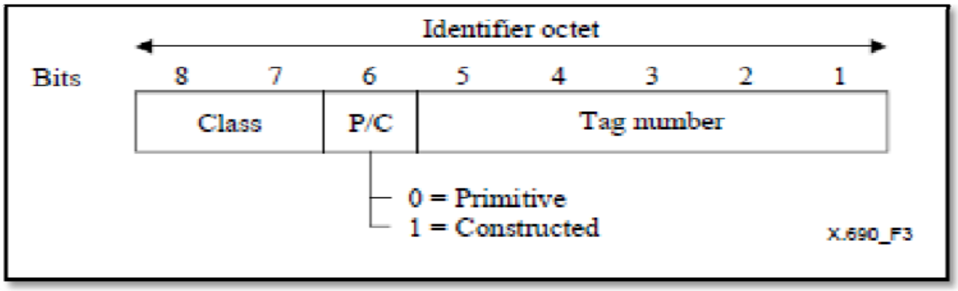
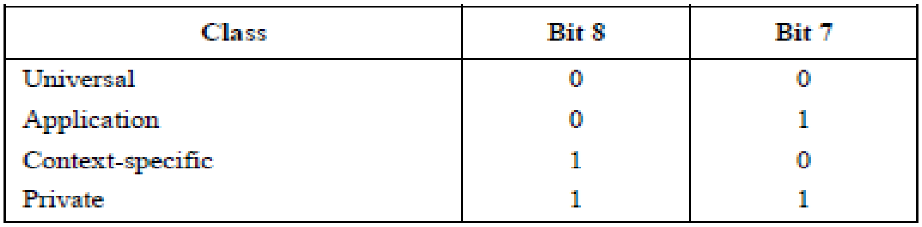


Figure 6.6: Field Identifier Octet (Tag definition)
(ITU-T, 2002)

For a “goosePDU”, the Class tag is defined as illustrated in the second row (01) of Table 6.6 (ITU-T, 2002).

Table 6.6: Encoding of class of tag



Class	Bit 8	Bit 7
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

The interpretation of Bit 8, 7 and 6 is given by Table 6.7:

Table 6.7: Encoding of class of tag and interpretation

Class of Tag: **Bit 8,7**	Primitive/ Constructed: **Bit 6**	Description, key words
0 0	0	Description: Universal Tag, Primitive Example keywords: INTEGER, BITSTRING, BOOLEAN
0 0	1	Description: Universal Tag, Constructed Example keyword: SEQUENCE, SEQUENCE OF
1 0	0	Description: Context Specific, Primitive Example Keyword: IMPLICIT
1 0	1	Description: Context Specific, Constructed Example Keywords: IMPLICIT SEQUENCE IMPLICIT SEQUENCE OF
0 1	1	Description: Application, constructed Example keywords: (None)

(SISCO, 1996)

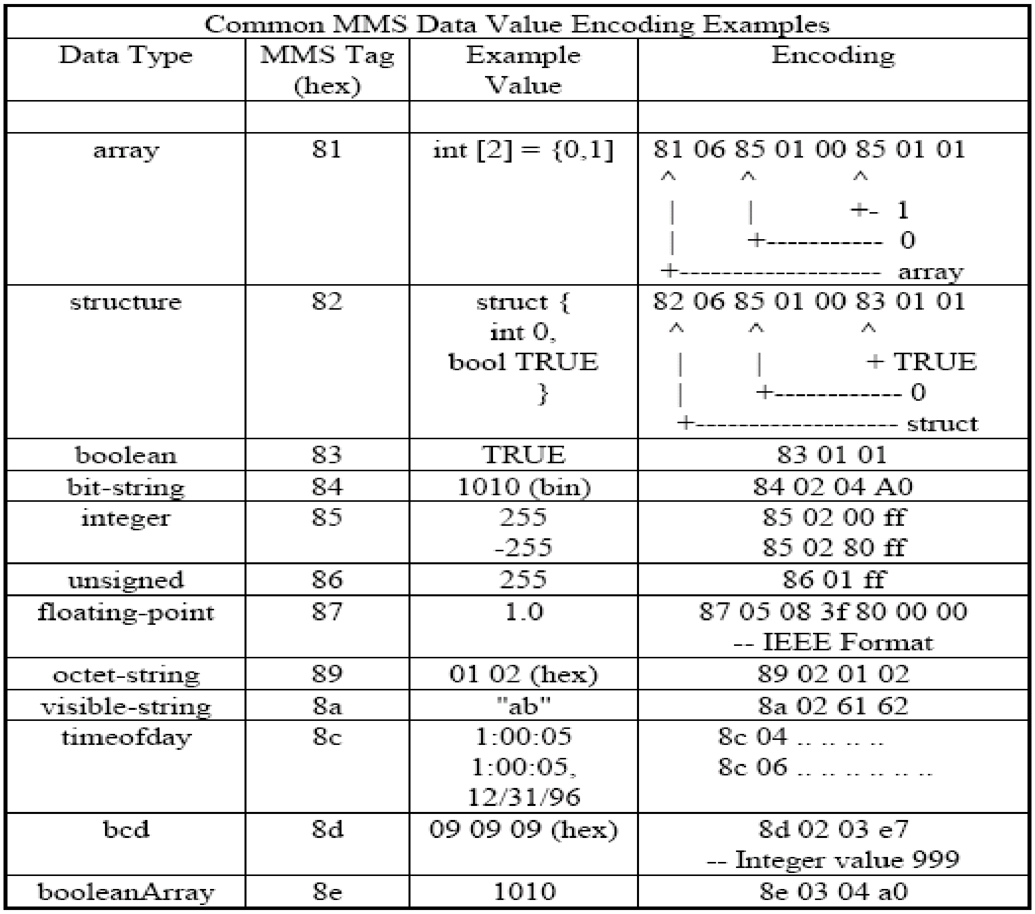
The remaining 5 least significant Bits correspond to the **Tag number** that refers to the keywords (or type of Data being encapsulated) (CESI RICERCA, 2006)**.**

[**Note**: The remaining 5 least significant Bits of tag (Tag number) have different meanings depending on the type of class.

- - In the case of the “**UNIVERSAL**” class, those bits represent a tag defined by ISO in the following table.
- - In the case of Class “**CONTEXT-SPECIFIC**”, those Bits are associated with the tag number which expresses the pointer of the “SEQUENCE” or “SET”
- - While in the case of the “**APPLICATION**” class, it distinguishes data types are which are characterized by a large range of values within a specific context. In this case the Tag number value for goosePDU is $(00001)_b$.
- In the case where the class Tag is “Context Specific” with Primitive (Keyword = IMPLICIT), those 5 Bits left out are used in association with the 3 first one to define a specific **Common GOOSE Data Value Encoding Tag**. This one derives from the MMS Data Value Encoding.]

Table 6.8 below shows the relationship between the “Data Type” and the associated “MMS Tag” column. Note: the GOOSE Value Encoding has been adopted from MMS Data Value Encoding (SISCO, 1996).

Table 6.8: Common GOOSE Data Value Encoding



Common MMS Data Value Encoding Examples			
Data Type	MMS Tag (hex)	Example Value	Encoding
array	81	int [2] = {0,1]	81 06 85 01 00 85 01 01 ^ ^ ^ | | +- 1 | +----------- 0 +------------------- array

structure	82	struct {
		int 0,
		bool TRUE
		}
			82 06 85 01 00 83 01 01
			^ ^ ^
			| | + TRUE
			| +------------ 0
			+------------------- struct

boolean	83	TRUE	83 01 01
bit-string	84	1010 (bin)	84 02 04 A0
integer	85	255 -255	85 02 00 ff 85 02 80 ff
unsigned	86	255	86 01 ff
floating-point	87	1.0	87 05 08 3f 80 00 00 -- IEEE Format
octet-string	89	01 02 (hex)	89 02 01 02
visible-string	8a	"ab"	8a 02 61 62
timeofday	8c	1:00:05 1:00:05, 12/31/96	8c 04 .. .. 8c 06 .. ..
bcd	8d	09 09 09 (hex)	8d 02 03 e7 -- Integer value 999
booleanArray	8e	1010	8e 03 04 a0

(SISCO, 1996)

- **Length definition**

The LENGTH has a simple and extended form. If the length is less than 128 bytes, a single octet is used with its MSB set to 0. If the length is greater than 128 bytes, the MSB is set to 1, and the remaining seven (7) bits represent the {length} (number of bytes) of a particular frame section (CESI RICERCA, 2006).

- **Content definition**

The value in the field must be interpreted according to the type of field. For example, if the field type is “string”, the octets must be decoded as characters, but if it is an “integer”, the value is calculated based on the binary content.

6.2.2 Details for encoding time of Data (t)

The time stamp for Data represents the moment at which the particular data has been published onto the network. This time is of type “**Coordinated Universal Time** (**UTC)** / **TimeStamp”**.

According to the IEC 61850-8-1 (2004, p132), the following specifications are acknowledged: “The UTC representation is specified in ISO 8601 as YYYY-MM-DD for the date and hh:mm:ss for the time in each day. The UTC Time type shall be an **OCTET STRING of length eight (8) octets.** The value shall be encoded as defined in **RFC 1305** (Network Time Protocol). The integer part contains: elapsed number of whole seconds since GMT midnight January 1, 1970 Sec. The fractional part contains the portion of a second elapsed since the last whole second”.

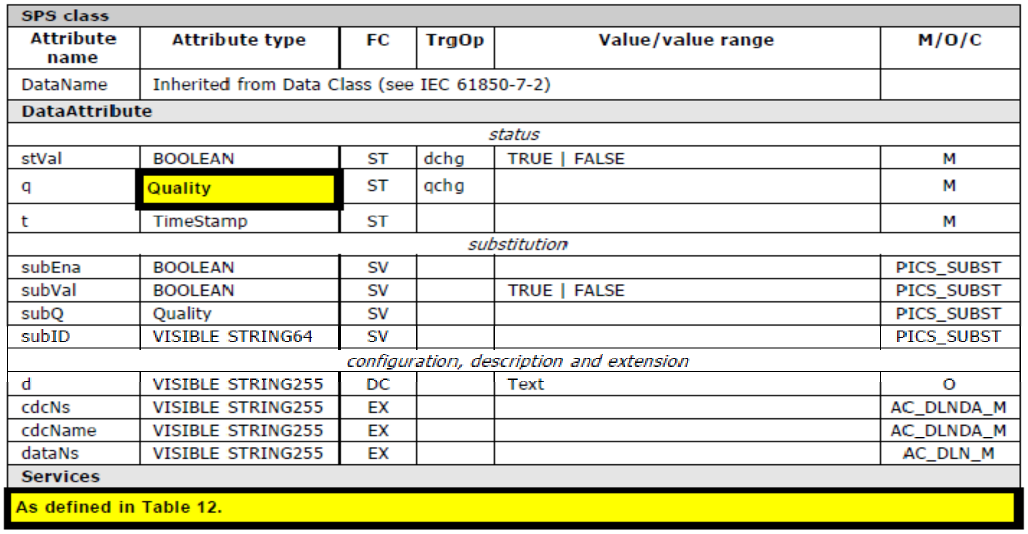
Knowing the number of seconds, one can determine how many years, months, days and time (hh: min: ss) have passed since 1st January 1970.

More details can be found in at the following sections of the IEC 61850 standard (Edition1): IEC 61850-7-1, p59; IEC 61850-7-2, p24, p154; IEC 61850-8-1, p128-133.

6.2.3 Details for encoding quality of Data (q)

It is important to publish the “Quality” of Data when exchanging messages on the network. Table 6.9 is taken from the IEC 61850-7-1/Ed1-p74 and details the composition of a Single point status common data class.

Table 6.9: Single point status common data class (SPS)

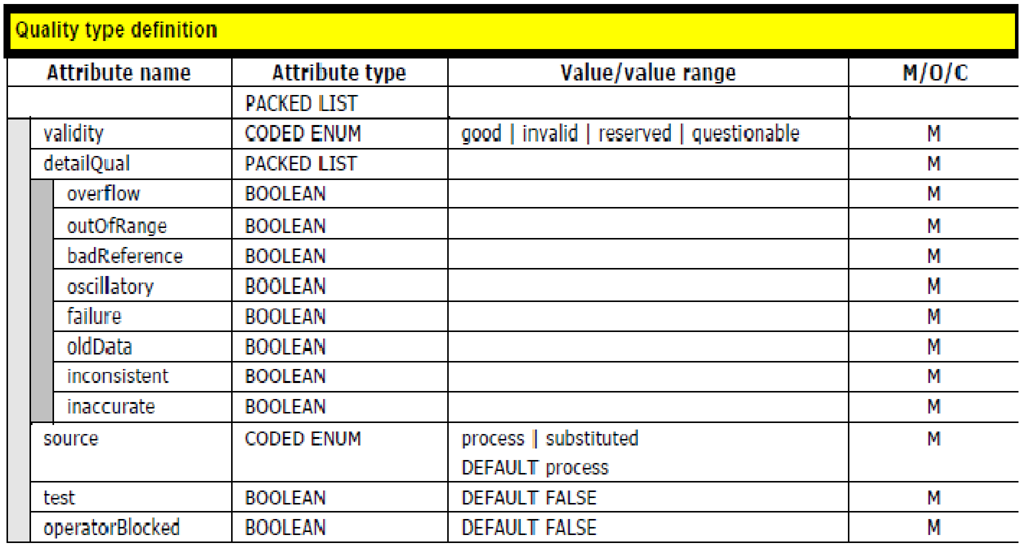


SPS class					
Attribute name	**Attribute type**	**FC**	**TrgOp**	**Value/value range**	**M/O/C**
DataName	Inherited from Data Class (see IEC 61850-7-2)				
DataAttribute					
			status		
stVal	BOOLEAN	ST	dchg	TRUE | FALSE	M
q	Quality	ST	qchg		M
t	TimeStamp	ST			M
			substitution		
subEna	BOOLEAN	SV			PICS_SUBST
subVal	BOOLEAN	SV		TRUE | FALSE	PICS_SUBST
subQ	Quality	SV			PICS_SUBST
subID	VISIBLE STRING64	SV			PICS_SUBST
			configuration, description and extension		
d	VISIBLE STRING255	DC		Text	O
cdcNs	VISIBLE STRING255	EX			AC_DLNDA_M
cdcName	VISIBLE STRING255	EX			AC_DLNDA_M
dataNs	VISIBLE STRING255	EX			AC_DLN_M
Services					
As defined in Table 12.					

(IEC 61850-7-1, 2003, p.74)

The quality is of type Bit-String (Coded Enum). The quality with respect to its mandatory subcomponent is coded over two (2) octets. From those two (2) octets there are only thirteen (13) bits that are used (Table 6.10). Thus, there is a **padding of 3 most significant bits**. Table 6.10 shows the elements that enter into the composition of the quality attribute. The quality is globally GOOD when all 13 bits are equal to zeros.

Table 6.10: Quality components attribute definition



Quality type definition			
Attribute name	**Attribute type**	**Value/value range**	**M/O/C**
	PACKED LIST		
validity	CODED ENUM	good | invalid | reserved | questionable	M
detailQual	PACKED LIST		M
overflow	BOOLEAN		M
outOfRange	BOOLEAN		M
badReference	BOOLEAN		M
oscillatory	BOOLEAN		M
failure	BOOLEAN		M
oldData	BOOLEAN		M
inconsistent	BOOLEAN		M
inaccurate	BOOLEAN		M
source	CODED ENUM	process | substituted DEFAULT process	M
test	BOOLEAN	DEFAULT FALSE	M
operatorBlocked	BOOLEAN	DEFAULT FALSE	M

(IEC 61850-7-1, 2003, p.74)

The Attribute “validity” is coded using two (2) bits, while the rest of the attribute contained in the “Quality components attribute definition” are coded using one (1) bit. This results in a total of thirteen (13) bits. So, the quality is constituted by three (3) groups of four (4) bits plus one more significant bit (i.e. to the left of the 12 bits).

The “hstring” encoding rule contained in the ASN.1 /BER is used to transmit the quality of the data attribute (ITU-T, 2002). More information on the “hstring” encoding rule can be found at: http://www.faqs.org/rfcs/rfc3641.html]

6.2.4 Introduction to IEDScout and Wireshark – for GOOSE simulation and Frame capture respectively

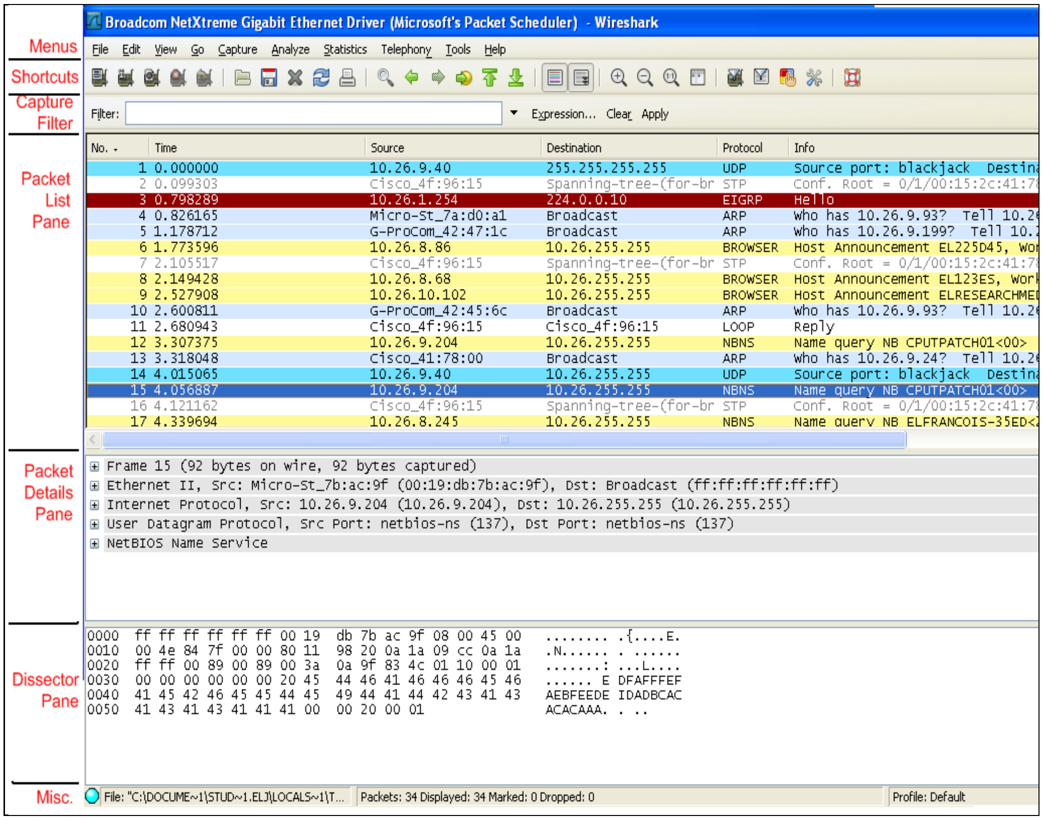
6.2.4.1 IEDScout

IEDScout is software produced by the company Omicron - one of the major vendors for Test Injection equipment (and associated products) for the power system industry. It is a universal client to IEC 61850 servers (such as substation IEDs) and a publisher/subscriber for GOOSE messages. A full product suite is provided with the Test Universe software package (also produced by Omicron). In this Thesis, IEDScout v2.11 (full version) has been used. A demo-version can also be downloaded at: https://www.omicron.at/en/content/iedscout/download-iedscout/

6.2.4.2 Wireshark

Wireshark is an open-source network protocol analyzer for UNIX and Windows systems. The latest version of Wireshark can be downloaded at http://www.wireshark.org/download.html. Wireshark uses the “pcap” network library to capture packets at the Data Link layer level.

Figure 6.7 shows a screen capture of Wireshark while “sniffing” on the local network.



```
No. -   Time        Source               Destination            Protocol  Info
      1 0.000000    10.26.9.40           255.255.255.255        UDP       Source port: blackjack  Destina
      2 0.099303    Cisco_4f:96:15       Spanning-tree-(for-br  STP       Conf. Root = 0/1/00:15:2c:41:7
      3 0.798289    10.26.1.254          224.0.0.10             EIGRP     Hello
      4 0.826165    Micro-St_7a:d0:a1    Broadcast              ARP       Who has 10.26.9.93?  Tell 10.2
      5 1.178712    G-ProCom_42:47:1c    Broadcast              ARP       Who has 10.26.9.199?  Tell 10.
      6 1.773596    10.26.8.86           10.26.255.255          BROWSER   Host Announcement EL225D45, Wo
      7 2.105517    Cisco_4f:96:15       Spanning-tree-(for-br  STP       Conf. Root = 0/1/00:15:2c:41:7
      8 2.149428    10.26.8.68           10.26.255.255          BROWSER   Host Announcement EL123ES, Wor
      9 2.527908    10.26.10.102         10.26.255.255          BROWSER   Host Announcement ELRESEARCHME
     10 2.600811    G-ProCom_42:45:6c    Broadcast              ARP       Who has 10.26.9.93?  Tell 10.2
     11 2.680943    Cisco_4f:96:15       Cisco_4f:96:15         LOOP      Reply
     12 3.307375    10.26.9.204          10.26.255.255          NBNS      Name query NB CPUTPATCH01<00>
     13 3.318048    Cisco_41:78:00       Broadcast              ARP       Who has 10.26.9.24?  Tell 10.2
     14 4.015065    10.26.9.40           10.26.255.255          UDP       Source port: blackjack  Destina
     15 4.056887    10.26.9.204          10.26.255.255          NBNS      Name query NB CPUTPATCH01<00>
     16 4.121162    Cisco_4f:96:15       Spanning-tree-(for-br  STP       Conf. Root = 0/1/00:15:2c:41:7
     17 4.339694    10.26.8.245          10.26.255.255          NBNS      Name query NB ELFRANCOIS-35ED<
```

```
⊞ Frame 15 (92 bytes on wire, 92 bytes captured)
⊞ Ethernet II, Src: Micro-St_7b:ac:9f (00:19:db:7b:ac:9f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Internet Protocol, Src: 10.26.9.204 (10.26.9.204), Dst: 10.26.255.255 (10.26.255.255)
⊞ User Datagram Protocol, Src Port: netbios-ns (137), Dst Port: netbios-ns (137)
⊞ NetBIOS Name Service
```

```
0000  ff ff ff ff ff ff 00 19  db 7b ac 9f 08 00 45 00   ........ .{....E.
0010  00 4e 84 7f 00 00 80 11  98 20 0a 1a 09 cc 0a 1a   .N...... . ......
0020  ff ff 00 89 00 89 00 3a  0a 9f 83 4c 01 10 00 01   .......: ...L....
0030  00 00 00 00 00 00 20 45  44 46 41 46 46 46 45 46   ...... E DFAFFFEF
0040  41 45 42 46 45 45 44 45  49 44 41 44 42 43 41 43   AEBFEEDE IDADBCAC
0050  41 43 41 43 41 41 41 00  00 20 00 01               ACACAAA. . ..
```

File: "C:\DOCUME~1\STUD~1.ELJ\LOCALS~1\T... | Packets: 34 Displayed: 34 Marked: 0 Dropped: 0 | Profile: Default

Figure 6.7: Wireshark capture window

A detailed description of the information represented in a Wireshark capture window can be found at: http://openmaniak.com/wireshark_use.php

The next section (6.2.5) deals with the analysis of the structure of GOOSE messages.

6.2.5 Simulation: Goose generation utilizing IEDScout, GOOSE capture using Wireshark, and structure analysis of the captured GOOSE message

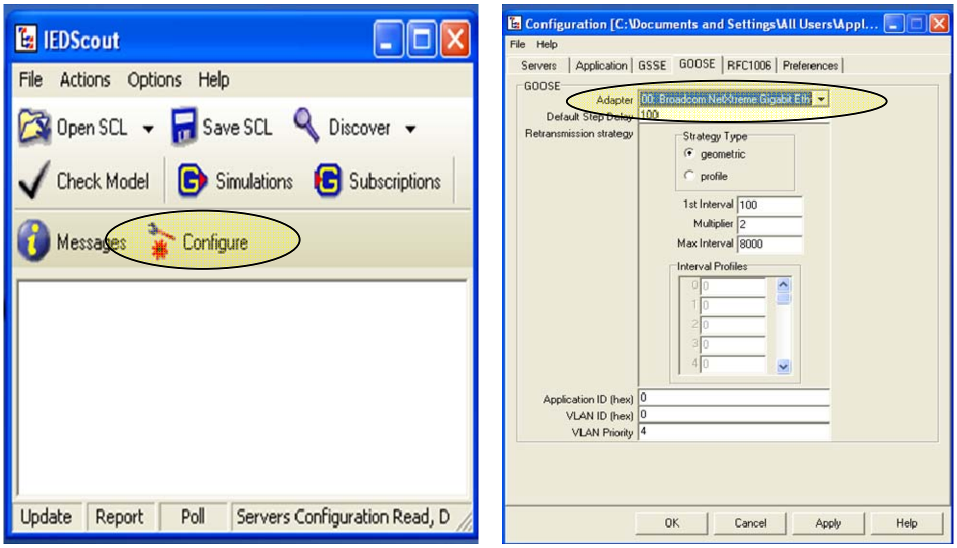
In the following Experiment IEDScout software is used to output a GOOSE message from a desktop computer. Wireshark software is then used to capture the GOOSE message frame on a second computer. The purpose of this experiment is to confirm the structure of the **GOOSE message frame** with respect to the one specified in IEC 61850-8- page114 and also to identify the content of the **GOOSE message** with respect to the one defined in IEC 61850-7-2 page 116 -Table 29.

The following simulation is described utilizing **Wireshark V1.2.8** and **IEDScout V2.10** on a Windows XP station.

6.2.5.1 Setting up IEDScout for GOOSE simulation

Start IEDScout (Figure 6.8) and go to "**Configure**", then select the "GOOSE" tab. On the GOOSE tab specify which Network Adapter you want to use (for first utilisation it is required to restart IEDScout).

Note that in order to generate GOOSE messages through the Ethernet adapter; it needs to be ensured **that an Ethernet cable is plugged into the RJ45 adapter**.



Servers Configuration Read, D

Configuration [C:\Documents and Settings\All Users\Appl...

Adapter 00: Broadcom NetXtreme Gigabit Eth

Default Step Delay 100

Retransmission strategy

Strategy Type

● geometric

○ profile

1st Interval 100

Multiplier 2

Max Interval 8000

Interval Profiles

0	0
1	0
2	0
3	0
4	0

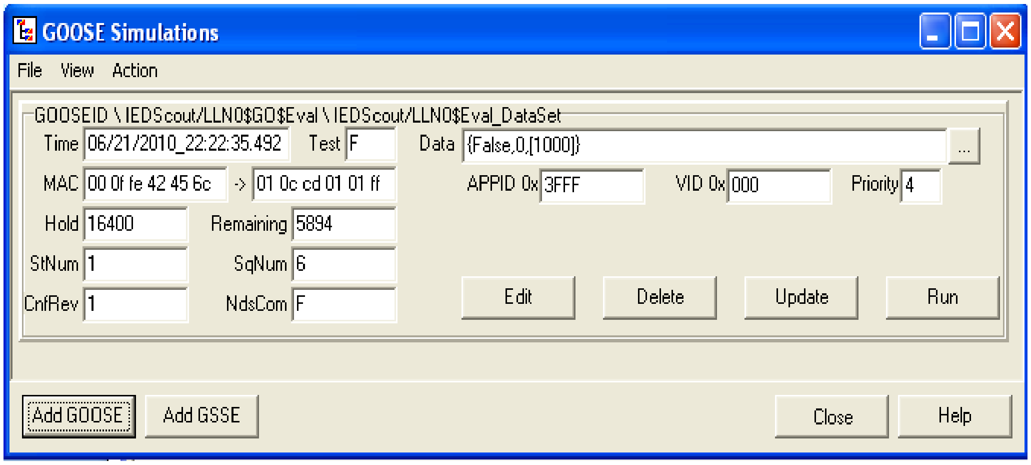
Application ID (hex) 0

VLAN ID (hex) 0

VLAN Priority 4

Figure 6.8: Configuration of IEDScout

On the Menu bar click "Simulations". Then click **Add GOOSE,** in window which opens up. The "GOOSE Simulations" window now appears (Figure. 6.9 below), wherein certain key parameters can be edited.



GOOSEID \ IEDScout/LLN0$GO$Eval \ IEDScout/LLN0$Eval_DataSet

Time 06/21/2010_22:22:35.492 Test F Data {False,0,[1000]}

MAC 00 0f fe 42 45 6c -> 01 0c cd 01 01 ff APPID 0x 3FFF VID 0x 000 Priority 4

Hold 16400 Remaining 5894

StNum 1 SqNum 6

CnfRev 1 NdsCom F

Figure 6.9: Setting parameter for GOOSE simulation

There are now GOOSE messages being transmitted on the Ethernet bus through the Network adapter card.

6.2.5.2 Setting up Wireshark

After successful installation of Wireshark, select a network interface and start a capture. GOOSE messages that are being transmitted on the network should be observed.

For more details on how to start a capture on Wireshark please consult the online manual at: http://openmaniak.com/wireshark.php

6.2.5.3 GOOSE composition

Using IEDScout, a specific “DataSet” can be defined for the simulation of the GOOSE message to be transmitted onto the network. By knowing the “DataSet” structure which had been specified, and the general structure for the GOOSE message as defined within the IEC61850 standard, one can predict exactly what the GOOSE message is going to be composed of in its different header sections. The expected GOOSE content is reported on in Table 6.11 (**NB**: hexadecimal format applies with 1 octet = 1 byte = 8 bits).

Table 6.11: Expected experimental GOOSE frame

	SECTION				VALUE			COMMENT and REF
Preamble	Preamble (7 bytes)				10101010			The preamble consists of 62 bits of alternating ones and zeros that allows the Ethernet card to Synchronize with the beginning of a frame.
Start of Frame	Start of Frame (1 byte)				10101011			The Start Frame Delimiter.
Header MAC [0->11] bytes	Address fields	Destination address (6 bytes)			01 0c cd 01 01 ff			This is a broadcast/ multicast address. IEC 61850-8-p115-113
		Source address (6 bytes)			00 0f fe 42 45 6c			No information within the standard. Left as local issue for Vendor.
Priority tagged /Virtual LAN [12->15] **bytes**	TPID				0x8100			Tag protocol Identifier. IEC 61850-8-p115
	TCI	User Priority	CFI	VID	4	0	0	Tag control information IEC 61850-8-p115
Ethertype and other Header [m+26] **bytes**	Ethertype				88-B8			IEC 61850-8-p115 -116
	APPID				0x3fff			IEC 61850-8-p116
	Length (m+8)				Not visible			IEC 61850-8-p116
	Reserved 1				0			IEC 61850-8-p116
	Reserved 2				0			IEC 61850-8-p116
	APDU (of length m)	gocbRef			IEDScout/LLN0$GO$Eval			This is the GOOSE-PDU (Protocol Data unit). Content of the GOOSE message. IEC 61850-8-p116 and IEC 61850-7-2-p116
		timeAllowedtoLive			(100+200)* (2^ sqNum)			
		datSet			IEDScout/LLN0$Eval_DataSet			
		goID			GOOSEID			
		t			xx			
		stNum			1			
		sqNum			xx			
		test			F			
		confRev			1			
		ndsCom			F			
		numDatSetEntries			3			
		allData			{False,0,[1000]}			
[1517] **bytes**	(Pad bytes if necessary)				-			
[1521] **bytes**	Frame check sequence				-			Generated/ Clear at a hardware level

xx- values which change with time

6.2.5.4 GOOSE capture and confirmation

With IEDScout a GOOSE message is now published onto the Network. The objective of this simulation is to allow for verification of the structure and content of the captured GOOSE with respect to the one defined earlier in Section 6.2.5.3.

[Note: The simulation can be generated (by IEDScout) and captured (by Wireshark) on a single PC, but it is recommended to do the simulation with two computers connected via an Ethernet switch as illustrated in Figure 6.10 below:]

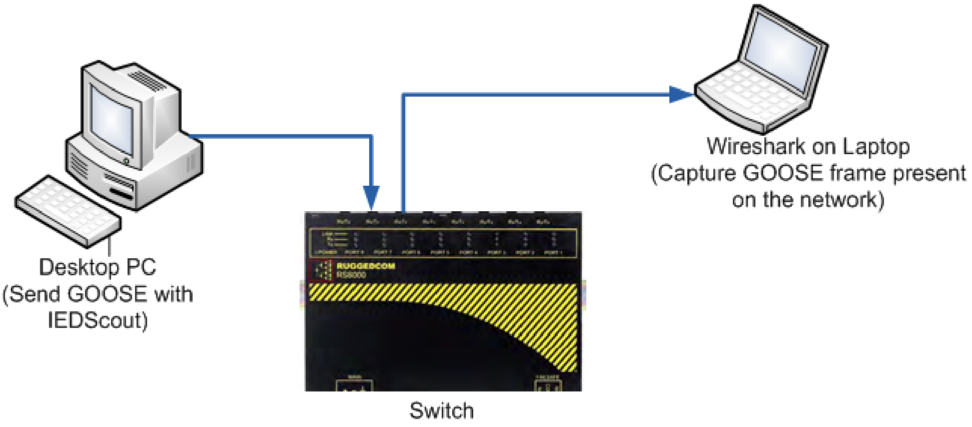
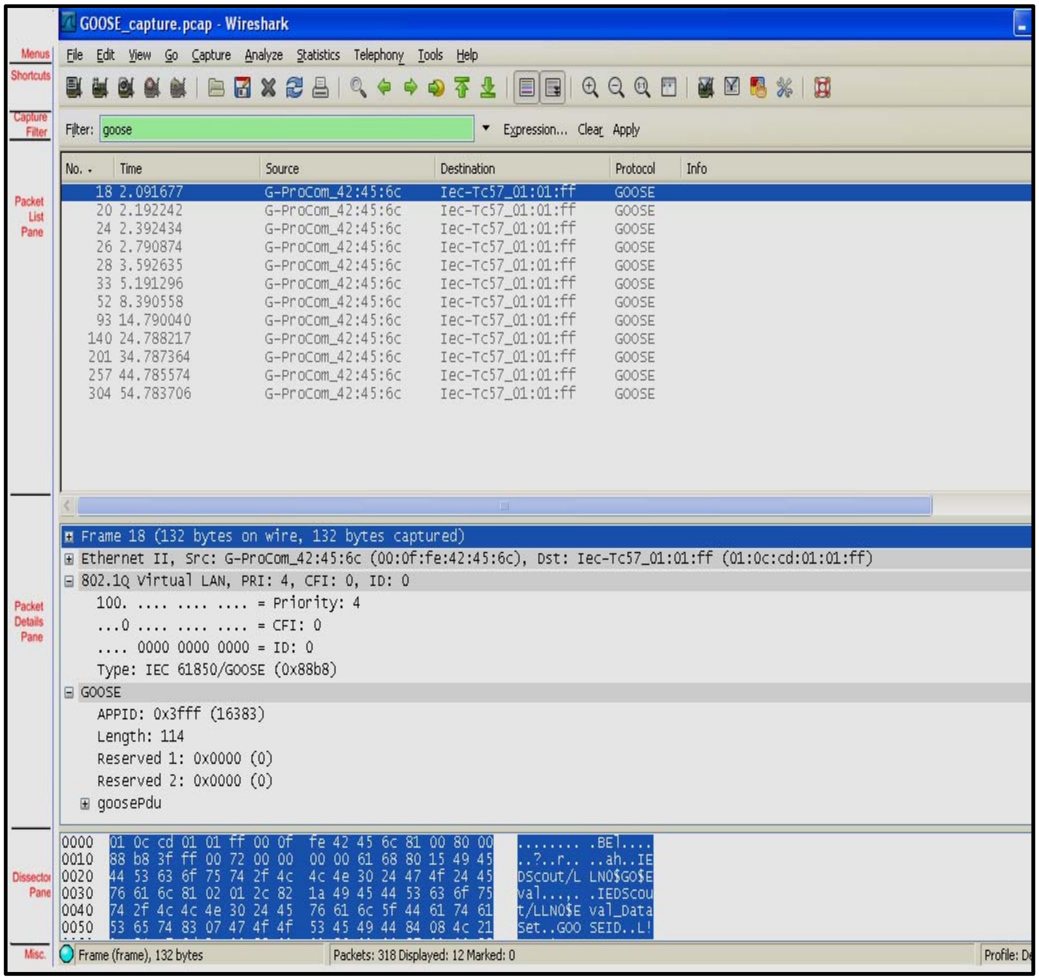


Figure 6.10: Topology for GOOSE simulation

You may now start the Capture on Wireshark and then activate the simulation with IEDScout. Figure 6.11 presents the result obtained.


GOOSE_capture.pcap - Wireshark
Menus
Shortcuts
Capture Filter
Packet List Pane
Packet Details Pane
Dissector Pane
Misc.
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Expression... Clear Apply

Filter: goose

```
No. -  Time       Source                Destination          Protocol  Info
   18 2.091677    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   20 2.192242    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   24 2.392434    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   26 2.790874    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   28 3.592635    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   33 5.191296    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   52 8.390558    G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
   93 14.790040   G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
  140 24.788217   G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
  201 34.787364   G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
  257 44.785574   G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
  304 54.783706   G-ProCom_42:45:6c     Iec-Tc57_01:01:ff    GOOSE
```

```
⊞ Frame 18 (132 bytes on wire, 132 bytes captured)
⊞ Ethernet II, Src: G-ProCom_42:45:6c (00:0f:fe:42:45:6c), Dst: Iec-Tc57_01:01:ff (01:0c:cd:01:01:ff)
⊟ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 0
    100. .... .... .... = Priority: 4
    ...0 .... .... .... = CFI: 0
    .... 0000 0000 0000 = ID: 0
    Type: IEC 61850/GOOSE (0x88b8)
⊟ GOOSE
    APPID: 0x3fff (16383)
    Length: 114
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊞ goosePdu
```

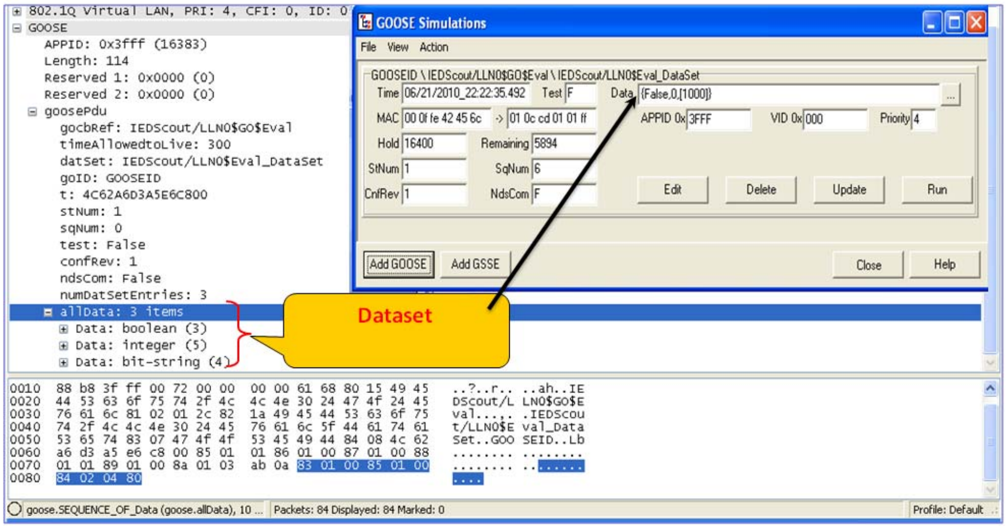
```
0000  01 0c cd 01 01 ff 00 0f  fe 42 45 6c 81 00 80 00   ........ .BEl....
0010  88 b8 3f ff 00 72 00 00  00 00 61 68 80 15 49 45   ..?..r.. ..ah..IE
0020  44 53 63 6f 75 74 2f 4c  4c 4e 30 24 47 4f 24 45   DScout/L LN0$GO$E
0030  76 61 6c 81 02 01 2c 82  1a 49 45 44 53 63 6f 75   val...,. .IEDScou
0040  74 2f 4c 4c 4e 30 24 45  76 61 6c 5f 44 61 74 61   t/LLN0$E val_Data
0050  53 65 74 83 07 47 4f 4f  53 45 49 44 84 08 4c 21   Set..GOO SEID..L!
```

Frame (frame), 132 bytes | Packets: 318 Displayed: 12 Marked: 0 | Profile: De

Figure 6.11: GOOSE simulation result

Result - The observed GOOSE frame is compliant with the specified frame structure given in at Figure 6.2, page 156 (IEC 61850-8-1, p 111-114).

On the Wireshark "packet detail pane" all expected components are present (Figure 6.12), with respect to the parameters originally set up in the dialogue box presented in Figure 6.9.



```
⊞ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 0
⊟ GOOSE
    APPID: 0x3fff (16383)
    Length: 114
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: IEDScout/LLN0$GO$Eval
      timeAllowedtoLive: 300
      datSet: IEDScout/LLN0$Eval_DataSet
      goID: GOOSEID
      t: 4C62A6D3A5E6C800
      stNum: 1
      sqNum: 0
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 3
    ⊟ allData: 3 items
      ⊞ Data: boolean (3)
      ⊞ Data: integer (5)
      ⊞ Data: bit-string (4)
```

```
0010  88 b8 3f ff 00 72 00 00  00 00 61 68 80 15 49 45   ..?..r.. ..ah..IE
0020  44 53 63 6f 75 74 2f 4c  4c 4e 30 24 47 4f 24 45   DScout/L LN0$GO$E
0030  76 61 6c 81 02 01 2c 82  1a 49 45 44 53 63 6f 75   val..,.. .IEDScou
0040  74 2f 4c 4c 4e 30 24 45  76 61 6c 5f 44 61 74 61   t/LLN0$E val_Data
0050  53 65 74 83 07 47 4f 4f  53 45 49 44 84 08 4c 62   Set..GOO SEID..Lb
0060  a6 d3 a5 e6 c8 00 85 01  01 86 01 00 87 01 00 88   ........ ........
0070  01 01 89 01 00 8a 01 03  ab 0a 83 01 00 85 01 00   ........ ........
0080  84 02 04 80                                        ....
```

goose.SEQUENCE_OF_Data (goose.allData), 10 ... | Packets: 84 Displayed: 84 Marked: 0 | Profile: Default

GOOSE Simulations

GOOSEID \ IEDScout/LLN0$GO$Eval \ IEDScout/LLN0$Eval_DataSet

Time 06/21/2010_22:22:35.492 Test F Data {False,0,[1000]}

MAC 00 0f fe 42 45 6c -> 01 0c cd 01 01 ff APPID 0x 3FFF VID 0x 000 Priority 4

Hold 16400 Remaining 5894

StNum 1 SqNum 6

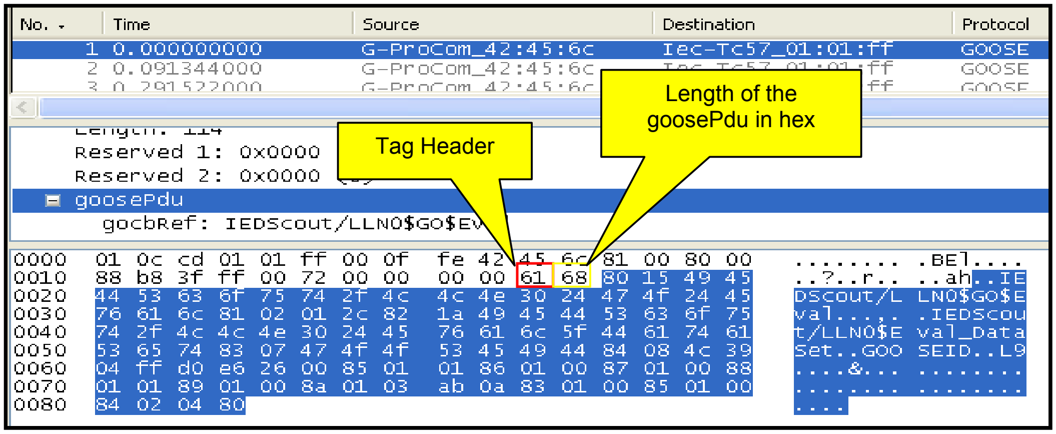
CnfRev 1 NdsCom F

Figure 6.12: Identification of dataset used

[**Note**: some hexadecimal values that are contained in the frame are defined in accordance with the ASN.1 encoding rule previously mentioned in section 6.2.1. The following section gives a better understanding of how to read and interpret those hexadecimal values.]

6.2.5.5 Identification and Interpretation of hexadecimal value

In this sub-section the hexadecimal values obtained by means of the simulation conducted with IEDScout and captured with Wireshark (Figure 6.13) are identified and interpreted. The result is compared to what was expected as per that specified in the IEC61850 standard (see section 6.2.5), and further details regarding the goosePDU encoding using ASN.1-BER are explored.



```
No.      Time            Source               Destination          Protocol
    1 0.000000000     G-ProCom_42:45:6c    Iec-Tc57_01:01:ff    GOOSE
    2 0.091344000     G-ProCom_42:45:6c    Iec-Tc57_01:01:ff    GOOSE
    3 0.291522000     G-ProCom_42:45:6c                 ff      GOOSE

    Length: 114
    Reserved 1: 0x0000
    Reserved 2: 0x0000
  ⊟ goosePdu
      gocbRef: IEDScout/LLN0$GO$Ev

0000  01 0c cd 01 01 ff 00 0f  fe 42 45 6c 81 00 80 00   ........ .BEl....
0010  88 b8 3f ff 00 72 00 00  00 00 61 68 80 15 49 45   ..?..r.. ..ah..IE
0020  44 53 63 6f 75 74 2f 4c  4c 4e 30 24 47 4f 24 45   DScout/L LN0$GO$E
0030  76 61 6c 81 02 01 2c 82  1a 49 45 44 53 63 6f 75   val..... .IEDScou
0040  74 2f 4c 4c 4e 30 24 45  76 61 6c 5f 44 61 74 61   t/LLN0$E val_Data
0050  53 65 74 83 07 47 4f 4f  53 45 49 44 84 08 4c 39   Set..GOO SEID..L9
0060  04 ff d0 e6 26 00 85 01  01 86 01 00 87 01 00 88   ....&... ........
0070  01 01 89 01 00 8a 01 03  ab 0a 83 01 00 85 01 00   ........ ........
0080  84 02 04 80                                        ....
```

Tag Header

Length of the goosePdu in hex

Figure 6.13: Interpretation of Hexadecimal value

With respect to the data displayed in frame in Figure 6.13, Table 6.12 below now gives the meaning of each byte of the **GOOSE**-**P**rotocol **D**ata **U**nit starting at the first tag header (Tag 61 as per Figure 6.13) and proceeding octet by octet through the data in the payload.

Table 6.12: Interpretation of hexadecimal value for GOOSE PDU

Value octet (Hexadecimal)	**Value octet** (Binary)	**Meaning (Interpretation)**	**Decimal (bytes)**
61	01100001	Class APPLICATION, Composition of data types Tag number 1. Identifier octet. Fieldname “goosePdu”	
68	01101000	Length of “goosePdu”	104
80	10000000	Class CONTEXT-SPECIFIC, Primitive, tag number [0], IMPLICIT VISIBLE-STRING, Identifier octet. Fieldname “goose.gocbRef”	
15	00010101	Length of “goose.gocbRef”	21
49 45 44 53 63 6f 75 74 2F 4C 4C 4E 30 24 47 4F 24 45 76 61 6C		Content of “goose.gocbRef” {IEDScout/LLN0$GO$Eval}	
81	10000001	Class CONTEXT-SPECIFIC, Primitive, tag number [1], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.timeAllowedtoLive”	
02	00000010	Length of “goose.timeAllowedtoLive”	2
01 2C	00000001 00101100	Content of “goose.timeAllowedtoLive” { 300 }	
82	10000010	Class CONTEXT-SPECIFIC, Primitive, tag number [2], IMPLICIT VISIBLE-STRING, Identifier octet. Fieldname “goose.datSet”	
1A	00011010	Length of “goose.datSet”	26
49 45 44 53 63 6F 75 74 2F 4c 4c 4e 30 24 45 76 61 6c 5f 44 61 74 61 53 65 74		Content of “goose.datSet” { IEDScout/LLN0$Eval_DataSet }	
83	10000011	Class CONTEXT-SPECIFIC, Primitive, tag number [3], IMPLICIT VISIBLE-STRING OPTIONAL, Identifier octet. Fieldname “goose.goID”	
07	00000111	Length of “goose.goID”	7
47 4f 4f 53 45 49 44		Content of “goose.goID”, { GOOSEID}	

Value octet (Hexadecimal)	**Value octet** (Binary)	**Meaning (Interpretation)**	**Decimal (bytes)**
84	10000100	Class CONTEXT-SPECIFIC, Primitive, tag number [4], IMPLICIT UtcTime, Identifier octet. Fieldname “goose.t”	
08	00001000	Length of “goose.t”	8
4c 39 04 ff d0 e6 26 00		Content of “goose.t”, { 4c:39:04:ff:d0:e6:26:00 }	
85	10000101	Class CONTEXT-SPECIFIC, Primitive, tag number [5], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.stNum”	
01	00000001	Length of “goose.stNum”	1
01	00000001	Content of “goose.stNum”, { 1 }	
86	10000110	Class CONTEXT-SPECIFIC, Primitive, tag number [6], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.sqNum”	
01	00000001	Length of “goose.sqNum”	1
00	00000000	Content of “goose.sqNum”, { 0 }	
87	10000111	Class CONTEXT-SPECIFIC, Primitive, tag number [7], IMPLICIT BOOLEAN DEFAULT FALSE, Identifier octet. Fieldname “goose.test”	
01	00000001	Length of “goose.test”	1
00	00000000	Content of “goose.test”, { 0 }	
88	10001000	Class CONTEXT-SPECIFIC, Primitive, tag number [8], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.confRev”	
01	00000001	Length of “goose.confRev”	1
01	00000001	Content of “goose.confRev”, { 1 }	
89	10001001	Class CONTEXT-SPECIFIC, Primitive, tag number [9], IMPLICIT BOOLEAN DEFAULT FALSE, Identifier octet. Fieldname “goose.ndsCom”	
01	00000001	Length of “goose.ndsCom”	1
00	00000000	Content of “goose.ndsCom”, { 0 }	
8a	10001010	Class CONTEXT-SPECIFIC, Primitive, tag number [10], IMPLICIT INTEGER, Identifier octet. Fieldname “goose.numDatSetEntries”	
01	00000001	Length of “goose.numDatSetEntries”	1
03	00000011	Content of “goose.numDatSetEntries”, { 3 }	3
Ab	10101011	Class CONTEXT-SPECIFIC, Constructor, tag number [11], IMPLICIT SEQUENCE OF Data, Identifier octet. Fieldname “goose.allData”	
0a	00001010	Length of “goose.allData”	10
83	10000011	Class CONTEXT-SPECIFIC, Primitive, Boolean (3) Identifier octet. Fieldname “Data: Boolean”	
01	00000001	Length of “Data: boolean”	1
00	00000000	Content of “Data: boolean ”, { 0 }	
85	10000101	Class CONTEXT-SPECIFIC, Primitive, Integer (5) Identifier octet. Fieldname “Data: integer”	
01	00000001	Length of “Data: integer ”	1
00	00000000	Content of “Data: integer ”, {0}	
84	10000100	Class CONTEXT-SPECIFIC, Primitive, bit-string (4) Identifier octet. Fieldname “Data: bit-string” “ber.bitstring.padding” “goose.bit_string”	
02	00000010	Length of “Data: bit-string”	2

04 80	00000100
	10000000
		Content of “Data.Padding”, {4}
		Content of “Data: bit-string”, { 80 }	

After the simulation with IEDScout of a GOOSE message in order to study its structure, it was also important to verify that the structure of GOOSE messages published by commercial IEDs used in this project conformed to the IEC 61850 standard. The next section expands on the details of the experiment set up, and the results obtained.

6.2.6 Experimentation: GOOSE generation using commercial (SIEMENS) Relays, and capture with Wireshark

The experiment introduces an example of how to quickly set up a basic Publish/Subscribe application for GOOSE messaging. The DIGSI (v4) is a software platform that allows for SIEMENS Device configuration and substation communication architecture design. It is provided together with the purchase of new SIEMENS relay devices. This experiment is derived from the SIEMENS user guide: "**Ethernet & IEC 61850-Start up"** (Release: 07.05.09. E50417-F1176-C324-A3). The user guide can be downloaded at:

http://**siemens**.siprotec.de/download_neu/.../**IEC61850_STARTUP**_A3_EN.pdf

6.2.6.1 Experiment objectives

The structure and content of a GOOSE message has already been indentified and confirmed in a simulated environment using IEDScout and Wireshark (section 6.2.5). In this section, the comparison of simulated information to that of a GOOSE message generated through experimentation with commercial IEDs is made.

6.2.6.2 Experiment description

In this experiment, the state (ON/OFF) of an LED-1 is controlled on 3 separate SIEMENS SIPROTEC devices. This is accomplished via the development of a Publisher/Subscriber application based on GOOSE messaging. In the case of this experiment, the first IED is assumed to be the publisher of a GOOSE message, while the two others are subscribing to the GOOSE message.

The logical connections for the experiment are shown in Figure 6.14.

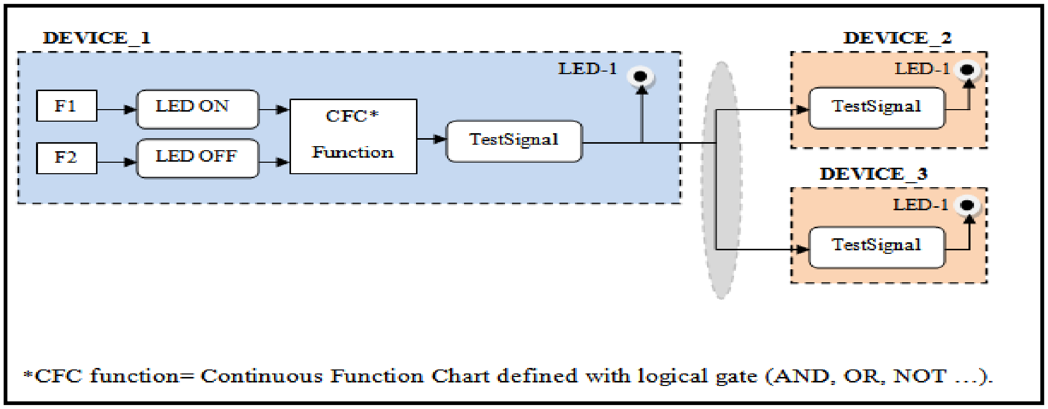


Figure 6.14: Logical connection for experimentation with IEDs
(Siemens, 2009)

Figure 6.15 presents the topology (bus) adopted:

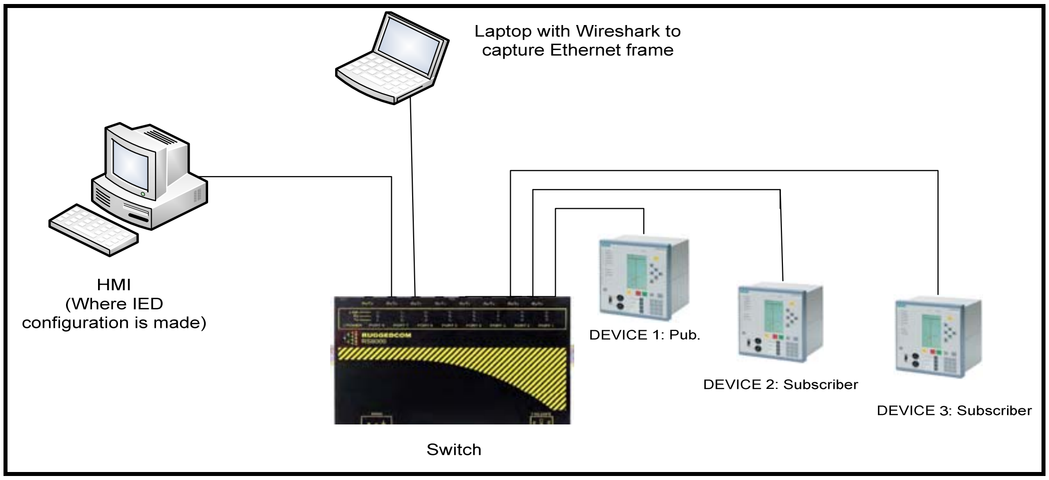


Figure 6.15: Network Topology for experimentation with IEDs

Figure 6.16 illustrates the communication test which begins by pressing the F1 key on Device 1. The illumination of LED-1 on the same device confirms this key-press. What is important is that an indication (via GOOSE) is sent also to the two other subscribing devices. LED-1 on each of these two devices is then turned ON to signal that the devices have received the indication.

Pressing the F2 Key on Device 1 generates a GOOSE message which carries Boolean information which cancels the indication on the publishing device and on the LEDs on all the subscribing devices.

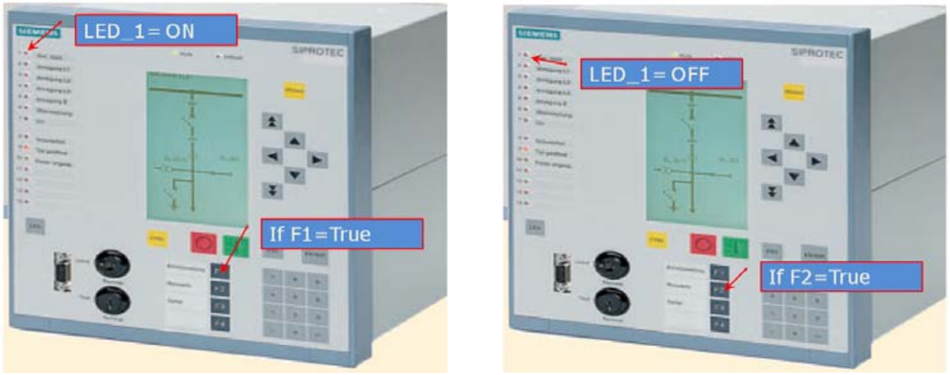
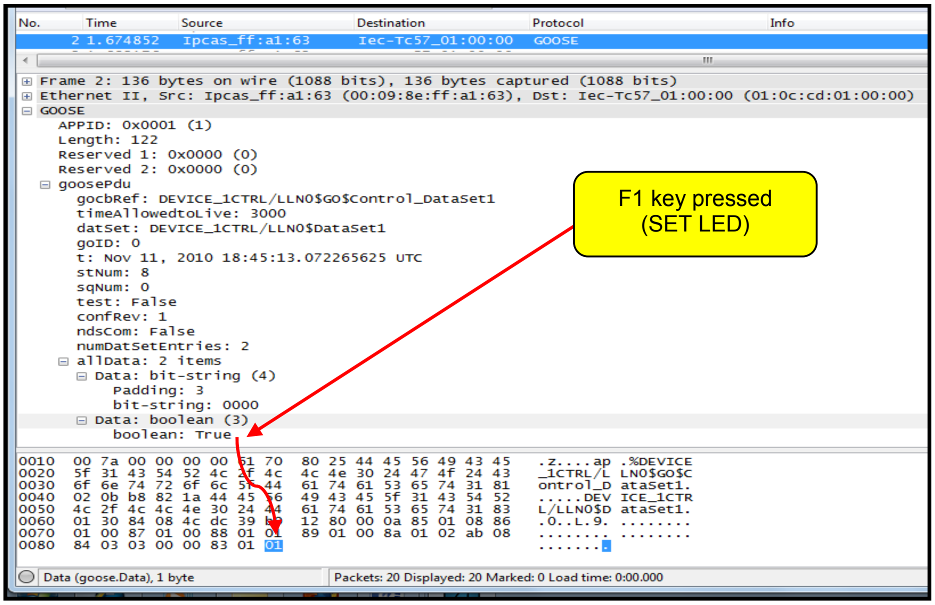


Figure 6.16: Illustration of input Key into LED

Only the results of the experiment are presented here. For complete details on how to set up the Relays, refer to the SIEMENS user guide: "**Ethernet & IEC61850 start up**" at [http://**siemens**.siprotec.de/download_neu/.../**IEC61850_STARTUP**_A3_EN.pdf](http://siemens.siprotec.de/download_neu/.../IEC61850_STARTUP_A3_EN.pdf)

6.2.6.3 Results analysis

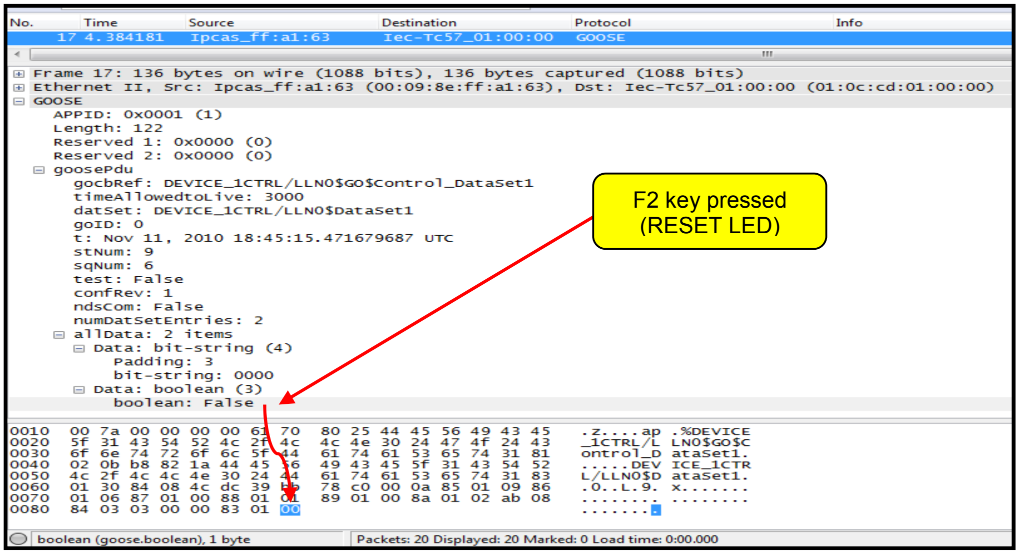
Figure 6.17 is a Wireshark capture of the GOOSE message published on the network when pressing the F1 key (SET LED). The Boolean Data that controls the state of LED1 is carried within the GOOSE message with the value True {01} as shown below.



```
No.     Time        Source            Destination         Protocol      Info
      2 1.674852    Ipcas_ff:a1:63    Iec-Tc57_01:00:00   GOOSE

⊞ Frame 2: 136 bytes on wire (1088 bits), 136 bytes captured (1088 bits)
⊞ Ethernet II, Src: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63), Dst: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
⊟ GOOSE
    APPID: 0x0001 (1)
    Length: 122
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: DEVICE_1CTRL/LLN0$GO$Control_DataSet1
      timeAllowedtoLive: 3000
      datSet: DEVICE_1CTRL/LLN0$DataSet1
      goID: 0
      t: Nov 11, 2010 18:45:13.072265625 UTC
      stNum: 8
      sqNum: 0
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 2
    ⊟ allData: 2 items
      ⊟ Data: bit-string (4)
          Padding: 3
          bit-string: 0000
      ⊟ Data: boolean (3)
          boolean: True

0010   00 7a 00 00 00 00 61 70  80 25 44 45 56 49 43 45   .z....ap .%DEVICE
0020   5f 31 43 54 52 4c 2f 4c  4c 4e 30 24 47 4f 24 43   _1CTRL/L LN0$GO$C
0030   6f 6e 74 72 6f 6c 5f 44  61 74 61 53 65 74 31 81   ontrol_D ataSet1.
0040   02 0b b8 82 1a 44 45 56  49 43 45 5f 31 43 54 52   .....DEV ICE_1CTR
0050   4c 2f 4c 4c 4e 30 24 44  61 74 61 53 65 74 31 83   L/LLN0$D ataSet1.
0060   01 30 84 08 4c dc 39 b[illegible]  12 80 00 0a 85 01 08 86   .0..L.9. ........
0070   01 00 87 01 00 88 01 01  89 01 00 8a 01 02 ab 08   ........ ........
0080   84 03 03 00 00 83 01 01                            ........

○ Data (goose.Data), 1 byte     Packets: 20 Displayed: 20 Marked: 0 Load time: 0:00.000
```

Figure 6.17: Experiment results (a)

Figure 6.18 is a Wireshark capture of the GOOSE message published on the network when F2 (RESET LED) key is pressed. The Boolean Data that controls the state of LED1 is carried within the GOOSE message with the value False {00} as shown below.

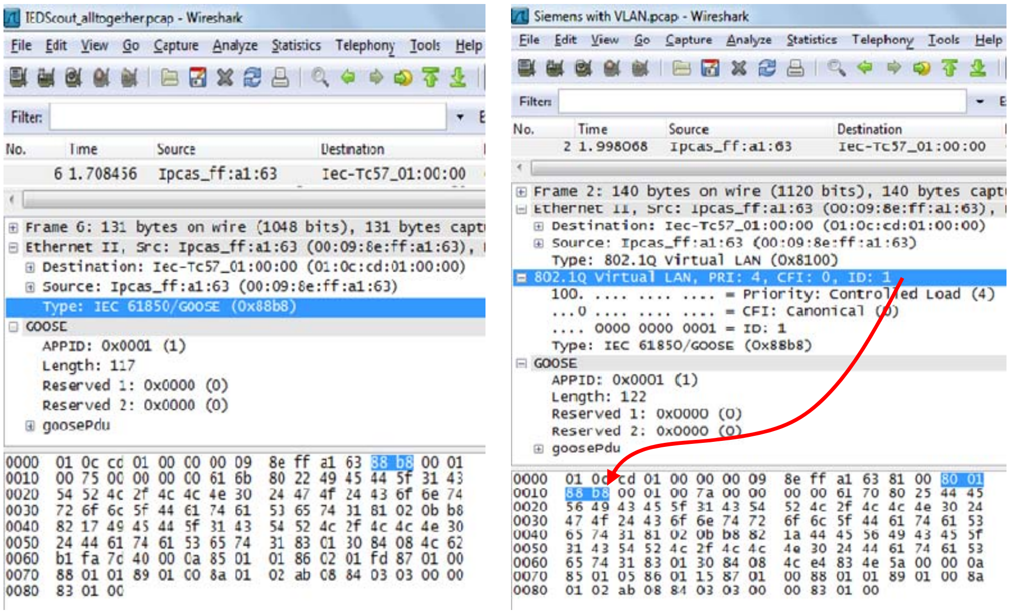


```
No.   Time         Source              Destination          Protocol       Info
   17 4.384181     Ipcas_ff:a1:63      Iec-Tc57_01:00:00    GOOSE

⊞ Frame 17: 136 bytes on wire (1088 bits), 136 bytes captured (1088 bits)
⊞ Ethernet II, Src: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63), Dst: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
⊟ GOOSE
    APPID: 0x0001 (1)
    Length: 122
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: DEVICE_1CTRL/LLN0$GO$Control_DataSet1
      timeAllowedtoLive: 3000
      datSet: DEVICE_1CTRL/LLN0$DataSet1
      goID: 0
      t: Nov 11, 2010 18:45:15.471679687 UTC
      stNum: 9
      sqNum: 6
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 2
    ⊟ allData: 2 items
      ⊟ Data: bit-string (4)
          Padding: 3
          bit-string: 0000
      ⊟ Data: boolean (3)
          boolean: False

0010   00 7a 00 00 00 00 61 70   80 25 44 45 56 49 43 45    .z....ap .%DEVICE
0020   5f 31 43 54 52 4c 2f 4c   4c 4e 30 24 47 4f 24 43    _1CTRL/L LN0$GO$C
0030   6f 6e 74 72 6f 6c 5f 44   61 74 61 53 65 74 31 81    ontrol_D ataSet1.
0040   02 0b b8 82 1a 44 45 56   49 43 45 5f 31 43 54 52    .....DEV ICE_1CTR
0050   4c 2f 4c 4c 4e 30 24 44   61 74 61 53 65 74 31 83    L/LLN0$D ataSet1.
0060   01 30 84 08 4c dc 39 bb   78 c0 00 0a 85 01 09 86    .0..L.9. x.......
0070   01 06 87 01 00 88 01 01   89 01 00 8a 01 02 ab 08    ........ ........
0080   84 03 03 00 00 83 01 00                              ........

boolean (goose.boolean), 1 byte        Packets: 20 Displayed: 20 Marked: 0 Load time: 0:00.000
```

F2 key pressed
(RESET LED)

Figure 6.18: Experiment result (b)

[**Note on VLAN Tags**: Being able to visualize the VLAN tag header on a Wireshark capture can be helpful especially when doing Network segregation or a debugging operation within the Substation Automation System (SAS), but depending on the Switch configuration (whether it is VLAN aware or not, or not configured appropriately) and also on your Network Card Adapter (**NCA**) settings, one might not see the VLAN tag header. Figure 6.19 illustrates both cases where two Wireshark captures represent the same GOOSE message published from one IED. On the right hand side the VLAN tag header is present, but missing on the left hand side.


IEDScout_alltogether.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Siemens with VLAN.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:

```
No.    Time         Source            Destination
     6 1.708456     Ipcas_ff:a1:63    Iec-Tc57_01:00:00

⊞ Frame 6: 131 bytes on wire (1048 bits), 131 bytes capt
⊟ Ethernet II, Src: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63),
  ⊞ Destination: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
  ⊞ Source: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63)
    Type: IEC 61850/GOOSE (0x88b8)
⊟ GOOSE
    APPID: 0x0001 (1)
    Length: 117
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊞ goosePdu

0000   01 0c cd 01 00 00 00 09   8e ff a1 63 88 b8 00 01
0010   00 75 00 00 00 00 61 6b   80 22 49 45 44 5f 31 43
0020   54 52 4c 2f 4c 4c 4e 30   24 47 4f 24 43 6f 6e 74
0030   72 6f 6c 5f 44 61 74 61   53 65 74 31 81 02 0b b8
0040   82 17 49 45 44 5f 31 43   54 52 4c 2f 4c 4c 4e 30
0050   24 44 61 74 61 53 65 74   31 83 01 30 84 08 4c 62
0060   b1 fa 7c 40 00 0a 85 01   01 86 02 01 fd 87 01 00
0070   88 01 01 89 01 00 8a 01   02 ab 08 84 03 03 00 00
0080   83 01 00
```

```
No.    Time         Source            Destination
     2 1.998068     Ipcas_ff:a1:63    Iec-Tc57_01:00:00

⊞ Frame 2: 140 bytes on wire (1120 bits), 140 bytes capt
⊟ Ethernet II, Src: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63),
  ⊞ Destination: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
  ⊞ Source: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63)
    Type: 802.1Q Virtual LAN (0x8100)
⊟ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 1
    100. .... .... .... = Priority: Controlled Load (4)
    ...0 .... .... .... = CFI: Canonical (0)
    .... 0000 0000 0001 = ID: 1
    Type: IEC 61850/GOOSE (0x88b8)
⊟ GOOSE
    APPID: 0x0001 (1)
    Length: 122
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊞ goosePdu

0000   01 0c cd 01 00 00 00 09   8e ff a1 63 81 00 80 01
0010   88 b8 00 01 00 7a 00 00   00 00 61 70 80 25 44 45
0020   56 49 43 45 5f 31 43 54   52 4c 2f 4c 4c 4e 30 24
0030   47 4f 24 43 6f 6e 74 72   6f 6c 5f 44 61 74 61 53
0040   65 74 31 81 02 0b b8 82   1a 44 45 56 49 43 45 5f
0050   31 43 54 52 4c 2f 4c 4c   4e 30 24 44 61 74 61 53
0060   65 74 31 83 01 30 84 08   4c e4 83 4e 5a 00 00 0a
0070   85 01 05 86 01 15 87 01   00 88 01 01 89 01 00 8a
0080   01 02 ab 08 84 03 03 00   00 83 01 00
```

Figure 6.19: GOOSE frame with and without 802.1Q vlan tag

The difference while doing the capture is that one capture (right hand side) has been done in a peer-to-peer mode (IED- to- Laptop). The laptop used was configured in a way that the Network Adapter Card did not strip the 802.1Q VLAN tags from incoming GOOSE messages. The second capture (left hand side) has been done through a switch in a VLAN-aware mode (Strip 802.1Q VLAN tags at switch level).

As the method to configure the Network devices in order to view an 802.1Q VLAN tag from incoming Ethernet frames falls out of the focus of this Thesis and also varies according to the equipment (NCA and switch) being considered, internet links reported in Table 6.13 are of great help for more information on this topic.

Table 6.13: Internet link for VLAN tag investigation

Area of interest	Internet links
Know how to set up your NCA to allow the VLAN tag header of a frame to be accessed at the data link layer.	http://wiki.wireshark.org/CaptureSetup/VLAN?action=raw
	http://www.intel.com/support/network/sb/CS-005897.htm
Know how to configure a Rugged Switch with respect to VLAN	http://www.ruggedcom.com

]

6.2.6.4 Comments on GOOSE structure analyses

This series of simulation and experimentation detailed in this section, has presented a detailed definition of the GOOSE message which is one of the communication protocols associated with the IEC 61850 standard. This section has successfully confirmed that the structure of the simulated GOOSE message conformed to the IEC 61850 standard and to that of a GOOSE message generated by commercially available IEDs.

The second set of experimentation results presented in section 6.3 below encompasses the testing and results analysis for what was the focus of the research project – the integration of the DK60 embedded platform from BECK-IPC and the IEC 61850 communication stack from SystemCorp into an IEC61850 compliant application.

6.3 Experimentation: Preparation of a Test Bench for an Overcurrent event – Goose messages generated between commercial IED and embedded system controller connected to an emulated circuit-breaker

This section describes the test procedure that has been performed on the research project in order to verify the functional and temporal operation of the actuator developed in compliance with the IEC61850 standard.

6.3.1 Testing methodology for circuit breaker actuator

In order to establish what an industry-standard GOOSE frame format looks like, three IEDs have previously been set up as illustrated in Figure 6.14 (page175) and Figure 6.15 (page 175). This has allowed confirmation in Section 6.2.6.3, that the GOOSE frame structure of the commercial IED used (SIPROTEC 7SD5) conformed to the one observed using a simulation tool in Section 6.2.5 (i.e. IEDScout), and also that both of those conform to the one specified in the IEC 61850 standard (Cf. Figure 6.2, page156).

The objectives of the present testing methodology are stated as follow:

- First, to verify the functional implementation of an overcurrent protection scheme distributed between a commercial IED and an embedded microcontroller platform (Actuator) connected to an emulated circuit-breaker; and,
- Secondly - to evaluate the temporal response of the system (actuator and emulated circuit breaker) when an overcurrent event occurs.

As shown below in Figure 6.20, a test injection set (OMICRON- CMC 256) is used to simulate an overcurrent fault that is to be detected by an industry compliant protection IED (SIEMENS_SIPROTEC 7SD5). Once a fault condition has been detected, the instantaneous Overcurrent element of the SIPROTEC device (IED_1) is going to operate and close the output contact. At that instant in time, a GOOSE message is published on the network via the Logical Node zero. The Dataset contained in the published GOOSE carries information relating to the status of the overcurrent element (e.g. IED_1.PROT.PTRC1.Tr.General)

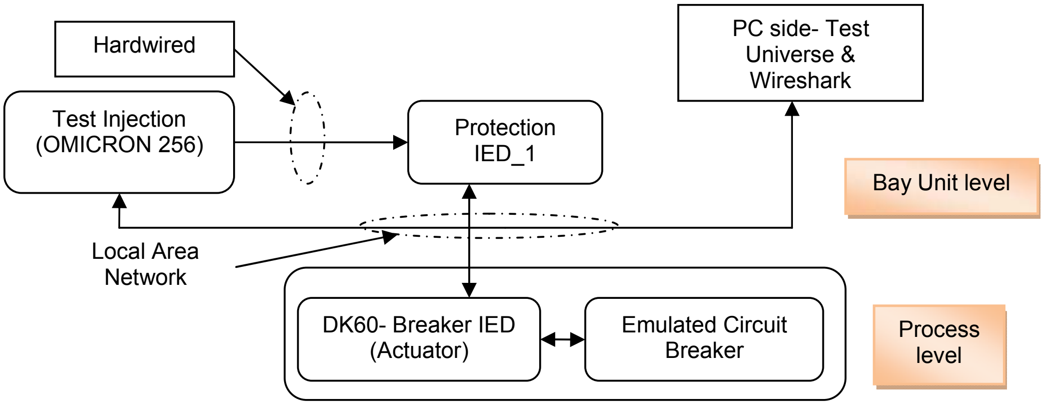


Figure 6.20: Basic topology for case study verification

The GOOSE message published by the SIPROTEC device (IED_1) in response to the overcurrent event carries information instructing the "Intelligent Circuit Breaker

Controller" (on the DK60 board) to open the Breaker. The system has been engineered so that the "Intelligent Circuit Breaker Controller" is also continuously sending the position status of its attached circuit breaker via GOOSE messaging from the embedded system to the industrial IED.

[Note: For this thesis and for reasons of simplicity, it is assumed that the "Intelligent Circuit Breaker Controller" has control over all three phases of the system (One breaker for 3 phases). Since the subscription mechanism is based on the General Trip Flag Alarm (IED_1.PROT.PTRC1.Tr.General), the type of Overcurrent (L-L or L-N etc) test performed is of no importance.
Considering that for all three (3) phases the current has to be within normal operating condition, if any current on any phase has to go beyond the set point limit the "general trip flag" of the PTRC1 logical node is going to be set to True {1}.]

The section to follow details the operational description of the "Intelligent Circuit Breaker Controller" developed in this project.

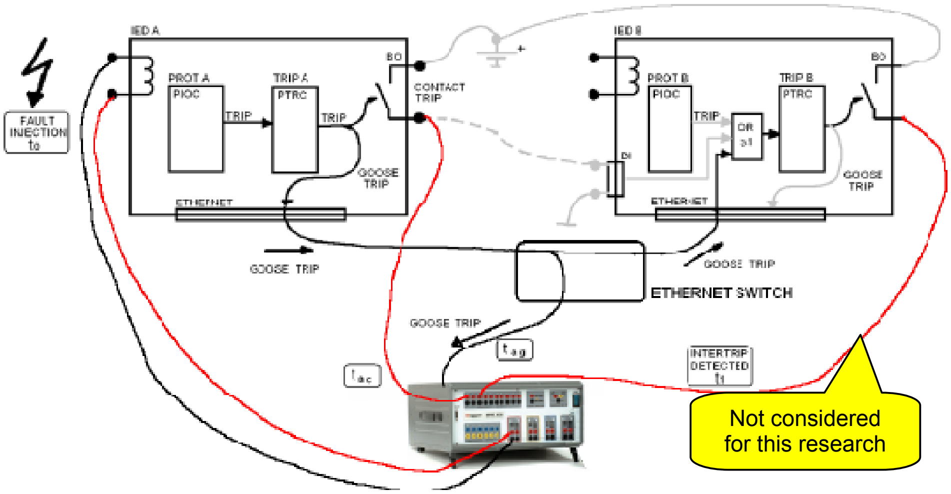
6.3.2 Operational description for the developed circuit breaker actuator

The breaker is put into a "close" position state at initialisation of the DK60 board. The actuator then checks the status of the GOOSE message being subscribed to. If the GOOSE status corresponds to Trip (Fault signalling), the actuator node is going to set the Trip signal (**SW_TRIP**) to high in order for the emulated circuit breaker to move to an "open" position (Cf. Figure 5.18). If no fault is signalled to the circuit breaker actuator at initialisation the emulated breaker is going to remain in its "close" position until a fault is signalled.

The IEC 61850 stack (PIS-10) used is event driven. In case of a fault signalled by the subscribed GOOSE a call-back function is going to be issued by the PIS-10 stack for the application to process the arrival of new information and take appropriate action (Trip the breaker).

6.3.3 Communication set up between the DK60 and other equipment

Figure 6.21 illustrates the inter-trip test setup method which was proposed originally by Bonetti and Douib (2010).



Not considered for this research

Figure 6.21: Original inter-trip test setup
(Bonetti and Douib, 2010)

The communication setup used to interconnect all communication devices is based on a star topology (assuming the existence of the Bay Unit and no process bus connectivity). As compared to the original inter-trip test setup and due to the fact that the DK 60 board does not have any power interface to send its tripping signal back to the Test injection; the signal that is sent back to the Test injection in the case of the current research thesis is the position status of the emulated breaker via information carried in GOOSE messages, rather than physical signals hardwired to the devices.

Figure 6.22 presents the equipment setup for the realisation of an inter-trip test. Note that the acronym “**DUT**” stands for Device Under Test.

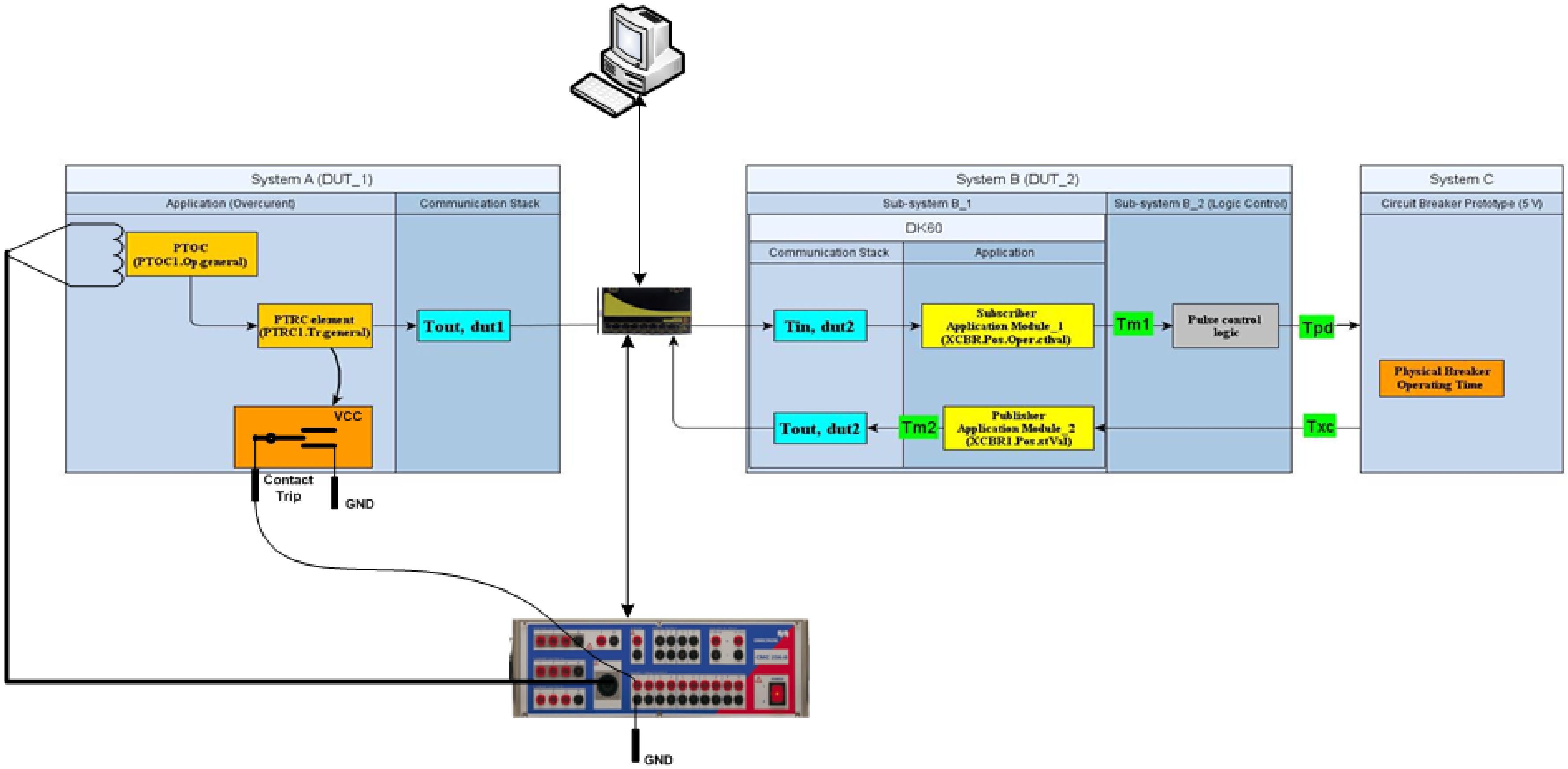


Figure 6.22: Overcurent test set up

A class C IP address range has been used for the network set up (more information on IP address classes can be found at: http://www.tcpipguide.com/free/index.htm). Table 6.14 gives a description of each communicator used and their associated IP Address.

Table 6.14: IP Addresses

Device	Usage	IP Address
DK 60 board	Hosts the Actuator node Application	172.16.10.5
Siemens Siprotec 7SD5 (IED_1)	Commercial IED used to generate GOOSE that is subscribed by the DK60 board.	172.16.10.2
Desktop PC (HMI)	Use to configure and/or control equipment	172.16.10.8
CMC 256	OMICRON test injection equipment used to simulate an overcurrent condition at IED_1	172.16.10.11
Ethernet switch	Ethernet switch used to internet connect all communicators	172.16.10.4

Note: for IP addresses of Class C, the network mask is “**255.255.255.0”.**

For details of the equipment configuration steps required in order to prepare for the generation of an Overcurrent event in the laboratory, please refer to APPENDIX J (page 309).

The next section presents the results obtained for the Overcurrent Test in terms of GOOSE frame structure conformance, performance analysis of GOOSE, and a note on interoperability. Additional work has been included on an aspect of a cyber security issue with respect to start-up from an initial condition (from the powered-down state), and lastly, communication exchange based on MMS.

6.4 Results analysis

6.4.1 Functional response and GOOSE frame format conformance

Figure 6.23 shows the complete lab setup of the embedded system platform together with commercial IED and the test injection set.

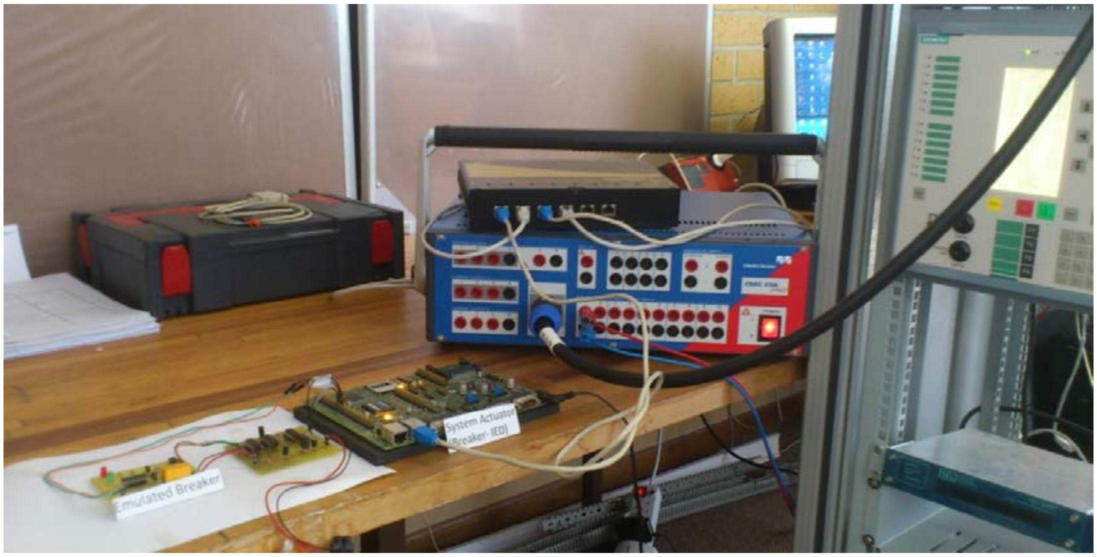


Figure 6.23: Complete lab setup of the case study

Figure 6.24 shows the System Actuator which is composed of the DK60 board and the logic control circuit for pulse generation (Trip and Close command); and the emulated circuit breaker (Red LED indicating "close" position).

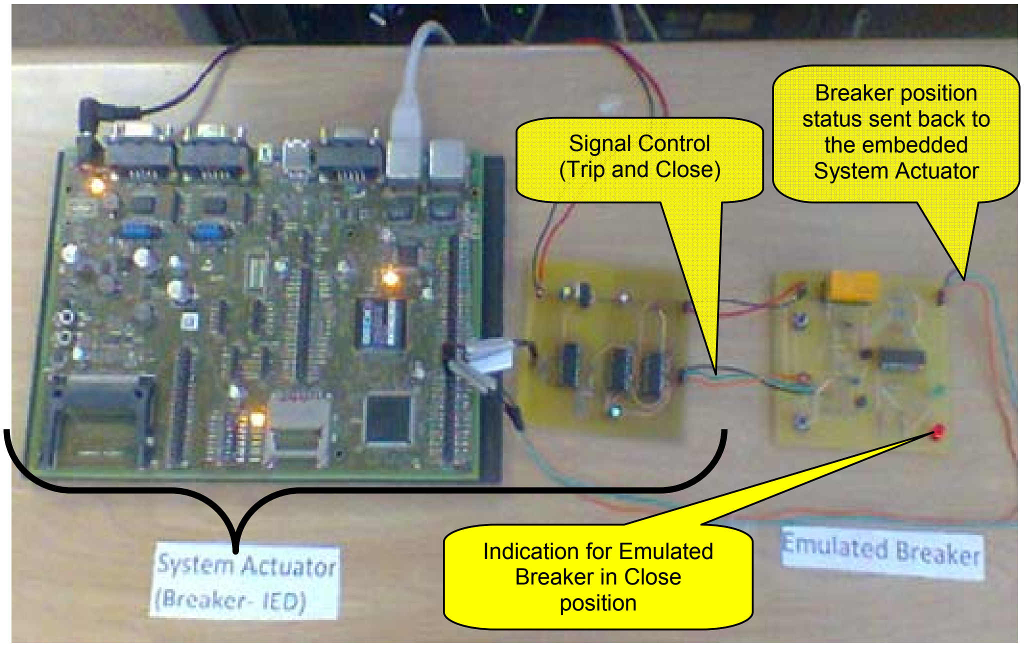

Signal Control (Trip and Close)
Breaker position status sent back to the embedded System Actuator
System Actuator (Breaker- IED)
Indication for Emulated Breaker in Close position
Emulated Breaker

Figure 6.24: Picture of System Actuator and Emulated circuit breaker

Figure 6.25 focuses on the emulated circuit breaker (now in open position)

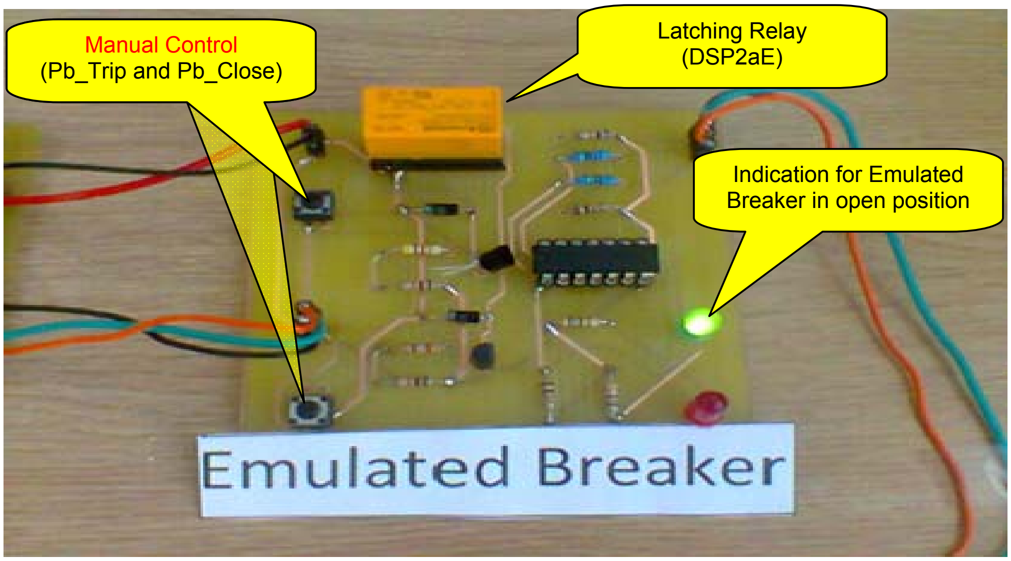
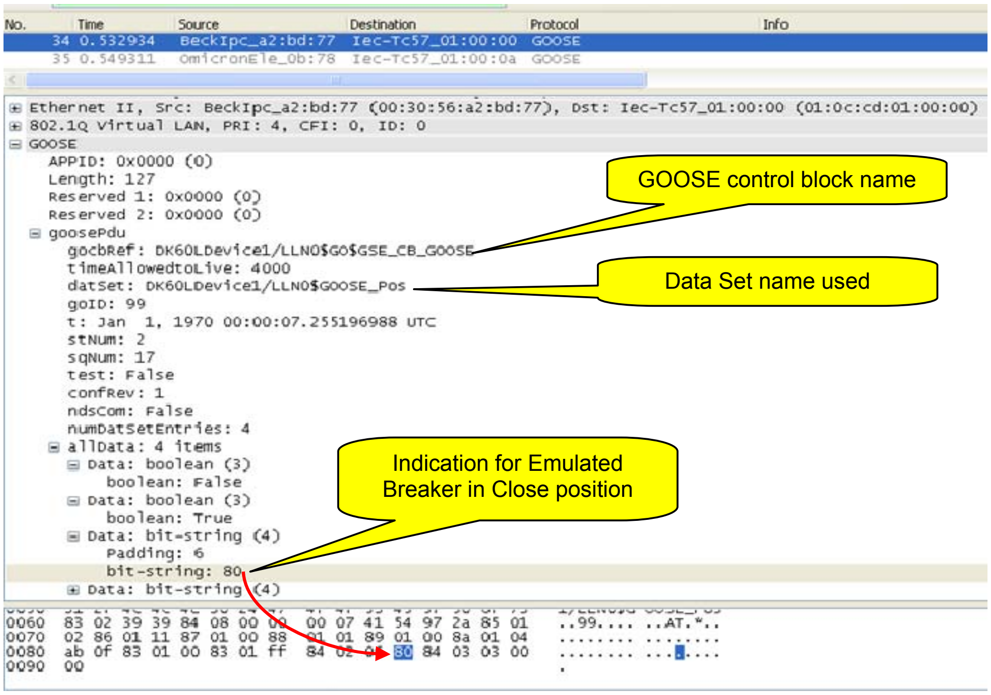


Figure 6.25: Emulated Circuit Breaker in Open positionn

Detailed frame structure of the GOOSE message has been commented upon in section 3.4.2. The observed frame structure of GOOSE messages originating from the DK60 board is similar to that of those originating from the SIPROTEC device. Both frame formats conform to the IEC 61850 specification.

There are some details of the GOOSE content that are assumed to be variable (message specific), such as the GOOSE control block reference name (gocbRef), the Dataset name (datset), and its content.

Figure 6.26 presents a capture of a published GOOSE from the “Intelligent Circuit Breaker Controller” which reports on the position status of the breaker before the initiation of the simulation of an Overcurrent event condition.



```
No.     Time        Source              Destination          Protocol      Info
     34 0.532934    BeckIpc_a2:bd:77    Iec-Tc57_01:00:00    GOOSE
     35 0.549311    OmicronEle_0b:78    Iec-Tc57_01:00:0a    GOOSE

⊞ Ethernet II, Src: BeckIpc_a2:bd:77 (00:30:56:a2:bd:77), Dst: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
⊞ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 0
⊟ GOOSE
    APPID: 0x0000 (0)
    Length: 127
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: DK60LDevice1/LLN0$GO$GSE_CB_GOOSE
      timeAllowedtoLive: 4000
      datSet: DK60LDevice1/LLN0$GOOSE_Pos
      goID: 99
      t: Jan  1, 1970 00:00:07.255196988 UTC
      stNum: 2
      sqNum: 17
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 4
    ⊟ allData: 4 items
      ⊟ Data: boolean (3)
          boolean: False
      ⊟ Data: boolean (3)
          boolean: True
      ⊟ Data: bit-string (4)
          Padding: 6
          bit-string: 80
      ⊞ Data: bit-string (4)

0060  83 02 39 39 84 08 00 00  00 07 41 54 97 2a 85 01   ..99.... ..AT.*..
0070  02 86 01 11 87 01 00 88  01 01 89 01 00 8a 01 04   ........ ........
0080  ab 0f 83 01 00 83 01 ff  84 02 06 80 84 03 03 00   ........ ...▮....
0090  00                                                 .
```

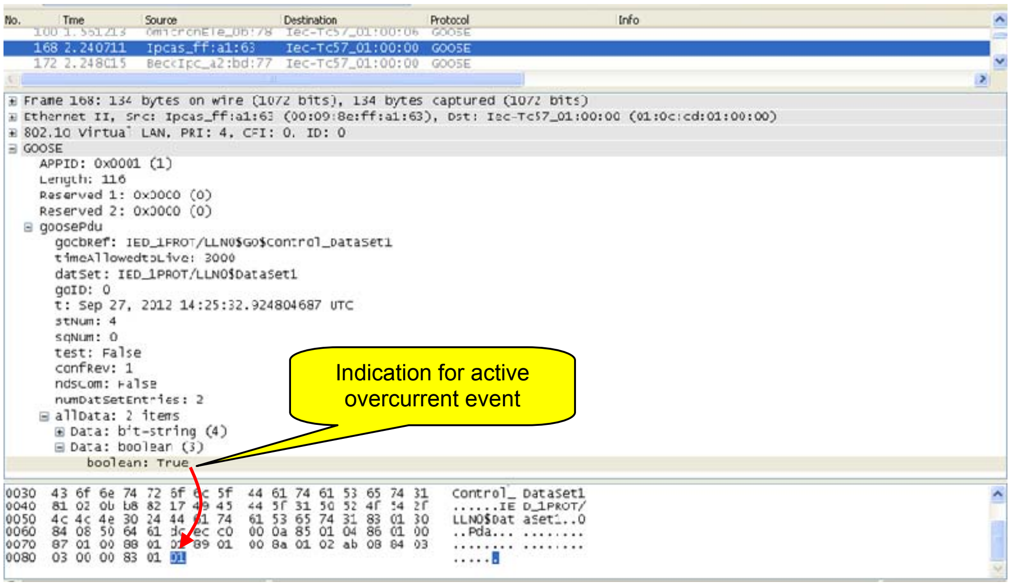
GOOSE control block name

Data Set name used

Indication for Emulated Breaker in Close position

Figure 6.26: Breaker position before overcurrent event

Figure 6.27 below shows the capture of the published GOOSE coming from IED_1 (SIPROTEC device) when an overcurrent condition has been detected (for any of the three phases).

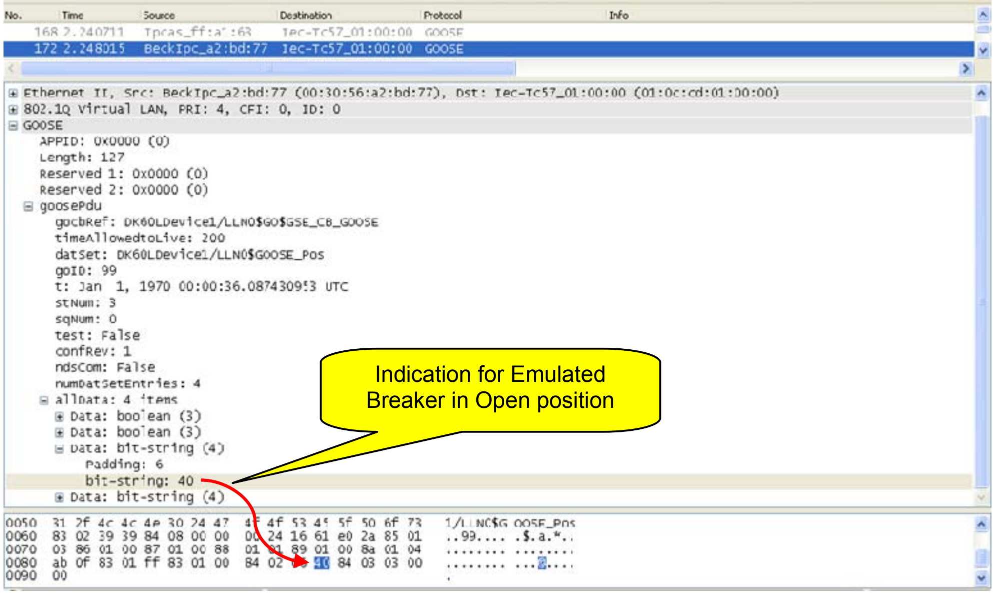


```
No.     Time        Source              Destination          Protocol      Info
    100 1.551213    OmicronEle_0b:78    Iec-Tc57_01:00:06    GOOSE
    168 2.240711    Ipcas_ff:a1:63      Iec-Tc57_01:00:00    GOOSE
    172 2.248015    BeckIpc_a2:bd:77    Iec-Tc57_01:00:00    GOOSE

⊞ Frame 168: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits)
⊞ Ethernet II, Src: Ipcas_ff:a1:63 (00:09:8e:ff:a1:63), Dst: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
⊞ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 0
⊟ GOOSE
    APPID: 0x0001 (1)
    Length: 116
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: IED_1PROT/LLN0$GO$Control_DataSet1
      timeAllowedtoLive: 3000
      datSet: IED_1PROT/LLN0$DataSet1
      goID: 0
      t: Sep 27, 2012 14:25:32.924804687 UTC
      stNum: 4
      sqNum: 0
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 2
    ⊟ allData: 2 items
      ⊞ Data: bit-string (4)
      ⊟ Data: boolean (3)
          boolean: True

0030  43 6f 6e 74 72 6f 6c 5f  44 61 74 61 53 65 74 31   Control_ DataSet1
0040  81 02 0b b8 82 17 49 45  44 5f 31 50 52 4f 54 2f   ......IE D_1PROT/
0050  4c 4c 4e 30 24 44 61 74  61 53 65 74 31 83 01 30   LLN0$Dat aSet1..0
0060  84 08 50 64 61 dc ec c0  00 0a 85 01 04 86 01 00   ..Pda... ........
0070  87 01 00 88 01 01 89 01  00 8a 01 02 ab 08 84 03   ........ ........
0080  03 00 00 83 01 01                                  .....▮
```

Indication for active overcurrent event

Figure 6.27: IED_1 when overcurrent fault detected

Figure 6.28 presents a capture of a published GOOSE from the “Intelligent Circuit Breaker Controller” which reports on the position of the breaker after the simulation of an Overcurrent event condition.



```
No.   Time       Source             Destination         Protocol   Info
  168 2.240711   Ipcas_ff:a1:63     Iec-Tc57_01:00:00   GOOSE
  172 2.248015   BeckIpc_a2:bd:77   Iec-Tc57_01:00:00   GOOSE
```

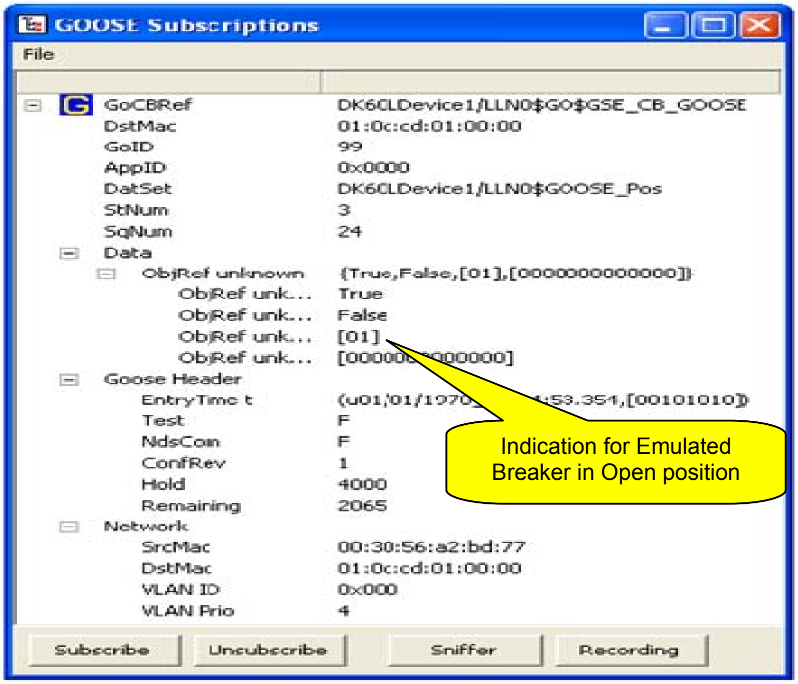
```
⊞ Ethernet II, Src: BeckIpc_a2:bd:77 (00:30:56:a2:bd:77), Dst: Iec-Tc57_01:00:00 (01:0c:cd:01:00:00)
⊞ 802.1Q Virtual LAN, PRI: 4, CFI: 0, ID: 0
⊟ GOOSE
    APPID: 0x0000 (0)
    Length: 127
    Reserved 1: 0x0000 (0)
    Reserved 2: 0x0000 (0)
  ⊟ goosePdu
      gocbRef: DK60LDevice1/LLN0$GO$GSE_CB_GOOSE
      timeAllowedtoLive: 200
      datSet: DK60LDevice1/LLN0$GOOSE_Pos
      goID: 99
      t: Jan  1, 1970 00:00:36.087430953 UTC
      stNum: 3
      sqNum: 0
      test: False
      confRev: 1
      ndsCom: False
      numDatSetEntries: 4
    ⊟ allData: 4 items
      ⊞ Data: boolean (3)
      ⊞ Data: boolean (3)
      ⊟ Data: bit-string (4)
          Padding: 6
          bit-string: 40
      ⊞ Data: bit-string (4)
```

```
0050  31 2f 4c 4c 4e 30 24 47  4f 4f 53 45 5f 50 6f 73   1/LLN0$G OOSE_Pos
0060  83 02 39 39 84 08 00 00  00 24 16 61 e0 2a 85 01   ..99.... .$.a.*..
0070  03 86 01 00 87 01 00 88  01 01 89 01 00 8a 01 04   ........ ........
0080  ab 0f 83 01 ff 83 01 00  84 02 06 40 84 03 03 00   ........ ...@....
0090  00                                                 .
```

Indication for Emulated Breaker in Open position

Figure 6.28: Breaker position after overcurrent event

Figure 6.29 shows an IEDScout capture of a GOOSE message originating from the "Intelligent Circuit Breaker Controller" after the breaker has tripped. This view from IEDScout is much more user friendly and meaningful for those who are not familiar with "Wireshark" software.



```
⊟ G GoCBRef              DK60LDevice1/LLN0$GO$GSE_CB_GOOSE
    DstMac               01:0c:cd:01:00:00
    GoID                 99
    AppID                0x0000
    DatSet               DK60LDevice1/LLN0$GOOSE_Pos
    StNum                3
    SqNum                24
  ⊟ Data
    ⊟ ObjRef unknown     {True,False,[01],[0000000000000]}
        ObjRef unk...    True
        ObjRef unk...    False
        ObjRef unk...    [01]
        ObjRef unk...    [0000000000000]
  ⊟ Goose Header
      EntryTime t        {u01/01/1970 [illegible]:53.354,[00101010]}
      Test               F
      NdsCom             F
      ConfRev            1
      Hold               4000
      Remaining          2065
  ⊟ Network
      SrcMac             00:30:56:a2:bd:77
      DstMac             01:0c:cd:01:00:00
      VLAN ID            0x000
      VLAN Prio          4
```

Indication for Emulated Breaker in Open position

Figure 6.29: Breaker position after overcurrent event (IEDScout view)

For any other structural information with regard to the GOOSE message please refer to section 6.2 on page 155.

The results summarized above, have confirmed the functional effectiveness of the software design developed, which was based upon the PIS-10 communication stack from "SystemCorp". It was demonstrated also that the structure of the GOOSE message from commercial IEDs such as the SIPROTEC 7SD5 device was basically the same as the one published by the actuator software application developed for the "Intelligent Circuit Breaker Controller". All the messages were in conformance with that specified by the IEC 61850 standard.

The next section evaluates the performance of the overall GOOSE message transfer time against the time limit specified in the IEC 61850 standard.

6.4.2 GOOSE performance (temporal response) evaluation

In this section the temporal response of the protection scheme based upon GOOSE messaging for the research project case-study is evaluated. GOOSE performance is evaluated with respect to the notion of transfer time. The IEC 61850 standard defines the transfer time of a packet as being the time from "the moment the sender puts the data content on top of its transmission stack up to the moment the receiver extracts the data from its transmission stack". This is illustrated in Figure 6.30.

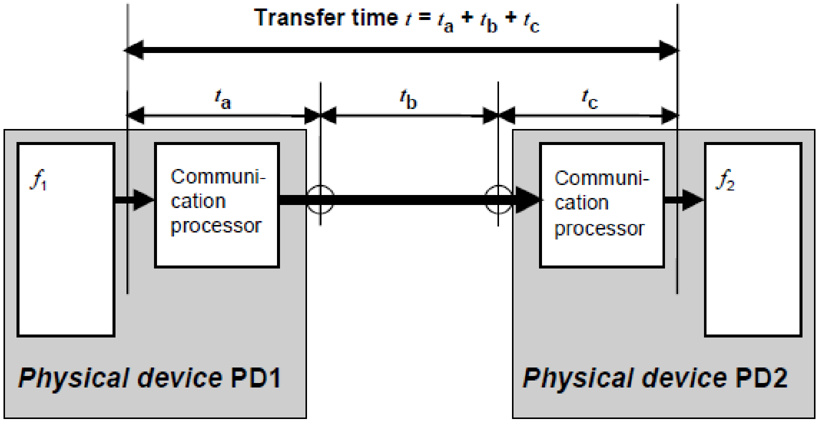
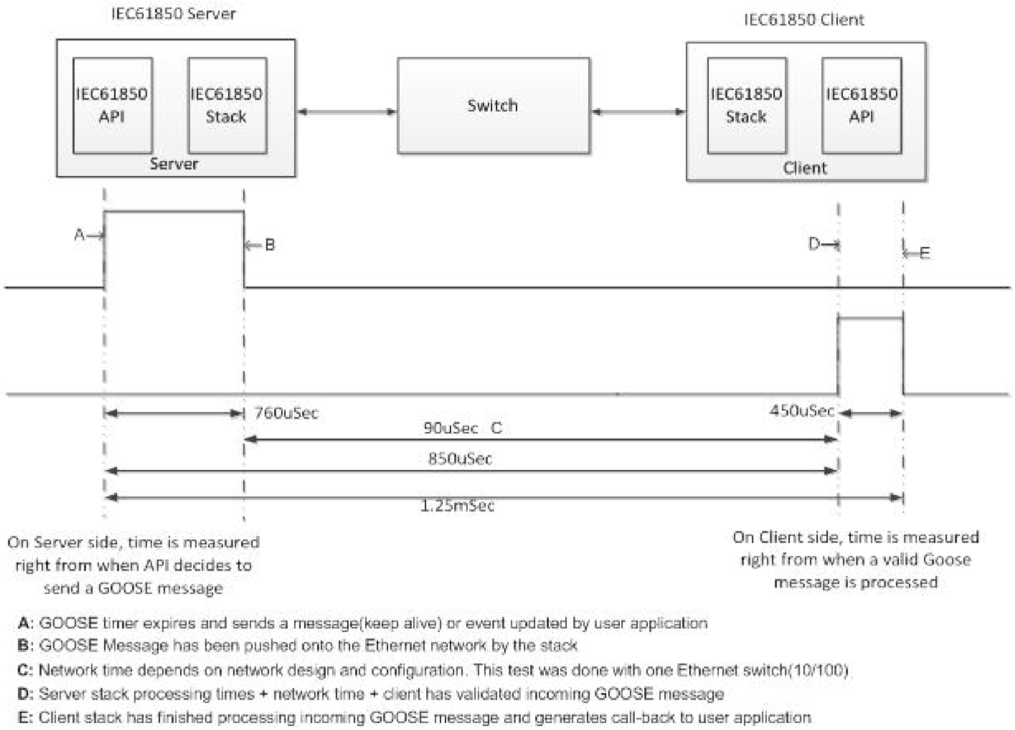


Figure 6.30: Definition of transfer time
(IEC 61850-5, 2003, p.45)

For the research project, which is classified as a Protection Application, the GOOSE transfer time should not be greater than 3 mS as specified within the IEC 61850 standard. (IEC 61850-5, 2003, p.46)

SystemCorp has made available on its site information on the time it takes for a GOOSE to be sent from one DK 60 device to another DK 60 device using the Client/Server services. Figure 6.31 shows the time transmission definition as estimated by SystemCorp.



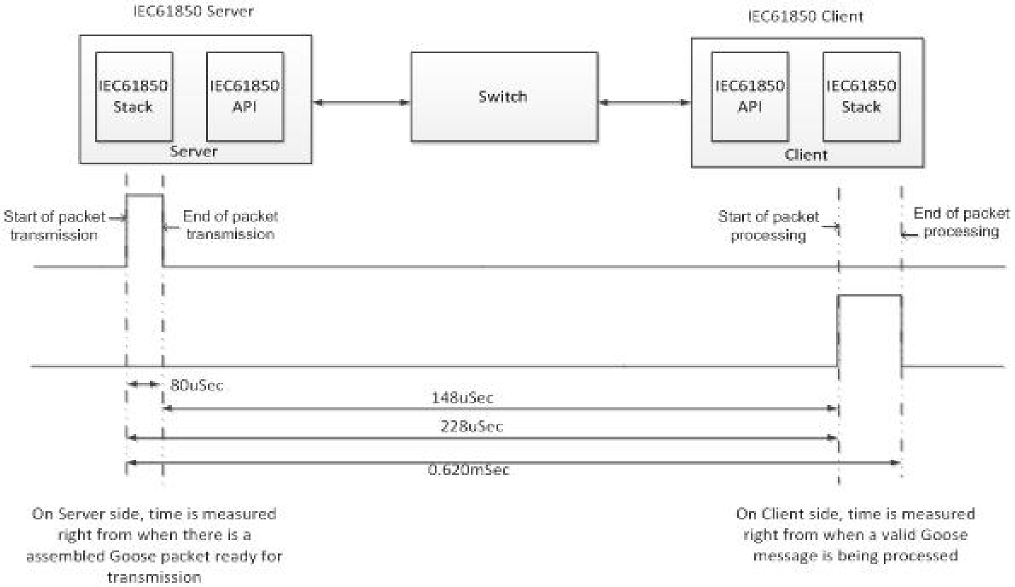
On Server side, time is measured right from when API decides to send a GOOSE message

On Client side, time is measured right from when a valid Goose message is processed

A: GOOSE timer expires and sends a message(keep alive) or event updated by user application
B: GOOSE Message has been pushed onto the Ethernet network by the stack
C: Network time depends on network design and configuration. This test was done with one Ethernet switch(10/100)
D: Server stack processing times + network time + client has validated incoming GOOSE message
E: Client stack has finished processing incoming GOOSE message and generates call-back to user application

Figure 6.31: GOOSE transfer time evaluation under "relative" network load
(SystemCorp_c, 2012)

SystemCorp state that under "relative load" the GOOSE transfer time is about 1.25 mS (SystemCorp_c, 2012). They however do not state under what condition this load was generated.



On Server side, time is measured right from when there is a assembled Goose packet ready for transmission

On Client side, time is measured right from when a valid Goose message is being processed

Figure 6.32: Transfer time evaluation without "relative" network load
(SystemCorp_c, 2012)

It is also claimed by SystemCorp that without any "relative" additional network load the GOOSE transfer time is reduced to 0.620 mS. Both claims (with and without

:relative" load) are clearly under the limit of 3 mS as per the requirement of the IEC 61850 standard. Relevant documents are accessible on:
https://www.systemcorp.com.au/news/83-iec-6185-beck-dk-61-library-release.html

For the inter-trip test setup realised in this thesis, the following times have been defined (please refer to **Figure 6.22**, page 182):

Tpd: Time delay introduced by the "Pulse control Logic".
Txc: Circuit Breaker operating time
TM1: Application processing time of a received GOOSE (after PIS-10 issues a call-back to "MyUpdateFunction")
TM2: Application processing time for a transmitted GOOSE (after process application has issued the call to "IEC61850_Update")
$T_{in,dut2}$: Time delay introduced by the communication stack at reception
$T_{out,dut2}$: Time delay introduced by the communication stack at transmission
$T_{out,dut1}$: Time stamp reference when subscribed GOOSE is issued
T_{net}: Time delay introduced by the communication network itself.

By neglecting **Tpd**, **TM1**, **TM2** and considering **Txc** , " $\mathbf{T_{in,dut2}}$: ", " $\mathbf{T_{out,dut2}}$", $\mathbf{T_{net}}$, the GOOSE Transfer Time ($\mathbf{T_{aprox}}$) is given as a first approximation by:

$$T_{aprox} = T_{out,dut1} + T_{net} + T_{in,dut2} \qquad \textbf{Equation 6.1}$$

Since the value of T_{net} and $T_{out,\ dut2}$ cannot easily be determined, Equation 6.2 is proposed, as referenced from the work done by Sidhu et al. (2010) and Bonetti and Douib (2010). The GOOSE transfer time has been estimated in this thesis using the proposed Equation 6.2:

$$T = \frac{(DK60_{GOOSE} - IED1_{GOOSE} - T_{XC})}{2} \qquad \textbf{Equation 6.2}$$

Where:
$\mathbf{DK60_{GOOSE}}$: is the time it takes for the "Intelligent Circuit Breaker Controller" to issue a new GOOSE message that reports the change of breaker position after the fault current injection has started.

$\mathbf{IED1_{GOOSE}}$: is the time it takes for the SIPROTEC device to release a GOOSE message carrying out a trip order after the fault current injection has started.

Txc: Is the time it takes for the prototype breaker to operate (5 mS /Cf.**DSP2a/ Appendix I).**

Table 6.15 below reports the different times measured during the testing period and relevant transfer times that have been computed. All time values are expressed in milliseconds.

Table 6.15: GOOSE Transfer Time results

	IED_1_Trip (mS)	**IED_1_GOOSE (mS)**	**DK60_GOOSE (mS)**	**GOOSE Transfer Time (mS)**
Number				
1	15.6	15.9	23	1.05
2	15.6	14.5	21.3	0.9
3	16.7	14.9	21.9	1
4	16	16	22.8	0.9
5	23.2	22	28.8	0.9
6	23.2	22	28.6	0.8
7	17.4	16.3	23.4	1.05
7	16.4	14	20.7	0.85
8	15.6	15.9	22.9	1
9	15.9	14.5	21.6	1.05
10	16.1	16.4	23.3	0.95
11	16.2	16.4	23.7	1.15
12	15.9	14.4	21	0.8
13	16.2	16.6	23.4	0.9
14	23.9	22.8	29.3	0.75
15	21.8	21.5	28.1	0.8
16	15.7	14.3	21.5	1.1
17	16.2	14.9	21.7	0.9
18	15.6	16.1	22.9	0.9
19	15.8	16.1	23	0.95
20	16.4	14.3	20.9	0.8
Average (mS)	18.27	17.49	24.69	0.975

As it can be seen from the table above the GOOSE message published by IED_1 (IED_1_GOOSE) is 1 mS faster on average compared to the time taken by the output contact to operate (IED_1_Trip).

Figure 6.33 below, is a graphical representation of results shown previously in Table 6.15 above.

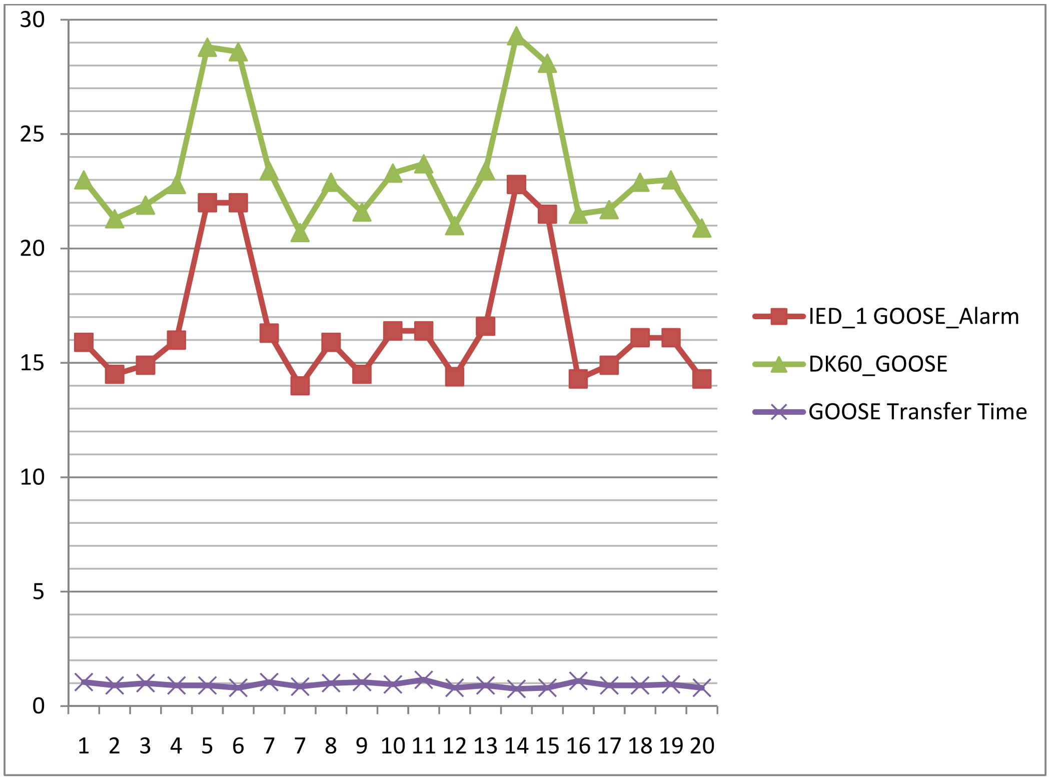


Figure 6.33: GOOSE Transfer time statistics

The GOOSE transfer time achieved through experimentation in this project is approximately **0.975 mS** against 0.62 mS as stated by SystemCorp under the condition of no "relative" network load. Nevertheless, this result is clearly under the limit of 3 mS as required by the IEC 61850 standard.

The experimental results deviate from what have been announced by SystemCorp for the following reasons:

- The application developed includes a server and a client which do not run concurrently, but instead they run in an event driven fashion.
- The use of a circuit breaker emulator prototype (System C on Figure 6.22, p 182) which introduces a delay that could only be estimated based upon information provided by the manufacturer i.e. the operating time of the latching relay used as part of the Circuit breaker emulator prototype is stated as 5 mS, but this time might have been a bit different during the execution of the test.

The next section comments on some of the interoperability issues experienced during the project development phase.

6.4.3 Interoperability achievement

For the demonstration of interoperability, the status position of the circuit breaker has been successfully mapped from the DK60 (vendor: Beck IPC) board back to the SIPROTEC device (vendor: Siemens) and to an SEL-421 (vendor: SEL) - i.e. not an operation where ONLY equipment from the SAME vendor is used to exchange GOOSE messages.

Some issues were encountered with respect to the manner in which the ICD Designer (version 1.1.0.12) builds the XML structure of SCL files. Due to the absence of some XML keywords or syntax issue of the DK60SV1.CID file (constructed with ICD Designer), the DIGSI 4 and the SEL Architect environment were not able to load the SCL file properly (DK60SV1.CID). This challenge needed to be solved to allow the interoperability between the devices from the different vendors – i.e. to engineer the system architecture.

A solution to the challenge required detailed knowledge of IEC 61850-6 (Ed.1), and includes also the use the of the SEL Architect XML file-checker as it is indeed much more precise when reporting errors on SCL files as compared to the facilities which IEDScout allows.

Normally fixing the error requires examination of part six (6) of the IEC 61850 standard and looking for specific information on syntax based on keyword errors provided to you by the “SEL Architect” environment. Modify the SCL file manually using XML marker or Notepad. Load the file back into the SEL Architect-XML file checker. Repeat until error-free. Subsequent to this labour-intensive iterative process, loading the file onto another manufacturer’s, platform such as DIGSI 4 should be successful. For more details on the procedural steps needed in order to fix possible errors please refer to APPENDIX K on page 321.

6.4.4 Cyber security risk evaluation

A cyber-security risk evaluation consists of verifying the behaviour of an IED when it is exposed to duplicated or corrupted communication messages, since this can be the case during a substation threat intrusion into the network.

A cyber-security test threat is simulated by creating an out of sequence legacy GOOSE message and then injecting this onto the network to establish the response of the DK 60 kit to it.

[Note: Using “Colasoft Packet Builder 1.0”, specific packets can be assembled (or loaded) that suit the need of the cyber threat to be simulated, then inject them onto the network being investigated. For more information on “Colasoft Packet Builder 1.0” at: http://www.colasoft.com/)]

The steps to generate the test-threat are detailed below:

1) Capture a GOOSE subscribed to by the actuator with Wireshark while performing the overcurrent test with the CMC 256 test injection (see Figure 6.34).

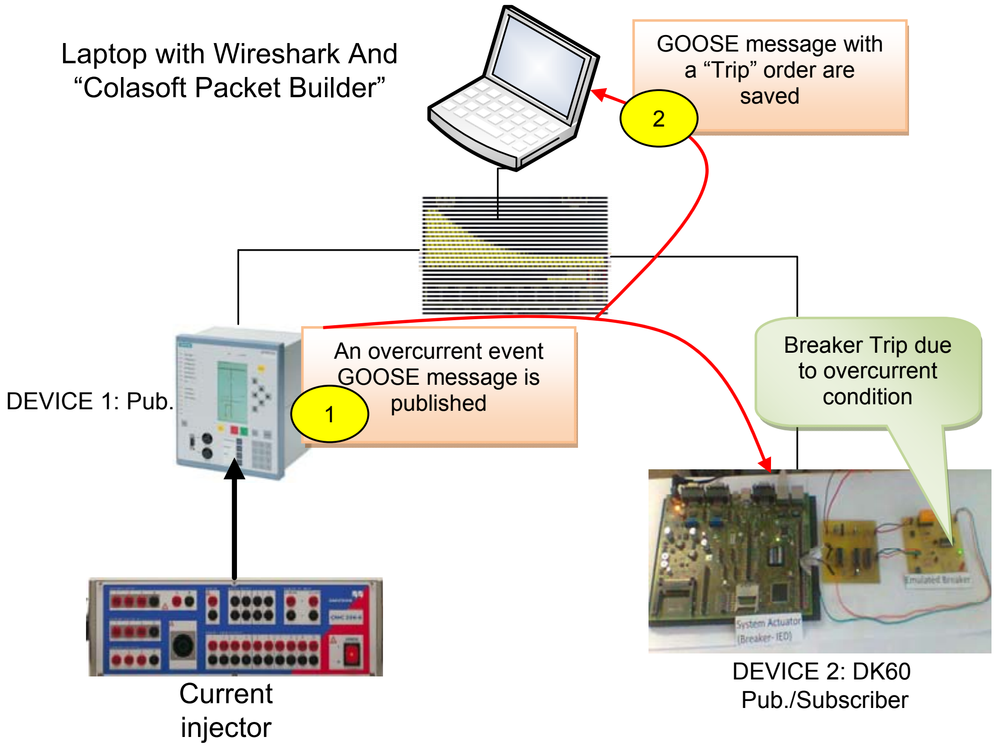

Laptop with Wireshark And
“Colasoft Packet Builder”
GOOSE message with a “Trip” order are saved
2
An overcurrent event GOOSE message is published
1
Breaker Trip due to overcurrent condition
DEVICE 1: Pub.
Current injector
DEVICE 2: DK60 Pub./Subscriber
System Actuator (Breaker- IED)
Emulated Breaker

Figure 6.34: First part of the cybert security test

2) Import the message captured during the fault test-injection into “Packet Builder”.

3) Run the system with no fault test-injection occurring (test-injection set off). GOOSE messages indicating no fault condition are now running on the network. Circuit breaker emulator rightfully reflects non-faulted operation.

4) Inject the message which had previously been imported into “Colasoft Packet Builder 1.0” onto the network. This now represents the intrusive out-of-sequence legacy GOOSE message – which should be discarded if the system has been designed against attack. Figure 6.34 and Figure 6.35 show the transmission of this packet onto the network.

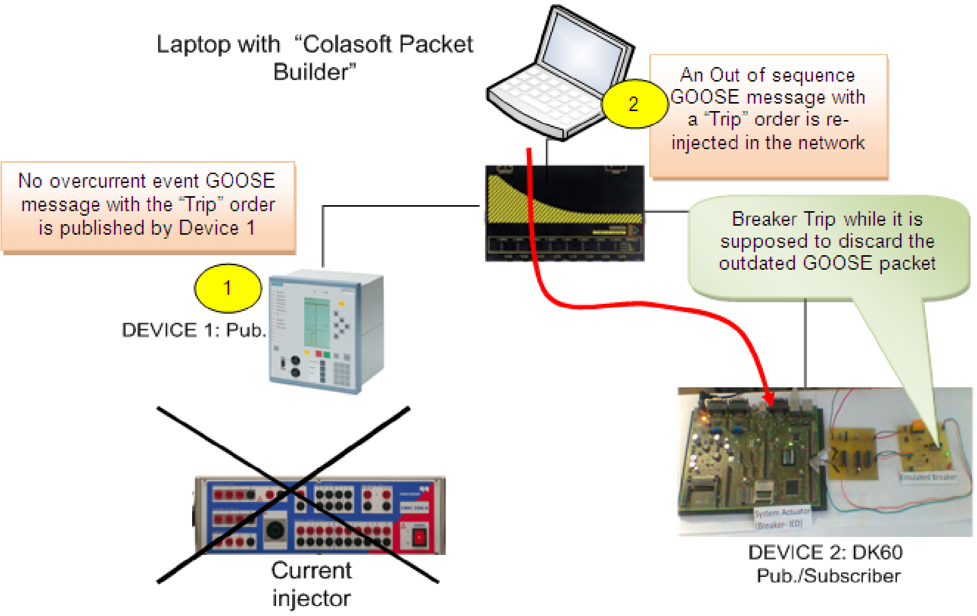
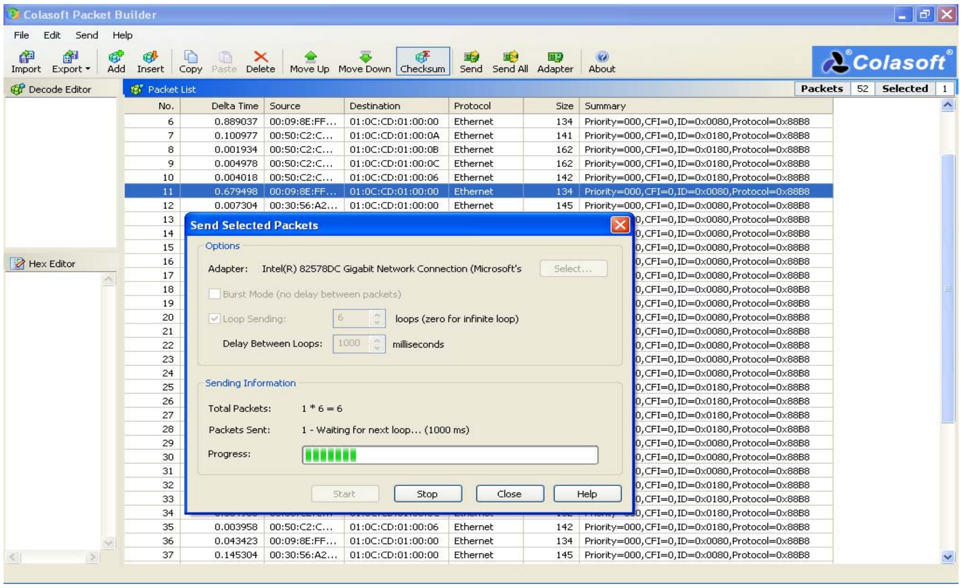

Laptop with "Colasoft Packet Builder"
2
An Out of sequence GOOSE message with a "Trip" order is re-injected in the network
No overcurrent event GOOSE message with the "Trip" order is published by Device 1
Breaker Trip while it is supposed to discard the outdated GOOSE packet
1
DEVICE 1: Pub.
Current injector
DEVICE 2: DK60 Pub./Subscriber

Figure 6.34: First part of the cybert security test


Colasoft Packet Builder
File Edit Send Help
Import Export Add Insert Copy Paste Delete Move Up Move Down Checksum Send Send All Adapter About
Colasoft
Decode Editor
Hex Editor
Packet List

Packets 52 Selected 1

No.	Delta Time	Source	Destination	Protocol	Size	Summary
6	0.889037	00:09:8E:FF...	01:0C:CD:01:00:00	Ethernet	134	Priority=000,CFI=0,ID=0x0080,Protocol=0x88B8
7	0.100977	00:50:C2:C...	01:0C:CD:01:00:0A	Ethernet	141	Priority=000,CFI=0,ID=0x0180,Protocol=0x88B8
8	0.001934	00:50:C2:C...	01:0C:CD:01:00:0B	Ethernet	162	Priority=000,CFI=0,ID=0x0180,Protocol=0x88B8
9	0.004978	00:50:C2:C...	01:0C:CD:01:00:0C	Ethernet	162	Priority=000,CFI=0,ID=0x0180,Protocol=0x88B8
10	0.004018	00:50:C2:C...	01:0C:CD:01:00:06	Ethernet	142	Priority=000,CFI=0,ID=0x0180,Protocol=0x88B8
11	0.679498	00:09:8E:FF...	01:0C:CD:01:00:00	Ethernet	134	Priority=000,CFI=0,ID=0x0080,Protocol=0x88B8
12	0.007304	00:30:56:A2...	01:0C:CD:01:00:00	Ethernet	145	Priority=000,CFI=0,ID=0x0080,Protocol=0x88B8
13						0,CFI=0,ID=0x0080,Protocol=0x88B8
14						0,CFI=0,ID=0x0080,Protocol=0x88B8
15						0,CFI=0,ID=0x0080,Protocol=0x88B8
16						0,CFI=0,ID=0x0080,Protocol=0x88B8
17						0,CFI=0,ID=0x0080,Protocol=0x88B8
18						0,CFI=0,ID=0x0080,Protocol=0x88B8
19						0,CFI=0,ID=0x0080,Protocol=0x88B8
20						0,CFI=0,ID=0x0080,Protocol=0x88B8
21						0,CFI=0,ID=0x0080,Protocol=0x88B8
22						0,CFI=0,ID=0x0080,Protocol=0x88B8
23						0,CFI=0,ID=0x0080,Protocol=0x88B8
24						0,CFI=0,ID=0x0080,Protocol=0x88B8
25						0,CFI=0,ID=0x0180,Protocol=0x88B8
26						0,CFI=0,ID=0x0180,Protocol=0x88B8
27						0,CFI=0,ID=0x0180,Protocol=0x88B8
28						0,CFI=0,ID=0x0180,Protocol=0x88B8
29						0,CFI=0,ID=0x0080,Protocol=0x88B8
30						0,CFI=0,ID=0x0080,Protocol=0x88B8
31						0,CFI=0,ID=0x0080,Protocol=0x88B8
32						0,CFI=0,ID=0x0180,Protocol=0x88B8
33						0,CFI=0,ID=0x0180,Protocol=0x88B8
34						0,CFI=0,ID=0x0180,Protocol=0x88B8
35	0.003958	00:50:C2:C...	01:0C:CD:01:00:06	Ethernet	142	Priority=000,CFI=0,ID=0x0180,Protocol=0x88B8
36	0.043423	00:09:8E:FF...	01:0C:CD:01:00:00	Ethernet	134	Priority=000,CFI=0,ID=0x0080,Protocol=0x88B8
37	0.145304	00:30:56:A2...	01:0C:CD:01:00:00	Ethernet	145	Priority=000,CFI=0,ID=0x0080,Protocol=0x88B8

Send Selected Packets

Options

Adapter: Intel(R) 82578DC Gigabit Network Connection (Microsoft's

☐ Burst Mode (no delay between packets)

☑ Loop Sending: 6 loops (zero for infinite loop)

Delay Between Loops: 1000 milliseconds

Sending Information

Total Packets: 1 * 6 = 6

Packets Sent: 1 - Waiting for next loop... (1000 ms)

Progress:

Figure 6.35: Introduction of outdated GOOSE packet back onto the network

5) The response to the intrusive out-of-sequence legacy GOOSE message was that the emulated breaker tripped. It should not have done so in response to this message - the GOOSE frame should simply have been discarded since the sequence number (sqNum) and the time stamp were outdated. Unfortunately for the case of the current thesis the aspect of cyber-security tested for is not implemented within the IEC 61850 protocol integration stack (PIS-10 version 1.36). Certainly this will be fixed in future releases of the PIS-10.

The same test (Injection of an out of sequence legacy GOOSE message) has been performed on an IED commonly used in the substation environment-SIPROTEC device from SIEMENS, in order to verify the behaviour of the device with respect to the same issue. All outdated GOOSE packets were discarded by the SIPROTEC device.

The cyber-security test-case outlined above, which compares the response of a commercially available IED to that of a development platform clearly illustrates the type of test which could be run to ascertain whether the system has the level of cyber-security required or specified.

The next section details a brief description of breaker control based MMS over TCP/IP.

6.4.5 Breaker control based on MMS over TCP/IP

An "Intelligent Circuit Breaker Controller" fully compatible with the IEC 61850 standard needs to support the **Select-Before-Operate** method. This is not the case in this project because the focus has primarily been put on GOOSE message communication (see Figure 3.15).

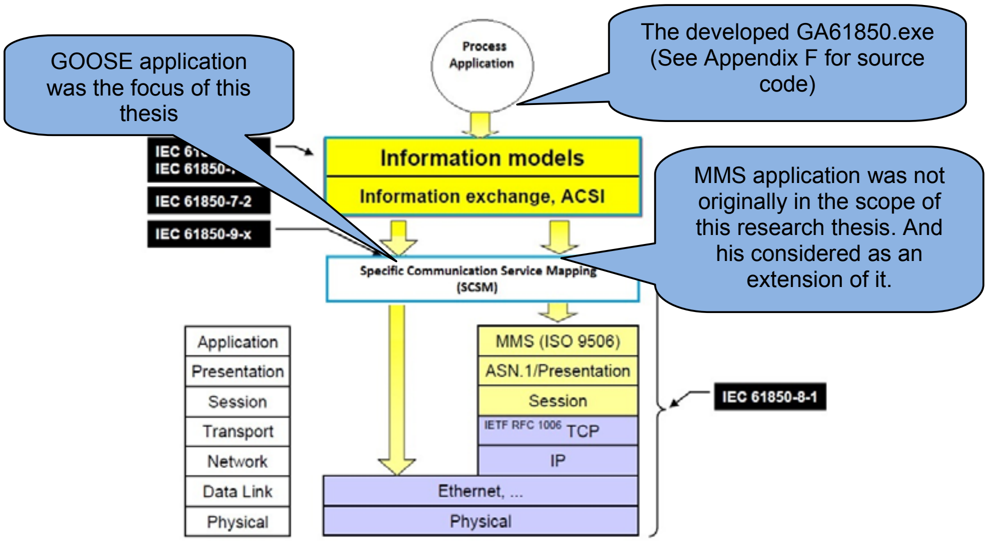

GOOSE application was the focus of this thesis
Process Application
The developed GA61850.exe (See Appendix F for source code)
IEC 61
IEC 61850-
Information models
Information exchange, ACSI
IEC 61850-7-2
IEC 61850-9-x
MMS application was not originally in the scope of this research thesis. And his considered as an extension of it.
Specific Communication Service Mapping (SCSM)
Application
Presentation
Session
Transport
Network
Data Link
Physical
MMS (ISO 9506)
ASN.1/Presentation
Session
IETF RFC 1006 TCP
IP
Ethernet, ...
Physical
IEC 61850-8-1

Figure 6.35: Mapping of IEC 61850 specifics layers to OSI model

Adapted from (IEC 61850-7-1, 2003, p.22)

Implementation of the Select-Before-Operate method will have required the use of a dedicated client HMI that could also support this method. As IEDScout (Universal Client) has been used extensively in this thesis, it has been decided to use it in order to establish basic MMS connectivity with the DK60 board.

Unfortunately the use of IEDScout as a Client to control the Breaker position has been possible only via a communication method call **direct-with-normal-security**, but the **Select-Before-Operate** method has not been achieved.

To view samples of MMS message capture done with MMS-Ethereal software, please refer to APPENDIX L on page 323.

The next section presents a summary of the current chapter.

6.5 Conclusion

Chapter 6 has presented the setup used in order to demonstrate the effectiveness of the design of an “Intelligent Circuit Breaker Controller” with limited capabilities. The Test setup demonstrates that a device has been developed that has a certain level of compliance with the IEC 61850 standard.

User friendliness of DK60 environment for engineering the system architecture
Interoperability has been achieved only through additional hand modification of the DK60SV1.CID file.

The inter-trip test developed here has demonstrated an average GOOSE transfer time of 0.975 mS which largely satisfies the requirement of the IEC 61850 on GOOSE transfer time (3 mS). Note that Sampled Value messages were not present on the Network while testing GOOSE transfer time – therefore. It is recommended for future development to investigate the performance more fully of an IEC 61850 process bus with regard to the concurrent use of GOOSE and SV messages.

On the design of the IEC 61850 communication stack, the following recommendations have been proposed to “SystemCorp” to allow users in the next release of the PIS-10 stack with the following configuration:

- Client object + Subscriber module (for Client IEDs such as HMI on top of the hierarchy).
- Server object + Publisher + Subscriber module (for field device such as Protective relay).

.Table 6.16 show clearly the current structure of the PIS-010 stack versus the proposed one for future release. This new proposed structure will allow directly the developer of an IEC 61850 compliant application to include subscription capability in its application without the need to create a client object within the source code of the software application.

Table 6.16: Current PIS-010 stack structure vs proposition for future release

		Communicator Default type		**Communicator association type**	
	IED Type	**Server**	**Client**	**Publisher**	**Subscriber**
Current PIS-010 stack configuration	HMI or SCADA systems	N/A	OK	N/A	OK
	Field device (Protection relays etc)	OK	-	OK	N/A
Proposition for future released of the PIS-010	HMI or SCADA systems	N/A	OK	N/A	OK
	Field device (Protection relays etc)	OK	-	OK	OK

Aspects of Cyber security is also an issue that “SystemCorp” IT engineers are currently trying to resolve (ongoing work).

In this research thesis, in order to develop an actuator for circuit breaker capable to subscribe to a GOOSE message and be able to send back the position status of an emulated circuit breaker via GOOSE message, the development of a Server object together with a client object was required.

The next chapter offers a general conclusion to the current project, a drawback analysis with respect to the thesis deliverables, application of results and future work.

CHAPTER SEVEN:

CONCLUSION AND FUTURE WORK

7.1 Introduction

The IEC 61850 standard for “Communication networks and systems in substations” (Ed.1) has been progressively released between the years 2003 and 2005. Since then, vendors of substation equipment had implemented their own interpretation of the standard and this has sometimes led to interoperability issues.

Currently there are only three companies that develop an IEC 61850 commercially available communication stack with the possibility for use in the public domain. These companies are:

- SISCO;
- Triangle Microworks and;
- SystemCorp

These software communication stacks are said to be platform (hardware plus operating system) independent, however their deployment onto a specific hardware platform still requires significant integration work and a detailed understanding of the hardware and the Real Time Operating System (RTOS) deployed on the hardware target. Due to on-going developments of these communication stacks it is vitally important that version revisions and functional capabilities are kept track of as deployed on the various platforms. Not doing so may lead to interoperability issues.

SystemCorp released their first IEC 61850 communication stack (PIS-10) in 2010. New features have been added to the PIS-10 stack and ongoing revisions made based on the feedback provided by research centres such as the CSAEMS at the Cape Peninsula University of Technology and other organisations around the world.

The current thesis with the title “**Development of an embedded system actuator node for integration into an IEC 61850 based substation automation application**“, describes the research work that has been done in order to provide a framework and knowledge-base sufficient to overcome challenges concerning the implementation of the IEC 61850 standard on an embedded platform. Consideration has been given to the aspects of IEC 61850 modelling, application software development, a protection scheme distributed between a commercially available IED and the controller developed on the embedded platform, GOOSE frame analysis and performance evaluation in terms of functional and temporal behaviour, and an aspect of cyber security.

7.2 Aims and objectives

From the problem statement expressed in the first chapter, the project aim and objectives were developed and investigated based on the IEC 61850 conceptual modelling together with the SystemCorp communication stack (PIS-10) implementation.

Methods for development of an IEC 61850 compliant software application based on an IEC61850 communication stack and API from SystemCorp were explored. The GOOSE performance of the developed software for a circuit breaker actuator deployed on the DK60 board manufactured by Beck IPC was analysed. The research focus was on the development of embedded software which is IEC 61850 compliant with particular emphasis placed on an overcurrent protection scheme distributed across a commercial IED and an embedded systems platform using GOOSE message communication.

The objectives which have been achieved are listed as follows:

- Literature review: history of IEDs and review of design theory for IED hardware and software.
- Literature review: analysis and comparison of method for IEC 61850 modelling - development and implementation aspects.
- GOOSE frame format analysis utilizing data networking utilities (Wireshark, MMS-Ethereal, and IEDScout).
- Development of a method for IEC 61850 embedded project development and system integration.
- Software development for the process application that is run on the SC143 chip of the DK 60 board.
- Development of a method for GOOSE performance evaluation in order to confirm that the communication stack used as well as the developed process application meet the temporal behaviour requirement specified by the IEC 61850 standard.
- Design, implementation and interfacing of a circuit breaker emulator to the actuator (DK 60 embedded system platform) successfully completed.
- Development of a test scenario for an aspect of cyber security.

7.3 Thesis deliverables

In order to achieve the aim and objectives of this research thesis, the following deliverables are proposed:

7.3.1 Literature review: history of IEDs and review of design theory for IED hardware and software

A search and review of the literature has yielded valuable background information on the chronological evolution of protection technology and substation automation. This was important in order to survey the knowledge-base in the field of Substation Automation Systems and appreciate how constant efforts of innovation over a period of more than a hundred years have contributed to current practice.IED hardware topologies and software algorithms have also been reviewed.

This contextual information was of assistance in delimiting the scope of the current research.

7.3.2 Literature review: analysis and comparison of method for IEC 61850 modelling - development and implementation aspects

A literature review on the analysis and interpretation of UML diagrams relating to a general understanding of the IEC 61850 standard was conducted. A method for the development of an IEC 61850 Data model and services is reviewed as well as the method for the deployment of a distributed protection scheme.

7.3.3 GOOSE frame format analysis utilizing data network utilities

A general study of the format of the Ethernet frame structure was conducted and details particular to the GOOSE PDU were analysed and interpreted: including the ASN.1 encoding and decoding of the hexadecimal data through the use of specialized network analyzer tools such as Wireshark and MMS-Ethereal.

7.3.4 Development of a method for IEC 61850 embedded project development and system integration for the DK 60 board

It was important to develop a step-by-step method showing the development steps one has to go through in order to develop a compliant IEC 61850 software application on a specific embedded system platform. In this method certain activities are proposed, in a certain order, that (a) allows for by-passing the ad-hoc trial and error first-time users encounter, (b) identification of project-specific challenges at an early stage, and (c) project error diagnosis. (see Figure 5.1)

A method for SCL file creation and manipulation in order to enable integration and interoperability between devices from different vendors is explained, based on the use of ICD Designer, SEL Architect, and other software tools. (Section 4.4.4)

7.3.5 Software development for the process application that is run on the SC143 chip of the DK 60 board

Based on experience with other IEC 61850 industrial substation equipment, it appears that the server functionality is associated with the publisher and also with the subscriber mechanism. This is not the case with the specific implementation by SystemCorp on their DK60 embedded platform (as of the time of writing: Oct 2012). The PIS-010 stack from SystemCorp implements server functionality which is associated only with the publisher and the client is associated with the subscriber mechanism. Thus, to have access to full publisher/subscriber functionality a method has to be developed in order to merge the server and the client modules into the same process application source code and have them interact properly. This was successfully achieved, i.e. demonstration of GOOSE communication from a commercial IED to the DK 60-based circuit breaker actuator and from the circuit breaker actuator to other devices on the network was achieved.

The process application software developed accesses and packages the binary information at the boundary between the DK60 board and the emulated circuit breaker. GOOSE messages for protection and status position are now exchanged between the commercial IED and the DK60 board. In the one direction a protection trip signal is sent, and in the other direction breaker position status information is sent via GOOSE message. The protection operations dependant on the GOOSE messaging (sent at the data-link layer level) is time-critical. The process application software developed also allows for the control of the emulated circuit breaker from a desktop computer using the MMS protocol (command sent through the TCP/IP stack) and the communication method known as “direct-with-normal-security”.

This research project demonstrates the use of SystemCorp’s IEC 61850 communication stack (PIS-10.lib) and it’s associated Application Program Interface (API). The contribution of the software developed for the process application can be stated as follows:

- The PIS010 stack does not support array data types, but does support single-bit Boolean data types. Unfortunately the breaker position status requires a value derived from a two-bit word (double point position status - Dbpos) to be inserted into the payload of the GOOSE message. A method was developed for concatenation of the individual Boolean bits so as to link to a structure which holds a value representative of the two-bit value.

- The demonstration program which was supplied with the DK60 board, only supported one data type (Boolean). In order to accommodate the specific design of this project (double point status data and Boolean data), the original database program had to be adapted significantly (up to approximately 90%) in order to accommodate multiple data types in the database. Some of the programming which had to be done related to:
    - Adapting the link between the database and the GOOSE message to accommodate two data types instead of one;
    - The real-time updating of two data types instead of one - to track GOOSE events on the DK60 platform; a typical scenario is the following - Digital I/O points continuously monitored (GOOSE performance affected by pause in this polling loop); status of breaker changes; status change detected and compared to value in historical field in database; computation for new value for double point status; update of the current individual bit value into the historical table within the database; call the “IEC61850_Update” function in the PIS010 stack to update the GOOSE value;
    - The procedure above refers to the case when the event is generated from within the DK60 environment (GOOSE transmitted). A different mechanism is utilized when the event is generated externally (GOOSE received).
- The relevant source code for the process application (GA61850.c) is available in Appendix F, the SCL files (Appendices C and D) and adapted hardware configuration (Appendix E).

[Note: that all developed software functions have been written in the “C” language within the file (GA61850.c) which is available in Appendix F. Other include files that are part of the software development process (e.g. clib.h, stdio.h, mem.h, time.h, IEC61850API.h etc.) can be freely downloaded from the Beck IPC and SystemCorp websites (upon registration)].

The functional response of the implemented circuit breaker actuator was verified in the CSAEMS laboratory and the GOOSE transmission time of 0.975 mS was within the range specified by the IEC 61850 standard (3 mS).

Table 7.1 describes some of the main functions that were used for this research project. An indication is made on the table for software functions that have been adapted or those which have been used as-is.

Table 7.1: Function list identification

In GA61850.C source code (Appendix F)			
Function/ method name	**Description and comments**	**Use for Server module**	**Use for Client module**
PIS-010 Object Management (Specific SystemCorp/ No modification added)			
IEC61850_Create	create a client/server object	☑	☑
IEC61850_LoadSCLfile	Configures the client/server from an SCL file	☑	☑
IEC61850_Start	Starts the services (e.g. GOOSE, MMS etc) within the client/server object.	☑	☑
IEC61850_Stop	Stop the services (e.g. GOOSE, MMS etc) within the client/server Object.	☑	☑
IEC61850_Free	Delete a client/server object	☑	☑
PIS-10 Data Attribute Access. (Unless otherwise mentioned, the following functions have been adapted to suit the project specification)			
IEC61850_Update	Server Function called from the user application indicating a change in the hardware to the IEC 61850 stack. (Have not been modified)	☑	□
My_User_Write [MyWriteFunction]	Function calls backs by the PIS-10 to handle a Write command coming from a client on the Network. Changes a value on the Server	☑	□
My_User_Read [MyReadFunction]	Function calls backs by the PIS-10 to handle a Read command coming from a Client on the network. Requests a Value from theServer	☑	□
My_User_Operate [MyOperateFunction]	Function calls backs by the PIS-10 to handle an Operate command on selected control/s with specified value	☑	□
My_User_Update [MyUpdateFunction]	Function calls backs by the PIS-10 to inform the user application of a change of a value on a published GOOSE issue by a Server on the network	□	☑

Function/ method name	Description	Use for Server module	Use for Client module
PIS-10 Time Function(Specific SystemCorp/ No modification added)			
getTime	get the NTP time	N/A	N/A
PIS-10 Support functions Specific SystemCorp/ No modification added)			
IEC61850_GetLibraryVersion	Returns the version of the PIS-10 library	N/A	N/A
IEC61850_GetLibraryBuildTime	Returns build time for PIS-10 Library	N/A	N/A
Other important functions			
CreateLocalDatabase	Create the process application local data base that will hold the application Object. (Have been adapted)	N/A	N/A
huge InputHandler	This is the function that will be executed continuously. This function has been entirely rewritten. They are different state machines implemented inside as well as the container for circuit breaker Double point position status. Other function are event driven (Have been adapted)	☑	N/A
RTX_Create_Task	This function is used to create and start an executable RTOS task. (Beck- IPC No Change)	N/A	N/A
RTX_Sleep_Time	This function is used to put the execution of the current program on hold. This is very important in multitasking environment. (Beck IPC- No Change)	N/A	N/A

The development of an IEC 61850 application that uses an off-the-shelf (commercial) communication stack helps to reduce the time-to-market for the manufacturing process of IEDs. The complexity of the software development is then reduced to the only the process application interfaced to an existing communication stack.

7.3.6 Development of a method for GOOSE performance evaluation in order to confirm that the communication stack used as well as the developed process application meet the temporal behaviour requirements specified by the IEC 61850 standard.

A method for GOOSE performance evaluation derived from the “inter trip test” has been tested with success. It was necessary to slightly adapt the original “inter trip test” due to the fact that no power interface for hard-wired connection has been developed between the circuit breaker actuator and the test injection device. An acceptable GOOSE transfer time of about 0.975 mS was achieved as compared to the IEC 61850 standard which specifies a transfer time average value of 3 mS for GOOSE messages used for protection applications.

7.3.7 Development of a test scenario for an aspect of cyber security

A test-method to determine the capabilities of an IED to discard a GOOSE packet in case of a network intrusion by an out-of-sequence legacy GOOSE message was developed. This method required the use of advanced networking tools such as "Colasoft Packet Builder".

The next section presents a summary of the challenges encountered in the implementation of the research project and how these were overcome.

7.4 Challenges Encountered

There have been significant - almost insurmountable - challenges that have arisen during the course of this research project implementation. These include:

a) Very little detailed implementation knowledge in the public domain – knowledge resides with vendors only. Almost no implementation work done in the academic sphere that is known of.

b) Very few platforms, tools and forums for the development or adaptation of IEC 61850 solutions in the public domain. Deficiency in documentation, functionality not corresponding with what was marketed, versions of software not being mature and these facts not being communicated to the user community have compounded the problems when doing development, and have hampered progress at all stages of the project.

c) A multi-disciplinary environment – the development of an IEC 61850 embedded application requires knowledge in multiple fields such as data networking, software modelling and development, embedded systems, electrical protection, and system simulation and testing methods. Sufficient familiarity with all these environments to the extent of being able to develop solutions is quite formidable.

d) Interpretation of the standard – a pre-requisite to development or adaptation of any IEC 61850 compliant device requires detailed knowledge of most parts of the standard; of which there are many.

Outside of the main vendor companies which have been mentioned, it has been reported that some (but very few) national power utilities and municipalities and associated companies (e.g. test injection equipment manufacturing companies, and companies manufacturing real-time simulation environments for power systems), have implemented their own IEC 61850 devices. Even the details of these solutions are not in the public domain.

The challenges outlined above clearly show that this project is a unique effort at implementing a solution that would normally only be in the domain of vendors, and thereby contribute to forming a foundation for the interpretation of standards, and determining what is of importance for implementation.

The following has been achieved in this research project: an excellent understanding of the IEC 61850 standard, sufficient familiarity with the multi-disciplinary aspects of data networking, software development, hardware interfacing and systems integration, and, through close dialogue with the developers of the embedded systems board (Beck-IPC) and the developers of the Protocol Integration Stack (SystemCorp), have addressed many of the considerable challenges which have arisen.

The overcurrent case study, the function of which was distributed between a commercial IED (SIEMENS Siprotec device) and the actuator-application developed on an embedded platform for this project (DK60 board). The system development and integration yielded a solution that was in compliance with the IEC 61850 standard and utilized GOOSE (Generic Object-Oriented Substation Event) messaging to successfully achieve both the functional and temporal behaviour required.

7.5 Application of results

The work of this research project includes areas which combine the aspects of data networking, embedded software and hardware development, power system protection principles and substation automation.

The results present the possibility for future research opportunities for embedded development of applications within the IEC 61850 domain.

Specific areas where the results could be applied:

- Possible software and product development for equipment for non-substation environments and potentially new domains e.g.
    - development of logical nodes for new domains e.g. new logical nodes for the hydro and wind domains have already been proposed.
    - development of logical nodes for possible Smartgrid applications
- For further investigation into the understanding and development of process bus devices (actuators and merging units).

7.6 Proposed improvements and ongoing work status

Recommendations that flow from this research work can be stated as follows:

- The current research recommends the adoption of a new communication stack architecture that will allow field devices (Protection relay, Actuator…) to deploy a "Server +Publisher +Subscriber" on one platform under one real-time operating system, independently from a client object. The current architecture forces the user to first create a client object before being able to subscribe to a GOOSE message. It will be recommended that the publisher/subscriber be independent from the client/server. Some efforts still need to be made from a cyber security point of view. Despite the comments above, this research has demonstrated the potential capabilities of the PIS-10 stack from "SystemCorp" to meet and achieve critical IEC 61850 standard specifications.

- In the first edition of the IEC 61850 standard, the mechanism by which a device publishes a GOOSE message is clearly specified and is well known. The subscription mechanism to a GOOSE message is however not specified in Edition 1 of the standard. This situation is responsible for the differences observed while experimenting with various commercial IEDs (SIEMENS, ABB, SEL…) and the embedded platform. The same implementation methodology to publish a GOOSE message is used, but every vendor has their own implementation for the subscription mechanism. These types of issues with respect to the GOOSE subscription mechanism are addressed in IEC 61850 Edition2.The IEC 61850 standard (Edition 2) has now defined a model for subscription by adding the definition of a new Logical Node for GOOSE subscription (LN LGOS). The LN LGOS is defined for the monitoring (or subscription) of GOOSE messages. It will certainly take a bit of time for vendors to adapt their subscription mechanisms to the new IEC 61850 specifications as per Edition 2 of the standard. More information regarding these issues are available at:

http://www.tissues.iec61850.com/tissue.mspx?issueid=831
http://www.tissues.iec61850.com/tissue.mspx?issueid=829

The definition list of new logical nodes as per IEC 61850 Ed.2 can be accessed from:
http://blog.iec61850.com/2012/07/list-of-almost-all-iec-61850-logical.html

Edition 2 of the IEC 61850 standard is now available (published in 2011).Most of the issues reported in Edition 1 has been resolved in Edition2 of the standard.

7.7 Future research work

Future directions for research work include the following:

- Investigate the use of PowerPC and ARM7 microprocessor chip platform for the development of Intelligent Electronic Devices as they can offer higher processing capabilities.
- Explore the possibilities offered by VxWorks Real-Time Operating system as it is extensively used by IED vendors and demonstrate clearly how to enable multitasking/multithreading execution for the considered Real-Time Operating System.
- Investigate the use of IEC 61131-3 (for PLC) or IEC 61499 (for distributed control and automation) together with IEC 61850 (for Communication network and systems for power utility automation-Ed.2) in order to explore the synergy between distributed factory automation and substation automation.
- Investigation of the MMS message structure and its application within the IEC 61850 standard.
- Investigate the development method for “select-before-operate” for an IEC 61850 Server Object (code to be written in C language in order to be compatible with the PIS-10.lib).
- Investigate the development method for “select-before-operate” for an IEC 61850 Client Object for HMI (code to be written in C# language in order to be compatible with the PIS-10.DLL).
- Investigate the performance and system stability of a protection scheme on which both GOOSE and Sampled Value are used on the process bus.

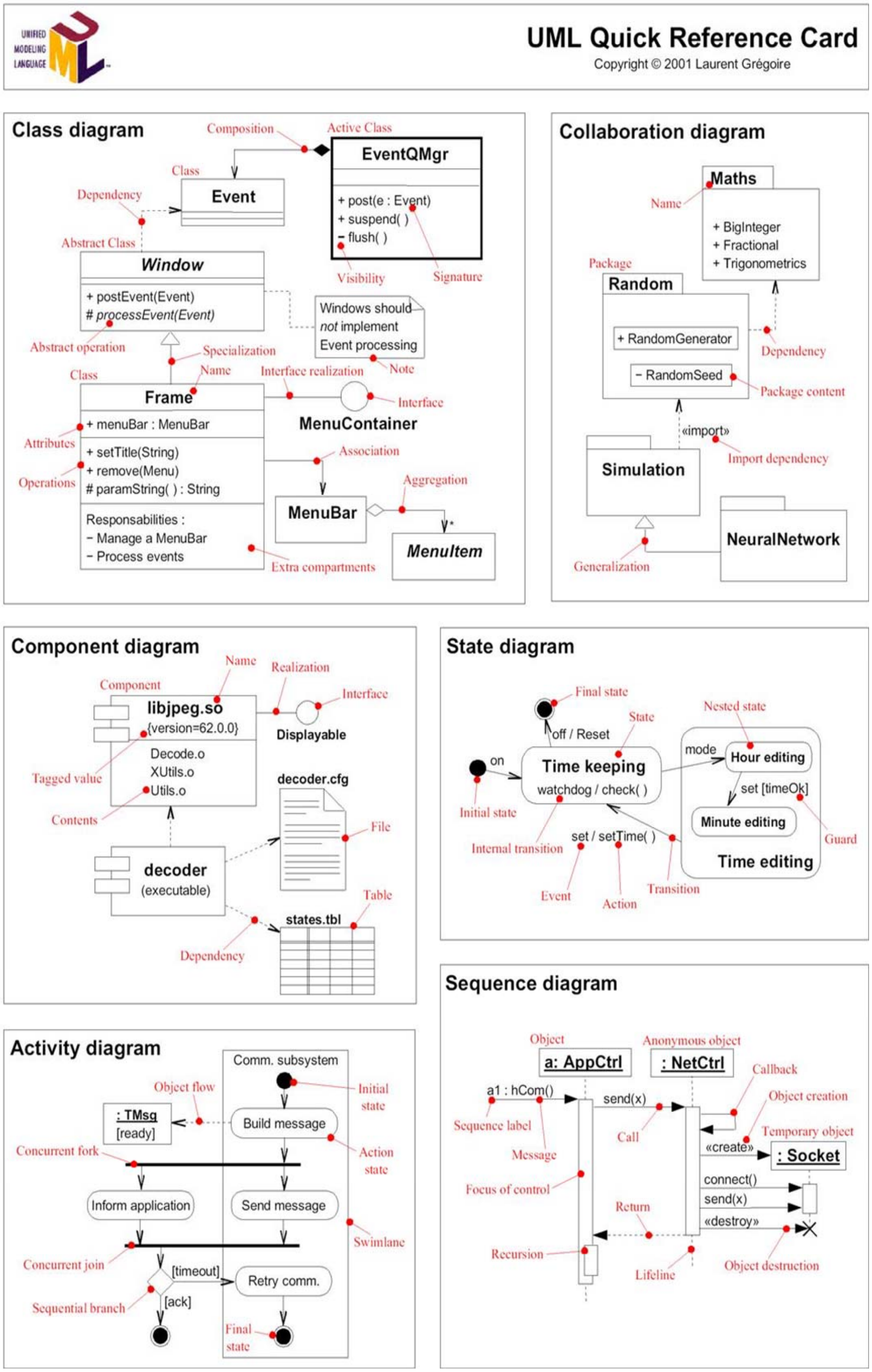
7.8 Publications

1. Retonda, J., Petev, P., Kriger, C., and Behardien, S., 2012. Software development for integration of an embedded system platform into a minimal lab-scale IEC 61850 application. Submitted to SAIEE Research Journal (awaiting confirmation for publication).
2. Kriger, C., Retonda, J., Luwaca, E. and Behardien, S., 2011. Analysis of GOOSE and Sampled Value Message Structure for Educational Purposes. PAC World Conference, Dublin.
3. Retonda, J. and Behardien, S., 2010. Simulation of an IEC 61850 based GOOSE message, and analysis of its structure. South African OMICRON User Conference 2010. Johannesburg. South Africa, 8-9 November 2010.

APPENDICES

APPENDIX A

APPENDIX A: UML representative diagram



APPENDIX B

APPENDIX B: ASN.1 code for GOOSE encoding/decoding

Taken from IEC 61850-8-1 (2004, p.111).

```
IEC61850 DEFINITIONS ::= BEGIN
IMPORTS Data FROM ISO-IEC-9506-2
IEC 61850-8-1 Specific Protocol ::= CHOICE {
gseMngtPdu [APPLICATION 0] IMPLICIT GSEMngtPdu,
goosePdu [APPLICATION 1] IMPLICIT IECGoosePdu,
… }
GSEMngtPdu ::= SEQUENCE {
StateID   [0] IMPLICIT INTEGER,
Security  [3] ANY OPTIONAL, -- reserved for future definition
CHOICE {
requests  [1] IMPLICIT GSEMngtRequests,
responses [2] IMPLICIT GSEMngtResponses
}
}
GSEMngtRequests ::= CHOICE {
getGoReference        [1] IMPLICIT GetReferenceRequestPdu
getGOOSEElementNumber [2] IMPLICIT GetElementRequestPdu,
getGsReference        [3] IMPLICIT GetReferenceRequestPdu,
getGSSEDataOffset     [4] IMPLICIT GetElementRequestPdu,
…
}
GSEMngtResponses ::= CHOICE {
gseMngtNotSupported   [0] IMPLICIT NULL,
getGoReference        [1] IMPLICIT GSEMngtResponsePdu,
getGOOSEElementNumber [2] IMPLICIT GSEMngtResponsePdu,
getGsReference        [3] IMPLICIT GSEMngtResponsePdu,
getGSSEDataOffset     [4] IMPLICIT GSEMngtResponsePdu,
…
}
GetReferenceRequestPdu ::= SEQUENCE {
ident  [0] IMPLICIT VISIBLE-STRING,
-- size shall support up to 65 octets
offset [1] IMPLICIT SEQUENCE OF INTEGER,
…
}
GetElementRequestPdu ::= SEQUENCE {
ident      [0] IMPLICIT VISIBLE-STRING,
-- size shall support up to 65 octets
references [1] IMPLICIT SEQUENCE OF VISIBLE-STRING
…
}
GSEMngtResponsePdu ::= SEQUENCE {
ident    [0] IMPLICIT VISIBLE-STRING,
-- echos the value of the request
confRev [1] IMPLICIT INTEGER OPTIONAL,
CHOICE {
responsePositive [2] IMPLICIT SEQUENCE {
datSet           [0] IMPLICIT VISIBLE_STRING OPTIONAL,
result           [1] IMPLICIT SEQUENCE OF RequestResults
},
responseNegative [3] IMPLICIT GlbErrors
},
…
}
RequestResults::= CHOICE {
offset    [0] IMPLICIT INTEGER,
reference [1] IMPLICIT IA5STRING,
error     [2] IMPLICIT ErrorReason
}
GlbErrors ::= INTEGER {
other(0),
unknownControlBlock(1),
responseTooLarge(2),
controlBlockConfigurationError (3),
…
}
ErrorReason ::= INTEGER {
other (0),
notFound (1),
…
}
IECGoosePdu ::= SEQUENCE {
gocbRef           [0] IMPLICIT VISIBLE-STRING,
timeAllowedtoLive [1] IMPLICIT INTEGER,
datSet            [2] IMPLICIT VISIBLE-STRING,
goID              [3] IMPLICIT VISIBLE-STRING OPTIONAL,
t                 [4] IMPLICIT UtcTime,
stNum             [5] IMPLICIT INTEGER,
sqNum             [6] IMPLICIT INTEGER,
test              [7] IMPLICIT BOOLEAN DEFAULT FALSE,
confRev           [8] IMPLICIT INTEGER,
ndsCom            [9] IMPLICIT BOOLEAN DEFAULT FALSE,
numDatSetEntries [10] IMPLICIT INTEGER,
allData          [11] IMPLICIT SEQUENCE OF Data,
security         [12] ANY OPTIONAL,
-- reserved for digital signature
}
UtcTime ::= OCTETSTRING -- format and size defined in 8.1.3.6.

END
```

APPENDIX C

APPENDIX C: DK60SV1.ICD

```
<?xml version="1.0" encoding="UTF-8"?>
<SCL xmlns="http://www.iec.ch/61850/2003/SCL">
   <Header id="" version="1" nameStructure="IEDName"/>
   <Communication>
      <SubNetwork name="SubNetworkName">
         <ConnectedAP iedName="DK60" apName="SubstationRing1">
            <Address>
               <P type="OSI-AP-Title">1,1,1,999,1</P>
               <P type="OSI-AE-Qualifier">12</P>
               <P type="OSI-PSEL">00000001</P>
               <P type="OSI-SSEL">0001</P>
               <P type="OSI-TSEL">0001</P>
               <P type="IP">172.16.10.5</P>
               <P type="IP-SUBNET">255.255.255.0</P>
               <P type="IP-GATEWAY">172.16.10.4</P>
            </Address>
            <GSE ldInst="LDevice1" cbName="GSE_CB_GOOSE">
               <Address>
                  <P type="VLAN-ID">0</P>
                  <P type="VLAN-PRIORITY">4</P>
                  <P type="MAC-Address">01-0c-cd-01-00-00</P>
                  <P type="APPID">0</P>
               </Address>
            </GSE>
         </ConnectedAP>
      </SubNetwork>
   </Communication>
   <IED type="Circuit Breaker" manufacturer="CPUT" configVersion="1"
name="DK60">
      <Services>
         <DynAssociation/>
         <GetDirectory/>
         <GetDataObjectDefinition/>
         <DataObjectDirectory/>
         <GetDataSetValue/>
         <SetDataSetValue/>
         <DataSetDirectory/>
         <GetCBValues/>
         <GOOSE client="true" max="10"/>
         <ConfDataSet modify="true" maxAttributes="0" max="0"/>
         <ReadWrite/>
         <TimerActivatedControl/>
         <ConfReportControl max="0"/>
         <ConfLogControl max="0"/>
         <GSEDir/>
         <GSSE max="0"/>
         <FileHandling/>
      </Services>
      <AccessPoint name="SubstationRing1">
         <Server timeout="30">
            <Authentication none="true"/>
            <LDevice inst="LDevice1">
               <LN0 lnClass="LLN0" inst="" lnType="LLN0_0">
                  <DataSet name="GOOSE_Pos">
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO"
lnInst="1" doName="Ind1" daName="stVal" fc="ST"/>
                     <FCDA ldInst="LDevice1" prefix="DIPS_" lnClass="GGIO"
lnInst="1" doName="Ind2" daName="stVal" fc="ST"/>
                     <FCDA ldInst="LDevice1" lnClass="XCBR" lnInst="1"
doName="Pos" daName="stVal" fc="ST"/>
```

```
                     <FCDA ldInst="LDevice1" lnClass="XCBR" lnInst="1"
doName="Pos" daName="q" fc="ST"/>
                  </DataSet>
                  <DataSet name="Alarm_DataSet">
                     <FCDA ldInst="LDevice1" lnClass="XCBR" lnInst="1"
doName="Pos" daName="stVal" fc="ST"/>
                  </DataSet>
                  <ReportControl rptID="DK60_RepURCB_ID" confRev="1"
intgPd="0" datSet="Alarm_DataSet" name="UNBUFFERED_RCB" desc="Unbuf RCB">
                     <TrgOps dchg="true" qchg="true" dupd="true"/>
                     <OptFields/>
                  </ReportControl>
                  <DOI name="Mod">
                     <DAI name="stVal" valKind="Set">
                        <Val sGroup="0">on</Val>
                     </DAI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val sGroup="0">status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="NamPlt">
                     <DAI name="vendor">
                        <Val>CPUT</Val>
                     </DAI>
                     <DAI name="swRev">
                        <Val>0.0</Val>
                     </DAI>
                     <DAI name="d">
                        <Val>Circuit breaker  Actuator</Val>
                     </DAI>
                     <DAI name="configRev">
                        <Val></Val>
                     </DAI>
                  </DOI>
                  <GSEControl type="GOOSE" appID="99" confRev="1"
datSet="GOOSE_Pos" name="GSE_CB_GOOSE"/>
               </LN0>
               <LN lnClass="LPHD" inst="1" prefix="" lnType="LPHD_0">
                  <DOI name="PhyNam">
                     <DAI name="vendor">
                        <Val>CPUT</Val>
                     </DAI>
                     <DAI name="hwRev">
                        <Val></Val>
                     </DAI>
                     <DAI name="swRev">
                        <Val></Val>
                     </DAI>
                     <DAI name="serNum">
                        <Val></Val>
                     </DAI>
                     <DAI name="model">
                        <Val></Val>
                     </DAI>
                     <DAI name="location">
                        <Val></Val>
                     </DAI>
                     <DAI name="cdcNs">
                        <Val></Val>
                     </DAI>
                     <DAI name="cdcName">
                        <Val></Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val></Val>
                     </DAI>
                  </DOI>
                  <DOI name="Proxy">
                     <DAI name="d">
                        <Val></Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="GGIO" inst="1" prefix="DIPS_" lnType="GGIO_0">
                  <DOI name="Mod">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="NamPlt">
                     <DAI name="vendor">
                        <Val>CPUT</Val>
                     </DAI>
                     <DAI name="swRev">
                        <Val>0.0</Val>
                     </DAI>
                     <DAI name="d">
                        <Val>Binary inputs</Val>
                     </DAI>
                     <DAI name="configRev">
                        <Val></Val>
                     </DAI>
                  </DOI>
                  <DOI name="Ind1">
                     <DAI name="d" valKind="Set">
                        <Val sGroup="0"></Val>
                     </DAI>
                     <DAI name="stVal">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="1" Field3="1"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="q">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="1" Field3="2"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="t">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="1" Field3="3"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                  </DOI>
                  <DOI name="Ind2">
                     <DAI name="d" valKind="Set">
                        <Val sGroup="0">0</Val>
                     </DAI>
                     <DAI name="stVal" valKind="Set">
                        <Private type="SystemCorp_Generic">
```

```
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="1" Field3="1"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="q">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="1" Field3="2"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="t">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="2" Field2="1" Field3="3"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="XCBR" inst="1" prefix="" lnType="XCBR_1">
                  <DOI name="Pos">
                     <SDI name="Oper">
                        <DAI name="ctlVal" valKind="Set">
                           <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="2" Field3="1"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                        </DAI>
                     </SDI>
                     <DAI name="stVal" valKind="Set">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="1" Field3="1"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="q" valKind="Set">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="1" Field3="2"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="t" valKind="Set">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="3" Field2="1" Field3="3"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val sGroup="0">direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Mod">
                     <DAI name="stVal" valKind="Set">
                        <Val sGroup="0">on</Val>
                     </DAI>
                     <DAI name="ctlModel" valKind="Set">
                        <Val sGroup="0">status-only</Val>
                     </DAI>
                  </DOI>
               </LN>
            </LDevice>
         </Server>
      </AccessPoint>
   </IED>
   <DataTypeTemplates>
      <LNodeType lnClass="LLN0" id="LLN0_0">
         <DO name="Mod" type="INC_0"/>
         <DO name="Beh" type="INS_0"/>
         <DO name="Health" type="INS_1"/>
         <DO name="NamPlt" type="LPL_0"/>
      </LNodeType>
      <LNodeType lnClass="LPHD" id="LPHD_0">
         <DO name="PhyHealth" type="INS_2"/>
         <DO name="PhyNam" type="DPL_0"/>
         <DO name="Proxy" type="SPS_0"/>
      </LNodeType>
      <LNodeType lnClass="GGIO" id="GGIO_0">
         <DO name="Mod" type="INC_0"/>
         <DO name="Beh" type="INS_0"/>
         <DO name="Health" type="INS_1"/>
         <DO name="NamPlt" type="LPL_0"/>
         <DO name="Ind1" type="SPS_0"/>
         <DO name="Ind2" type="SPS_1"/>
      </LNodeType>
      <LNodeType lnClass="XCBR" id="XCBR_1">
         <DO name="Mod" type="ENC_0"/>
         <DO name="Beh" type="ENS_0"/>
         <DO name="Health" type="ENS_0"/>
         <DO name="NamPlt" type="LPL_1"/>
         <DO name="Loc" type="SPS_2"/>
         <DO name="OpCnt" type="INS_5"/>
         <DO name="Pos" type="DPC_1"/>
         <DO name="BlkOpn" type="SPC_1"/>
         <DO name="BlkCls" type="SPC_1"/>
         <DO name="CBOpCap" type="INS_5"/>
      </LNodeType>
      <DOType cdc="INC" id="INC_0">
         <DA fc="ST" name="stVal" bType="Enum" type="Mod"/>
         <DA fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
      </DOType>
      <DOType cdc="INS" id="INS_0">
         <DA fc="ST" name="stVal" bType="Enum" type="Beh"/>
         <DA fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="INS" id="INS_1">
         <DA fc="ST" name="stVal" bType="Enum" type="Health"/>
         <DA fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="LPL" id="LPL_0">
         <DA fc="DC" name="vendor" bType="VisString255"/>
         <DA fc="DC" name="swRev" bType="VisString255"/>
         <DA fc="DC" name="d" bType="VisString255"/>
         <DA fc="DC" name="configRev" bType="VisString255"/>
      </DOType>
      <DOType cdc="SPS" id="SPS_0">
         <DA fc="ST" name="stVal" bType="BOOLEAN"/>
```

```
         <DA fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="DC" name="d" bType="VisString255"/>
      </DOType>
      <DOType cdc="INS" id="INS_2">
         <DA fc="ST" name="stVal" bType="INT8"/>
         <DA fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="DPL" id="DPL_0">
         <DA fc="DC" name="vendor" bType="VisString255"/>
         <DA fc="DC" name="hwRev" bType="VisString255"/>
         <DA fc="DC" name="swRev" bType="VisString255"/>
         <DA fc="DC" name="serNum" bType="VisString255"/>
         <DA fc="DC" name="model" bType="VisString255"/>
         <DA fc="DC" name="location" bType="VisString255"/>
         <DA fc="EX" name="cdcNs" bType="VisString255"/>
         <DA fc="EX" name="cdcName" bType="VisString255"/>
         <DA fc="EX" name="dataNs" bType="VisString255"/>
      </DOType>
      <DOType cdc="SPS" id="SPS_1" desc="Single point status">
         <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="DC" name="d" bType="VisString255"/>
      </DOType>
      <DOType cdc="ENC" id="ENC_0" desc="Controllable integer status">
         <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Mod"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA dchg="true" fc="CF" name="ctlModel" bType="Enum"
type="ctlModel"/>
      </DOType>
      <DOType cdc="ENS" id="ENS_0" desc="Enumerated status">
         <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Mod"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="LPL" id="LPL_1" desc="Logical Node name plate">
         <DA fc="DC" name="vendor" bType="VisString255"/>
         <DA fc="DC" name="swRev" bType="VisString255"/>
         <DA fc="DC" name="d" bType="VisString255"/>
      </DOType>
      <DOType cdc="SPS" id="SPS_2" desc="Single point status">
         <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="INS" id="INS_5" desc="Integer status">
         <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="DPC" id="DPC_1" desc="Controllable double point">
         <DA fc="CO" name="Oper" bType="Struct" type="SBOwDbpos_0"/>
         <DA dchg="true" fc="ST" name="stVal" bType="Dbpos"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
      </DOType>
      <DOType cdc="SPC" id="SPC_1" desc="Controllable single point">
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
      </DOType>
```

```
<DAType id="Originator_0">
   <BDA name="orCat" bType="Enum" type="orCat"/>
   <BDA name="orIdent" bType="Octet64"/>
</DAType>
<DAType id="SBOwDbpos_0">
   <BDA name="ctlVal" bType="BOOLEAN"/>
   <BDA name="origin" bType="Struct" type="Originator_0"/>
   <BDA name="ctlNum" bType="INT8U"/>
   <BDA name="T" bType="Timestamp"/>
   <BDA name="Test" bType="BOOLEAN"/>
   <BDA name="Check" bType="Check"/>
</DAType>
<EnumType id="ctlModel">
   <EnumVal ord="0">status-only</EnumVal>
   <EnumVal ord="1">direct-with-normal-security</EnumVal>
   <EnumVal ord="2">sbo-with-normal-security</EnumVal>
   <EnumVal ord="3">direct-with-enhanced-security</EnumVal>
   <EnumVal ord="4">sbo-with-enhanced-security</EnumVal>
</EnumType>
<EnumType id="orCat">
   <EnumVal ord="0">not-supported</EnumVal>
   <EnumVal ord="1">bay-control</EnumVal>
   <EnumVal ord="2">station-control</EnumVal>
   <EnumVal ord="3">remote-control</EnumVal>
   <EnumVal ord="4">automatic-bay</EnumVal>
   <EnumVal ord="5">automatic-station</EnumVal>
   <EnumVal ord="6">automatic-remote</EnumVal>
   <EnumVal ord="7">maintenance</EnumVal>
   <EnumVal ord="8">process</EnumVal>
</EnumType>
<EnumType id="sboClass">
   <EnumVal ord="0">operate-once</EnumVal>
   <EnumVal ord="1">operate-many</EnumVal>
</EnumType>
<EnumType id="Beh">
   <EnumVal ord="1">on</EnumVal>
   <EnumVal ord="2">blocked</EnumVal>
   <EnumVal ord="3">test</EnumVal>
   <EnumVal ord="4">test/blocked</EnumVal>
   <EnumVal ord="5">off</EnumVal>
</EnumType>
<EnumType id="Mod">
   <EnumVal ord="1">on</EnumVal>
   <EnumVal ord="2">blocked</EnumVal>
   <EnumVal ord="3">test</EnumVal>
   <EnumVal ord="4">test/blocked</EnumVal>
   <EnumVal ord="5">off</EnumVal>
</EnumType>
<EnumType id="CBOpCap">
   <EnumVal ord="1">None</EnumVal>
   <EnumVal ord="2">Open</EnumVal>
   <EnumVal ord="3">Close-Open</EnumVal>
   <EnumVal ord="4">Open-Close-Open</EnumVal>
   <EnumVal ord="5">Close-Open-Close-Open</EnumVal>
</EnumType>
<EnumType id="Health">
   <EnumVal ord="1">Ok</EnumVal>
   <EnumVal ord="2">Warning</EnumVal>
   <EnumVal ord="3">Alarm</EnumVal>
</EnumType>
<EnumType id="Check">
   <EnumVal ord="0">no-check</EnumVal>
   <EnumVal ord="1">synchrocheck</EnumVal>
```

```
         <EnumVal ord="2">interlocking-check</EnumVal>
         <EnumVal ord="3">both</EnumVal>
      </EnumType>
      <EnumType id="Dbpos">
         <EnumVal ord="0">intermediate-state</EnumVal>
         <EnumVal ord="1">off</EnumVal>
         <EnumVal ord="2">on</EnumVal>
         <EnumVal ord="3">bad-state</EnumVal>
      </EnumType>
      <EnumType id="orCategory">
         <EnumVal ord="0">not-supported</EnumVal>
         <EnumVal ord="1">bay-control</EnumVal>
         <EnumVal ord="2">station-control</EnumVal>
         <EnumVal ord="3">remote-control</EnumVal>
         <EnumVal ord="4">automatic-bay</EnumVal>
         <EnumVal ord="5">automatic-station</EnumVal>
         <EnumVal ord="6">automatic-remote</EnumVal>
         <EnumVal ord="7">maintenance</EnumVal>
         <EnumVal ord="8">process</EnumVal>
      </EnumType>
   </DataTypeTemplates>
</SCL>
```

APPENDIX D

APPENDIX D: DK60CL1.SCD

```
<?xml version="1.0" encoding="UTF-8"?>
<SCL xmlns="http://www.iec.ch/61850/2003/SCL"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <Private type="Siemens-SclLib-Version">V02.11.00</Private>
   <Private type="Siemens-SclLib-EditTime">10.06.2012 15:01:28</Private>
   <Private type="Siemens-CRC">-1817809451</Private>
   <Header id="" nameStructure="IEDName">
      <Text>IEC61850 station</Text>
      <History>
         <Hitem version="1" revision="1" when="Thursday, June 07, 2012
6:33:42 PM" who="Licenced DIGSI user: cputJohn" what="Icd-
Import:D:\Siemens\Digsi4\D4PROJ\J_Over_1\P7DI\GV\SD\00000001\en100par.icd
as IED_0002" why="Automatic Digsi IEC61850 Configurator Startup"></Hitem>
         <Hitem version="1" revision="2" when="Thursday, June 07, 2012
6:33:42 PM" who="Licenced DIGSI user: cputJohn" what="Icd-
Import:D:\Siemens\Digsi4\D4PROJ\J_Over_1\P7DI\GV\OD\00000001\otherdev.icd
as DK60" why="Automatic Digsi IEC61850 Configurator Startup"></Hitem>
         <Hitem version="1" revision="3" when="Thursday, June 07, 2012
6:33:45 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="4" when="Thursday, June 07, 2012
6:35:43 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="5" when="Thursday, June 07, 2012
6:38:14 PM" who="Licenced DIGSI user: cputJohn" what="Icd-
Import:D:\Siemens\Digsi4\D4PROJ\J_Over_1\P7DI\GV\OD\00000002\otherdev.icd
as DK60" why="Automatic Digsi IEC61850 Configurator Startup"></Hitem>
         <Hitem version="1" revision="6" when="Thursday, June 07, 2012
6:38:17 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="7" when="Thursday, June 07, 2012
6:52:15 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="8" when="Thursday, June 07, 2012
7:01:35 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="9" when="Thursday, June 07, 2012
7:02:04 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="10" when="Thursday, June 07, 2012
7:12:01 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="11" when="Thursday, June 07, 2012
7:37:50 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="12" when="Thursday, June 07, 2012
7:40:55 PM" who="Licenced DIGSI user: cputJohn" what="Icd-
Import:D:\Siemens\Digsi4\D4PROJ\J_Over_1\P7DI\GV\SD\00000002\en100par.icd
as IED_0002" why="Automatic Digsi IEC61850 Configurator Startup"></Hitem>
         <Hitem version="1" revision="13" when="Thursday, June 07, 2012
7:40:58 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="14" when="Thursday, June 07, 2012
7:43:09 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="15" when="Thursday, June 07, 2012
7:43:24 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
```

```
         <Hitem version="1" revision="16" when="Thursday, June 07, 2012
8:20:31 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="17" when="Thursday, June 07, 2012
8:29:18 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="18" when="Friday, June 08, 2012
4:53:36 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="19" when="Sunday, June 10, 2012
1:52:55 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
         <Hitem version="1" revision="20" when="Sunday, June 10, 2012
3:01:14 PM" who="Licenced DIGSI user: cputJohn" what="Icd-
Update:D:\Siemens\Digsi4\D4PROJ\J_Over_1\P7DI\GV\SD\00000002\en100par.icd
as IED_1" why="Automatic Digsi IEC61850 Configurator Startup"></Hitem>
         <Hitem version="1" revision="21" when="Sunday, June 10, 2012
3:01:28 PM" who="Licenced DIGSI user: cputJohn" what="SCD-file saved to
disk." why="Digsi IEC61850 Configurator User Action"></Hitem>
      </History>
   </Header>
   <Substation name="IEC61850 station">
      <VoltageLevel name="Voltage level1">
         <Bay name="Field1"/>
      </VoltageLevel>
   </Substation>
   <Communication>
      <Private type="Siemens-AppId-Correction_Fully">true</Private>
      <SubNetwork name="SubNetworkName">
         <Text></Text>
         <Private type="Siemens-Start-Address">172.16.0.1</Private>
         <Private type="Siemens-Default-Subnet-
Mask">255.255.255.0</Private>
         <Private type="Siemens-Application">GOOSE
application1||0001||PriorityLow||10|2000|000|4</Private>
         <ConnectedAP iedName="MyClient" apName="P1">
            <Address>
               <P type="OSI-AP-Title">1,1,1,999,1</P>
               <P type="OSI-AE-Qualifier">12</P>
               <P type="OSI-PSEL">00000001</P>
               <P type="OSI-SSEL">0001</P>
               <P type="OSI-TSEL">0001</P>
               <P type="IP">172.16.10.5</P>
               <P type="IP-SUBNET">255.255.255.0</P>
               <P type="IP-GATEWAY">172.16.10.4</P>
            </Address>
         </ConnectedAP>
         <ConnectedAP iedName="IED_1" apName="P1">
            <Private type="Siemens-Application-GSEControl">GOOSE
application1|Control_DataSet1</Private>
            <Address>
               <P type="IP">172.16.10.2</P>
               <P type="IP-SUBNET">255.255.255.0</P>
               <P type="OSI-AP-Title">1,3,9999,23</P>
               <P type="OSI-AE-Qualifier">23</P>
               <P type="OSI-PSEL">00000001</P>
               <P type="OSI-SSEL">0001</P>
               <P type="OSI-TSEL">0001</P>
               <P type="IP-GATEWAY">172.16.0.254</P>
               <P type="S-Profile">1</P>
            </Address>
            <GSE ldInst="PROT" cbName="Control_DataSet1">
               <Address>
```

```
                  <P type="VLAN-ID">000</P>
                  <P type="VLAN-PRIORITY">4</P>
                  <P type="MAC-Address">01-0C-CD-01-00-00</P>
                  <P type="APPID">0001</P>
               </Address>
               <MinTime unit="s" multiplier="m">10</MinTime>
               <MaxTime unit="s" multiplier="m">2000</MaxTime>
            </GSE>
         </ConnectedAP>
      </SubNetwork>
   </Communication>
   <IED name="MyClient">
      <AccessPoint name="P1">
         <LN lnClass="GGIO" inst="1" prefix="" lnType="not_needed"/>
      </AccessPoint>
   </IED>
   <IED type="Siprotec-7SDx" manufacturer="SIEMENS" configVersion="1.0"
name="IED_1" desc="7SD533 V4.6">
      <Private type="Siemens-Increment">2</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">PROT\LLN0\brcbA|0</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">PROT\LLN0\brcbB|0</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">MEAS\LLN0\brcbA|0</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">MEAS\LLN0\brcbB|0</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">DR\LLN0\brcbA|0</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">DR\LLN0\brcbB|0</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">CTRL\LLN0\brcbA|100</Private>
      <Private type="Siemens-BRCB-
StoragePercentageMemoryForBuffer">CTRL\LLN0\brcbB|0</Private>
      <Private type="Siemens-NetworkFrequency">50 Hz</Private>
      <Private type="Siemens-s7ManagerName">7SD533 V4.6</Private>
      <Private type="Siemens-ICD-Language">en-GB</Private>
      <Private type="Siemens-IED-Id">IED_3</Private>
      <Services>
         <DynAssociation/>
         <GetDirectory/>
         <GetDataObjectDefinition/>
         <DataObjectDirectory/>
         <GetDataSetValue/>
         <DataSetDirectory/>
         <ConfDataSet modify="true" maxAttributes="60" max="15"/>
         <DynDataSet maxAttributes="60" max="15"/>
         <ReadWrite/>
         <ConfReportControl max="10"/>
         <GetCBValues/>
         <ReportSettings rptID="Dyn" optFields="Dyn" bufTime="Dyn"
trgOps="Dyn" intgPd="Dyn" cbName="Fix" datSet="Dyn"/>
         <GOOSE client="true" max="16"/>
         <FileHandling/>
         <ConfLNs fixPrefix="true" fixLnInst="true"/>
      </Services>
      <AccessPoint name="P1">
         <Server timeout="30">
            <Authentication none="true"/>
            <LDevice inst="PROT" desc="Protection">
               <LN0 lnClass="LLN0" inst="" lnType="IED_1/PROT/LLN0"
desc="General">
```

```
                  <DataSet name="DataSet1">
                     <FCDA ldInst="PROT" prefix="" lnClass="PTRC"
lnInst="1" doName="Tr" daName="q" fc="ST"/>
                     <FCDA ldInst="PROT" prefix="" lnClass="PTRC"
lnInst="1" doName="Tr" daName="general" fc="ST"/>
                  </DataSet>
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel" valKind="Set">
                        <Val sGroup="0">status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>160412115234078</Val>
                     </DAI>
                     <DAI name="ldNs">
                        <Val>IEC 61850-7-4:2003(E)</Val>
                     </DAI>
                  </DOI>
                  <GSEControl type="GOOSE" appID="0" confRev="1"
datSet="DataSet1" name="Control_DataSet1"/>
               </LN0>
               <LN lnClass="LPHD" inst="1" lnType="IED_1/PROT/LPHD1"
desc="Device">
                  <DOI name="PhyNam" desc="Info">
                     <DAI name="cdcNs">
                        <Val>7SDx IED Configuration Description File
scdConfigRev same Revision as the Station Configuration Description</Val>
                     </DAI>
                  </DOI>
                  <DOI name="PhyHealth" desc="State"/>
                  <DOI name="Proxy" desc="Proxy"/>
               </LN>
               <LN lnClass="PDIS" inst="1" lnType="IED_1/PROT/PDIS1"
desc="Dist Zone Z1">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="StrAG" desc="Pickup L1-E">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrBG" desc="Pickup L2-E">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrCG" desc="Pickup L3-E">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrAB" desc="Pickup L12">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrBC" desc="Pickup L23">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrCA" desc="Pickup L31">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PDIS" inst="2" lnType="IED_1/PROT/PDIS2"
desc="Dist Zone Z2">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PDIS" inst="3" prefix=""
lnType="IED_1/PROT/PDIS2" desc="Dist Zone Z3">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PDIS" inst="4" prefix=""
lnType="IED_1/PROT/PDIS2" desc="Dist Zone Z4">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
```

```
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PDIS" inst="5" lnType="IED_1/PROT/PDIS2"
desc="Dist Zone Z5">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PDIS" inst="10" lnType="IED_1/PROT/PDIS1"
desc="Dist Zone Z1B">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="StrAG" desc="Pickup L1-E">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrBG" desc="Pickup L2-E">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrCG" desc="Pickup L3-E">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrAB" desc="Pickup L12">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrBC" desc="Pickup L23">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrCA" desc="Pickup L31">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PTOC" inst="1" prefix=""
lnType="IED_1/PROT/PTOC1" desc="IDMT Ip">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PTOC" inst="2" prefix=""
lnType="IED_1/PROT/PTOC1" desc="DMT I&gt;">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PTOC" inst="3" prefix=""
lnType="IED_1/PROT/PTOC1" desc="DMT I&gt;&gt;">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PTOC" inst="4" lnType="IED_1/PROT/PTOC1"
desc="DMT I&gt;&gt;&gt;">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
```

```
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PTRC" inst="1" prefix=""
lnType="IED_1/PROT/PTRC1" desc="Total">
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Tr" desc="TRIP">
                     <DAI name="general" valKind="Set">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="2" Field3="1"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                     <DAI name="q" valKind="Set">
                        <Private type="SystemCorp_Generic">
<SystemCorp_Generic:GenericPrivateObject Field1="1" Field2="2" Field3="2"
Field4="0" Field5="0"
xmlns:SystemCorp_Generic="http://www.systemcorp.com.au/61850/SCL/Generic"/>
</Private>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="FinTr" desc="AR fin OFF">
                     <DAI name="d">
                        <Val>FinTr stands for Auto reclose Lockout</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Mod">
                     <DAI name="ctlModel" valKind="Set">
                        <Val sGroup="0">status-only</Val>
                     </DAI>
                     <DAI name="stVal" valKind="Set">
                        <Val sGroup="0">on</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="PTRC" inst="2" lnType="IED_1/PROT/PTRC2"
desc="Dist general">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Op" desc="TRIP"/>
                  <DOI name="Str" desc="Pickup"/>
               </LN>
               <LN lnClass="XCBR" inst="2" lnType="IED_1/PROT/XCBR2"
desc="CB L1">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SumSwARs" desc="Sum I">
                     <DAI name="d">
                        <Val>SumSwARs stands for Accumulation of
interrupted current L1</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="CBOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="XCBR" inst="3" lnType="IED_1/PROT/XCBR2"
desc="CB L2">
                  <DOI name="Mod" desc="Mode">
```

```
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter">
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SumSwARs" desc="Sum I">
                     <DAI name="d">
                        <Val>SumSwARs1 stands for Accumulation of
interrupted current L2</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="CBOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="XCBR" inst="4" lnType="IED_1/PROT/XCBR2"
desc="CB L3">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter">
```

```
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SumSwARs" desc="Sum I">
                     <DAI name="d">
                        <Val>SumSwARs2 stands for Accumulation of
interrupted current L3</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="CBOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="PDIF" inst="1" lnType="IED_1/PROT/PDIF1"
desc="DIFF I&gt;">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PDIF" inst="2" lnType="IED_1/PROT/PDIF1"
desc="DIFF I&gt;&gt;">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="Op" desc="TRIP"/>
               </LN>
               <LN lnClass="PTRC" inst="3" lnType="IED_1/PROT/PTRC3"
desc="DIFF general">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Op" desc="TRIP"/>
                  <DOI name="Str" desc="Pickup"/>
                  <DOI name="StrA" desc="Pickup L1">
                     <DAI name="d">
                        <Val>StrA stands for Start Phase A</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrAG" desc="Pickup L1E">
                     <DAI name="d">
                        <Val>StrAG stands for Start Phase AG</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrB" desc="Pickup L2">
                     <DAI name="d">
                        <Val>StrB stands for Start Phase B</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrBG" desc="Pickup L2E">
                     <DAI name="d">
                        <Val>StrBG stands for Start Phase BG</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrC" desc="Pickup L3">
                     <DAI name="d">
                        <Val>StrC stands for Start Phase C</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="StrCG" desc="Pickup L3E">
                     <DAI name="d">
```

```
      <Val>StrCG stands for Start Phase CG</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrAB" desc="Pickup L12">
   <DAI name="d">
      <Val>StrAB stands for Start Phase AB</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrBC" desc="Pickup L23">
   <DAI name="d">
      <Val>StrBC stands for Start Phase BC</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrCA" desc="Pickup L31">
   <DAI name="d">
      <Val>StrCA stands for Start Phase CA</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrABC" desc="Pickup L123">
   <DAI name="d">
      <Val>StrABC stands for Start Phase ABC</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrABG" desc="Pickup L12E">
   <DAI name="d">
      <Val>StrABG stands for Start Phase ABG</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrBCG" desc="Pickup L23E">
   <DAI name="d">
      <Val>StrBCG stands for Start Phase BCG</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrCAG" desc="Pickup L31E">
   <DAI name="d">
      <Val>StrCAG stands for Start Phase CAG</Val>
   </DAI>
   <DAI name="dataNs">
      <Val>7SDx IED Configuration Description File</Val>
   </DAI>
</DOI>
<DOI name="StrABCG" desc="Pickup L123E">
```

```
                     <DAI name="d">
                        <Val>StrABCG stands for Start Phase ABCG</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
               </LN>
            </LDevice>
            <LDevice inst="MEAS" desc="Measurement">
               <LN0 lnClass="LLN0" inst="" lnType="IED_1/MEAS/LLN0"
desc="General">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>160412115234078</Val>
                     </DAI>
                     <DAI name="ldNs">
                        <Val>IEC 61850-7-4:2003(E)</Val>
                     </DAI>
                  </DOI>
               </LN0>
               <LN lnClass="LPHD" inst="1" lnType="IED_1/PROT/LPHD1"
desc="Device">
                  <DOI name="PhyNam" desc="Info">
                     <DAI name="cdcNs">
                        <Val>7SDx IED Configuration Description File
scdConfigRev same Revision as the Station Configuration Description</Val>
                     </DAI>
                  </DOI>
                  <DOI name="PhyHealth" desc="State"/>
                  <DOI name="Proxy" desc="Proxy"/>
               </LN>
               <LN lnClass="MMTR" inst="1" lnType="IED_1/MEAS/MMTR1"
desc="Energy counter">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SupWh" desc="Wp output"/>
                  <DOI name="SupVArh" desc="Wq output"/>
                  <DOI name="DmdWh" desc="Wp Input"/>
                  <DOI name="DmdVArh" desc="Wq Input"/>
               </LN>
               <LN lnClass="MMXU" inst="1" lnType="IED_1/MEAS/MMXU1"
desc="Op meas value">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
```

```
   </DAI>
</DOI>
<DOI name="Beh" desc="Behaviour"/>
<DOI name="Health" desc="State"/>
<DOI name="NamPlt" desc="Info">
   <DAI name="configRev">
      <Val>210404153620012</Val>
   </DAI>
</DOI>
<DOI name="TotW" desc="P">
   <DAI name="db">
      <Val>10000</Val>
   </DAI>
</DOI>
<DOI name="TotVAr" desc="Q">
   <DAI name="db">
      <Val>10000</Val>
   </DAI>
</DOI>
<DOI name="TotVA" desc="S">
   <DAI name="db">
      <Val>10000</Val>
   </DAI>
</DOI>
<DOI name="TotPF" desc="cos">
   <DAI name="db">
      <Val>10000</Val>
   </DAI>
</DOI>
<DOI name="Hz" desc="f">
   <DAI name="db">
      <Val>100</Val>
   </DAI>
</DOI>
<DOI name="PPV" desc="U">
   <SDI name="phsAB">
      <DAI name="db">
         <Val>2000</Val>
      </DAI>
   </SDI>
   <SDI name="phsBC">
      <DAI name="db">
         <Val>2000</Val>
      </DAI>
   </SDI>
   <SDI name="phsCA">
      <DAI name="db">
         <Val>2000</Val>
      </DAI>
   </SDI>
</DOI>
<DOI name="PhV" desc="UE">
   <SDI name="phsA">
      <DAI name="db">
         <Val>2000</Val>
      </DAI>
   </SDI>
   <SDI name="phsB">
      <DAI name="db">
         <Val>2000</Val>
      </DAI>
   </SDI>
   <SDI name="phsC">
```

```
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                  </DOI>
                  <DOI name="A" desc="I">
                     <SDI name="phsA">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsB">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsC">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="neut">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                  </DOI>
               </LN>
               <LN lnClass="MMXU" inst="2" lnType="IED_1/MEAS/MMXU2"
desc="Measure relay1">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="RelId" desc="Relay ID">
                     <DAI name="d">
                        <Val>RelId stands for id of a relay in a protection
constellation</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="PhV" desc="UE">
                     <SDI name="phsA">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsB">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsC">
```

```
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <DAI name="angRef">
                        <Val>Va</Val>
                     </DAI>
                  </DOI>
                  <DOI name="A" desc="I">
                     <SDI name="phsA">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsB">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsC">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <DAI name="angRef">
                        <Val>Va</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="MMXU" inst="3" lnType="IED_1/MEAS/MMXU2"
desc="Measure relay2">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="RelId" desc="Relay ID">
                     <DAI name="d">
                        <Val>RelId stands for id of a relay in a protection
constellation</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="PhV" desc="UE">
                     <SDI name="phsA">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsB">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsC">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <DAI name="angRef">
                        <Val>Va</Val>
                     </DAI>
                  </DOI>
                  <DOI name="A" desc="I">
                     <SDI name="phsA">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsB">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="phsC">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <DAI name="angRef">
                        <Val>Va</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="MSQI" inst="1" lnType="IED_1/MEAS/MSQI1"
desc="Sym comp value">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SeqA" desc="Current">
                     <SDI name="c1">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="c2">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="c3">
                        <DAI name="db">
                           <Val>10000</Val>
                        </DAI>
                     </SDI>
                     <DAI name="seqT">
                        <Val>pos-neg-zero</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SeqV" desc="Voltage">
                     <SDI name="c1">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="c2">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <SDI name="c3">
                        <DAI name="db">
                           <Val>2000</Val>
                        </DAI>
                     </SDI>
                     <DAI name="seqT">
                        <Val>pos-neg-zero</Val>
                     </DAI>
                  </DOI>
               </LN>
            </LDevice>
            <LDevice inst="DR" desc="Disturb Rec">
               <LN0 lnClass="LLN0" inst="" lnType="IED_1/MEAS/LLN0"
desc="General">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>160412115234078</Val>
                     </DAI>
                     <DAI name="ldNs">
                        <Val>IEC 61850-7-4:2003(E)</Val>
                     </DAI>
                  </DOI>
               </LN0>
               <LN lnClass="LPHD" inst="1" lnType="IED_1/PROT/LPHD1"
desc="Device">
                  <DOI name="PhyNam" desc="Info">
                     <DAI name="cdcNs">
                        <Val>7SDx IED Configuration Description File
scdConfigRev same Revision as the Station Configuration Description</Val>
                     </DAI>
                  </DOI>
                  <DOI name="PhyHealth" desc="State"/>
                  <DOI name="Proxy" desc="Proxy"/>
               </LN>
               <LN lnClass="RDRE" inst="1" lnType="IED_1/DR/RDRE1"
desc="Recording">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
```

```
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="RcdMade" desc="Made"/>
                  <DOI name="FltNum" desc="Fault number"/>
                  <DOI name="GriFltNum" desc="Grid fault num"/>
                  <DOI name="RcdStr" desc="Start"/>
               </LN>
            </LDevice>
            <LDevice inst="CTRL" desc="Control">
               <LN0 lnClass="LLN0" inst="" lnType="IED_1/CTRL/LLN0"
desc="General">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>160412115234078</Val>
                     </DAI>
                     <DAI name="ldNs">
                        <Val>IEC 61850-7-4:2003(E)</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="LEDRs" desc="LED acknwldgmnt">
                     <DAI name="ctlModel">
                        <Val>direct-with-normal-security</Val>
                     </DAI>
                  </DOI>
               </LN0>
               <LN lnClass="LPHD" inst="1" lnType="IED_1/CTRL/LPHD1"
desc="Device">
                  <DOI name="PhyNam" desc="Info">
                     <DAI name="cdcNs">
                        <Val>7SDx IED Configuration Description File
scdConfigRev same Revision as the Station Configuration Description</Val>
                     </DAI>
                  </DOI>
                  <DOI name="PhyHealth" desc="State"/>
                  <DOI name="Proxy" desc="Proxy"/>
                  <DOI name="DevStr" desc="Startup">
                     <DAI name="d">
                        <Val>DevStr stands for either InitialStart or
Resume</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="CtlNum" desc="Test oper Nr">
                     <DAI name="d">
                        <Val>CtlNum stands for number of Test
Operations</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CALH" inst="1" lnType="IED_1/CTRL/CALH1"
desc="Fault">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="GrAlm" desc="Group alarms"/>
                  <DOI name="GrWrn" desc="Group Warning"/>
                  <DOI name="ErrBoard1" desc="Error Board 1">
                     <DAI name="d">
                        <Val>ErrBoard1 stands for Error on board 1</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="ErrBoard2" desc="Error Board 2">
                     <DAI name="d">
                        <Val>ErrBoard2 stands for Error on board 2</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="ErrBoard3" desc="Error Board 3">
                     <DAI name="d">
                        <Val>ErrBoard3 stands for Error on board 3</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="ErrBoard4" desc="Error Board 4">
                     <DAI name="d">
                        <Val>ErrBoard4 stands for Error on board 4</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="ErrBoard5" desc="Error Board 5">
                     <DAI name="d">
                        <Val>ErrBoard5 stands for Error on board 5</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="ErrBoard6" desc="Error Board 6">
                     <DAI name="d">
                        <Val>ErrBoard6 stands for Error on board 6</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
                  <DOI name="ErrBoard7" desc="Error Board 7">
                     <DAI name="d">
                        <Val>ErrBoard7 stands for Error on board 7</Val>
                     </DAI>
                     <DAI name="dataNs">
                        <Val>7SDx IED Configuration Description File</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="XCBR" inst="1" prefix="Q0"
lnType="IED_1/CTRL/Q0XCBR1" desc="BreakerXCBR">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="CBOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CSWI" inst="1" prefix="Q0"
lnType="IED_1/CTRL/Q0CSWI1" desc="BreakerCSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>sbo-with-enhanced-security</Val>
                     </DAI>
                     <DAI name="sboTimeout">
                        <Val>300000</Val>
                     </DAI>
                     <DAI name="sboClass">
                        <Val>operate-once</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CILO" inst="1" prefix="Q0"
lnType="IED_1/CTRL/Q0CILO1" desc="BreakerCILO">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="EnaOpn" desc="Enable Open"/>
                  <DOI name="EnaCls" desc="Enable Close"/>
               </LN>
               <LN lnClass="XSWI" inst="1" prefix="Q1"
lnType="IED_1/CTRL/Q1XSWI1" desc="Disc.Swit.XSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwTyp" desc="Switch type">
                     <DAI name="stVal">
                        <Val>Disconnector</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CSWI" inst="1" prefix="Q1"
lnType="IED_1/CTRL/Q0CSWI1" desc="Disc.Swit.CSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>sbo-with-enhanced-security</Val>
                     </DAI>
                     <DAI name="sboTimeout">
                        <Val>300000</Val>
                     </DAI>
                     <DAI name="sboClass">
                        <Val>operate-once</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CILO" inst="1" prefix="Q1"
lnType="IED_1/CTRL/Q0CILO1" desc="Disc.Swit.CILO">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="EnaOpn" desc="Enable Open"/>
                  <DOI name="EnaCls" desc="Enable Close"/>
               </LN>
               <LN lnClass="XSWI" inst="1" prefix="Q2"
lnType="IED_1/CTRL/Q1XSWI1" desc="Q2 Op/ClXSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
```

```
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwTyp" desc="Switch type">
                     <DAI name="stVal">
                        <Val>Disconnector</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CSWI" inst="1" prefix="Q2"
lnType="IED_1/CTRL/Q0CSWI1" desc="Q2 Op/ClCSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>sbo-with-enhanced-security</Val>
                     </DAI>
                     <DAI name="sboTimeout">
                        <Val>300000</Val>
                     </DAI>
                     <DAI name="sboClass">
                        <Val>operate-once</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CILO" inst="1" prefix="Q2"
lnType="IED_1/CTRL/Q0CILO1" desc="Q2 Op/ClCILO">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="EnaOpn" desc="Enable Open"/>
                  <DOI name="EnaCls" desc="Enable Close"/>
               </LN>
               <LN lnClass="XSWI" inst="1" prefix="Q8"
lnType="IED_1/CTRL/Q1XSWI1" desc="EarthSwitXSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwTyp" desc="Switch type">
                     <DAI name="stVal">
                        <Val>Earthing Switch</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CSWI" inst="1" prefix="Q8"
lnType="IED_1/CTRL/Q0CSWI1" desc="EarthSwitCSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>sbo-with-enhanced-security</Val>
                     </DAI>
                     <DAI name="sboTimeout">
                        <Val>300000</Val>
                     </DAI>
                     <DAI name="sboClass">
                        <Val>operate-once</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CILO" inst="1" prefix="Q8"
lnType="IED_1/CTRL/Q0CILO1" desc="EarthSwitCILO">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="EnaOpn" desc="Enable Open"/>
                  <DOI name="EnaCls" desc="Enable Close"/>
               </LN>
               <LN lnClass="XSWI" inst="1" prefix="Q9"
lnType="IED_1/CTRL/Q1XSWI1" desc="Q9 Op/ClXSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="OpCnt" desc="TripComdCounter"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkOpn" desc="Open block">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="BlkCls" desc="Close block">
                     <DAI name="ctlModel">
```

```
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwTyp" desc="Switch type">
                     <DAI name="stVal">
                        <Val>Disconnector</Val>
                     </DAI>
                  </DOI>
                  <DOI name="SwOpCap" desc="Switch capablty">
                     <DAI name="stVal">
                        <Val>None</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CSWI" inst="1" prefix="Q9"
lnType="IED_1/CTRL/Q0CSWI1" desc="Q9 Op/ClCSWI">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Loc" desc="Control Auth"/>
                  <DOI name="Pos" desc="Position">
                     <DAI name="ctlModel">
                        <Val>sbo-with-enhanced-security</Val>
                     </DAI>
                     <DAI name="sboTimeout">
                        <Val>300000</Val>
                     </DAI>
                     <DAI name="sboClass">
                        <Val>operate-once</Val>
                     </DAI>
                  </DOI>
               </LN>
               <LN lnClass="CILO" inst="1" prefix="Q9"
lnType="IED_1/CTRL/Q0CILO1" desc="Q9 Op/ClCILO">
                  <DOI name="Mod" desc="Mode">
                     <DAI name="ctlModel">
                        <Val>status-only</Val>
                     </DAI>
                  </DOI>
                  <DOI name="Beh" desc="Behaviour"/>
                  <DOI name="Health" desc="State"/>
                  <DOI name="NamPlt" desc="Info">
                     <DAI name="configRev">
                        <Val>210404153620012</Val>
                     </DAI>
                  </DOI>
                  <DOI name="EnaOpn" desc="Enable Open"/>
                  <DOI name="EnaCls" desc="Enable Close"/>
               </LN>
            </LDevice>
         </Server>
      </AccessPoint>
   </IED>
   <DataTypeTemplates>
```

```xml
<LNodeType lnClass="LLN0" id="IED_1/PROT/LLN0">
   <DO name="Mod" type="myMod_0"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_6"/>
</LNodeType>
<LNodeType lnClass="LPHD" id="IED_1/PROT/LPHD1">
   <DO name="PhyNam" type="myPhyNam_8"/>
   <DO name="PhyHealth" type="myHealth_5"/>
   <DO name="Proxy" type="SPS_2"/>
</LNodeType>
<LNodeType lnClass="PDIS" id="IED_1/PROT/PDIS1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Str" type="myStr_15"/>
   <DO name="StrAG" type="myStrAG_16"/>
   <DO name="StrBG" type="myStrAG_16"/>
   <DO name="StrCG" type="myStrAG_16"/>
   <DO name="StrAB" type="myStrAG_16"/>
   <DO name="StrBC" type="myStrAG_16"/>
   <DO name="StrCA" type="myStrAG_16"/>
   <DO name="Op" type="myOp_22"/>
</LNodeType>
<LNodeType lnClass="PDIS" id="IED_1/PROT/PDIS2">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Str" type="myStr_15"/>
   <DO name="Op" type="myOp_22"/>
</LNodeType>
<LNodeType lnClass="PTOC" id="IED_1/PROT/PTOC1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Str" type="myStr_15"/>
   <DO name="Op" type="myOp_64"/>
</LNodeType>
<LNodeType lnClass="PTRC" id="IED_1/PROT/PTRC1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Tr" type="myOp_22"/>
   <DO name="Str" type="myStr_88"/>
   <DO name="FinTr" type="myFinTr_89"/>
</LNodeType>
<LNodeType lnClass="PTRC" id="IED_1/PROT/PTRC2">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Op" type="myOp_22"/>
   <DO name="Str" type="myStr_88"/>
</LNodeType>
<LNodeType lnClass="XCBR" id="IED_1/PROT/XCBR2">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Loc" type="SPS_2"/>
   <DO name="OpCnt" type="myOpCnt_101"/>
   <DO name="Pos" type="myPos_102"/>
   <DO name="BlkOpn" type="myBlkOpn_103"/>
   <DO name="BlkCls" type="myBlkOpn_103"/>
   <DO name="SumSwARs" type="mySumSwARs_105"/>
   <DO name="CBOpCap" type="myCBOpCap_107"/>
</LNodeType>
<LNodeType lnClass="PDIF" id="IED_1/PROT/PDIF1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Str" type="myStr_15"/>
   <DO name="Op" type="myOp_64"/>
</LNodeType>
<LNodeType lnClass="PTRC" id="IED_1/PROT/PTRC3">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Op" type="myOp_22"/>
   <DO name="Str" type="myStr_88"/>
   <DO name="StrA" type="myFinTr_89"/>
   <DO name="StrAG" type="myFinTr_89"/>
   <DO name="StrB" type="myFinTr_89"/>
   <DO name="StrBG" type="myFinTr_89"/>
   <DO name="StrC" type="myFinTr_89"/>
   <DO name="StrCG" type="myFinTr_89"/>
   <DO name="StrAB" type="myFinTr_89"/>
   <DO name="StrBC" type="myFinTr_89"/>
   <DO name="StrCA" type="myFinTr_89"/>
   <DO name="StrABC" type="myFinTr_89"/>
   <DO name="StrABG" type="myFinTr_89"/>
   <DO name="StrBCG" type="myFinTr_89"/>
   <DO name="StrCAG" type="myFinTr_89"/>
   <DO name="StrABCG" type="myFinTr_89"/>
</LNodeType>
<LNodeType lnClass="LLN0" id="IED_1/MEAS/LLN0">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_6"/>
</LNodeType>
<LNodeType lnClass="MMTR" id="IED_1/MEAS/MMTR1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="SupWh" type="mySupWh_175"/>
   <DO name="SupVArh" type="mySupWh_175"/>
   <DO name="DmdWh" type="mySupWh_175"/>
   <DO name="DmdVArh" type="mySupWh_175"/>
</LNodeType>
<LNodeType lnClass="MMXU" id="IED_1/MEAS/MMXU1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="TotW" type="myTotW_187"/>
   <DO name="TotVAr" type="myTotW_187"/>
   <DO name="TotVA" type="myTotW_187"/>
   <DO name="TotPF" type="myTotW_187"/>
   <DO name="Hz" type="myTotW_187"/>
   <DO name="PPV" type="myPPV_207"/>
   <DO name="PhV" type="myPhV_226"/>
   <DO name="A" type="myA_245"/>
</LNodeType>
<LNodeType lnClass="MMXU" id="IED_1/MEAS/MMXU2">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="RelId" type="myRelId_274"/>
   <DO name="PhV" type="myPhV_275"/>
   <DO name="A" type="myPhV_275"/>
</LNodeType>
<LNodeType lnClass="MSQI" id="IED_1/MEAS/MSQI1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="SeqA" type="mySeqA_372"/>
   <DO name="SeqV" type="mySeqA_372"/>
</LNodeType>
<LNodeType lnClass="RDRE" id="IED_1/DR/RDRE1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="RcdMade" type="SPS_2"/>
   <DO name="FltNum" type="INS_5"/>
   <DO name="GriFltNum" type="INS_5"/>
   <DO name="RcdStr" type="SPS_2"/>
</LNodeType>
<LNodeType lnClass="LLN0" id="IED_1/CTRL/LLN0">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_6"/>
   <DO name="Loc" type="SPS_2"/>
   <DO name="LEDRs" type="myLEDRs_430"/>
</LNodeType>
<LNodeType lnClass="LPHD" id="IED_1/CTRL/LPHD1">
   <DO name="PhyNam" type="myPhyNam_8"/>
   <DO name="PhyHealth" type="myHealth_5"/>
   <DO name="Proxy" type="SPS_2"/>
   <DO name="DevStr" type="myRelId_274"/>
   <DO name="CtlNum" type="myRelId_274"/>
</LNodeType>
<LNodeType lnClass="CALH" id="IED_1/CTRL/CALH1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="GrAlm" type="SPS_2"/>
   <DO name="GrWrn" type="SPS_2"/>
   <DO name="ErrBoard1" type="myFinTr_89"/>
   <DO name="ErrBoard2" type="myFinTr_89"/>
   <DO name="ErrBoard3" type="myFinTr_89"/>
   <DO name="ErrBoard4" type="myFinTr_89"/>
   <DO name="ErrBoard5" type="myFinTr_89"/>
   <DO name="ErrBoard6" type="myFinTr_89"/>
   <DO name="ErrBoard7" type="myFinTr_89"/>
</LNodeType>
<LNodeType lnClass="XCBR" id="IED_1/CTRL/Q0XCBR1">
```

```
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Loc" type="SPS_2"/>
   <DO name="OpCnt" type="INS_5"/>
   <DO name="Pos" type="myPos_102"/>
   <DO name="BlkOpn" type="myBlkOpn_103"/>
   <DO name="BlkCls" type="myBlkOpn_103"/>
   <DO name="CBOpCap" type="myCBOpCap_107"/>
</LNodeType>
<LNodeType lnClass="CSWI" id="IED_1/CTRL/Q0CSWI1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Loc" type="SPS_2"/>
   <DO name="Pos" type="myPos_467"/>
</LNodeType>
<LNodeType lnClass="CILO" id="IED_1/CTRL/Q0CILO1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="EnaOpn" type="SPS_2"/>
   <DO name="EnaCls" type="SPS_2"/>
</LNodeType>
<LNodeType lnClass="XSWI" id="IED_1/CTRL/Q1XSWI1">
   <DO name="Mod" type="myMod_11"/>
   <DO name="Beh" type="ENS_0"/>
   <DO name="Health" type="myHealth_5"/>
   <DO name="NamPlt" type="myNamPlt_14"/>
   <DO name="Loc" type="SPS_2"/>
   <DO name="OpCnt" type="INS_5"/>
   <DO name="Pos" type="myPos_102"/>
   <DO name="BlkOpn" type="myBlkOpn_103"/>
   <DO name="BlkCls" type="myBlkOpn_103"/>
   <DO name="SwTyp" type="mySwTyp_490"/>
   <DO name="SwOpCap" type="mySwOpCap_491"/>
</LNodeType>
<DOType cdc="LPL" id="myNamPlt_14">
   <DA fc="DC" name="vendor" bType="VisString255"/>
   <DA fc="DC" name="swRev" bType="VisString255"/>
   <DA fc="DC" name="d" bType="VisString255"/>
   <DA fc="DC" name="configRev" bType="VisString255"/>
</DOType>
<DOType cdc="ENS" id="ENS_0" desc="Enumerated status">
   <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Beh"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="SPS" id="SPS_2" desc="Single point status">
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="INS" id="INS_5" desc="Integer status">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="INC" id="myMod_0">
   <DA fc="ST" name="origin" bType="Struct" type="myorigin_1"/>
```

```
         <DA fc="ST" name="ctlNum" bType="INT8U"/>
         <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Beh"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
         <DA fc="EX" name="cdcNs" bType="VisString255"/>
         <DA fc="CO" name="Oper" bType="Struct" type="myOper_2"/>
      </DOType>
      <DOType cdc="INS" id="myHealth_5">
         <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Health"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="LPL" id="myNamPlt_6">
         <DA fc="DC" name="vendor" bType="VisString255"/>
         <DA fc="DC" name="swRev" bType="VisString255"/>
         <DA fc="DC" name="d" bType="VisString255"/>
         <DA fc="DC" name="configRev" bType="VisString255"/>
         <DA fc="EX" name="ldNs" bType="VisString255"/>
      </DOType>
      <DOType cdc="DPL" id="myPhyNam_8">
         <DA fc="DC" name="vendor" bType="VisString255"/>
         <DA fc="DC" name="hwRev" bType="VisString255"/>
         <DA fc="DC" name="swRev" bType="VisString255"/>
         <DA fc="DC" name="serNum" bType="VisString255"/>
         <DA fc="DC" name="model" bType="VisString255"/>
         <DA fc="DC" name="location" bType="VisString255"/>
         <DA fc="EX" name="cdcNs" bType="VisString255"/>
         <DA fc="DC" name="scdConfigRev" bType="VisString255"/>
      </DOType>
      <DOType cdc="INC" id="myMod_11">
         <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="Beh"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
      </DOType>
      <DOType cdc="ACD" id="myStr_15">
         <DA dchg="true" fc="ST" name="general" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="dirGeneral" bType="Enum"
type="dir"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="SPS" id="myStrAG_16">
         <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="EX" name="dataNs" bType="VisString255"/>
      </DOType>
      <DOType cdc="ACT" id="myOp_22">
         <DA dchg="true" fc="ST" name="general" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="phsA" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="phsB" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="phsC" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="ACT" id="myOp_64">
         <DA dchg="true" fc="ST" name="general" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="ACD" id="myStr_88">
```

```
         <DA dchg="true" fc="ST" name="general" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="dirGeneral" bType="Enum"
type="dir"/>
         <DA dchg="true" fc="ST" name="phsA" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="dirPhsA" bType="Enum"
type="dirPhs"/>
         <DA dchg="true" fc="ST" name="phsB" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="dirPhsB" bType="Enum"
type="dirPhs"/>
         <DA dchg="true" fc="ST" name="phsC" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="dirPhsC" bType="Enum"
type="dirPhs"/>
         <DA dchg="true" fc="ST" name="neut" bType="BOOLEAN"/>
         <DA dchg="true" fc="ST" name="dirNeut" bType="Enum"
type="dirPhs"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="SPS" id="myFinTr_89">
         <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="DC" name="d" bType="VisString255"/>
         <DA fc="EX" name="dataNs" bType="VisString255"/>
      </DOType>
      <DOType cdc="INS" id="myOpCnt_101">
         <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="EX" name="dataNs" bType="VisString255"/>
      </DOType>
      <DOType cdc="DPC" id="myPos_102">
         <DA dchg="true" fc="ST" name="stVal" bType="Dbpos" type="Dbpos"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
      </DOType>
      <DOType cdc="SPC" id="myBlkOpn_103">
         <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
      </DOType>
      <DOType cdc="BCR" id="mySumSwARs_105">
         <DA dchg="true" fc="ST" name="actVal" bType="INT32"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="units" bType="Struct" type="myunits_106"/>
         <DA fc="CF" name="pulsQty" bType="FLOAT32"/>
         <DA fc="DC" name="d" bType="VisString255"/>
         <DA fc="EX" name="dataNs" bType="VisString255"/>
      </DOType>
      <DOType cdc="INS" id="myCBOpCap_107">
         <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="CBOpCap"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
      </DOType>
      <DOType cdc="BCR" id="mySupWh_175">
         <DA dchg="true" fc="ST" name="actVal" bType="INT32"/>
         <DA qchg="true" fc="ST" name="q" bType="Quality"/>
         <DA fc="ST" name="t" bType="Timestamp"/>
         <DA fc="CF" name="units" bType="Struct" type="myunits_106"/>
         <DA fc="CF" name="pulsQty" bType="FLOAT32"/>
```

```
</DOType>
<DOType cdc="MV" id="myTotW_187">
   <DA fc="MX" name="instMag" bType="Struct" type="myinstMag_188"/>
   <DA fc="MX" name="mag" bType="Struct" type="myinstMag_188"/>
   <DA qchg="true" fc="MX" name="q" bType="Quality"/>
   <DA fc="MX" name="t" bType="Timestamp"/>
   <DA fc="CF" name="units" bType="Struct" type="myunits_106"/>
   <DA fc="CF" name="db" bType="INT32U"/>
</DOType>
<DOType cdc="DEL" id="myPPV_207">
   <SDO type="myphsAB_208" name="phsAB"/>
   <SDO type="myphsAB_208" name="phsBC"/>
   <SDO type="myphsAB_208" name="phsCA"/>
</DOType>
<DOType cdc="CMV" id="myphsAB_208">
   <DA fc="MX" name="instCVal" bType="Struct" type="myinstCVal_209"/>
   <DA fc="MX" name="cVal" bType="Struct" type="myinstCVal_209"/>
   <DA fc="CF" name="units" bType="Struct" type="myunits_106"/>
   <DA qchg="true" fc="MX" name="q" bType="Quality"/>
   <DA fc="MX" name="t" bType="Timestamp"/>
   <DA fc="CF" name="db" bType="INT32U"/>
</DOType>
<DOType cdc="WYE" id="myPhV_226">
   <SDO type="myphsAB_208" name="phsA"/>
   <SDO type="myphsAB_208" name="phsB"/>
   <SDO type="myphsAB_208" name="phsC"/>
</DOType>
<DOType cdc="WYE" id="myA_245">
   <SDO type="myphsAB_208" name="phsA"/>
   <SDO type="myphsAB_208" name="phsB"/>
   <SDO type="myphsAB_208" name="phsC"/>
   <SDO type="myphsAB_208" name="neut"/>
</DOType>
<DOType cdc="INS" id="myRelId_274">
   <DA dchg="true" fc="ST" name="stVal" bType="INT32"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
   <DA fc="DC" name="d" bType="VisString255"/>
   <DA fc="EX" name="dataNs" bType="VisString255"/>
</DOType>
<DOType cdc="WYE" id="myPhV_275">
   <SDO type="myphsA_276" name="phsA"/>
   <SDO type="myphsA_276" name="phsB"/>
   <SDO type="myphsA_276" name="phsC"/>
   <DA fc="CF" name="angRef" bType="Enum" type="angid"/>
</DOType>
<DOType cdc="CMV" id="myphsA_276">
   <DA fc="MX" name="instCVal" bType="Struct" type="myinstCVal_209"/>
   <DA fc="MX" name="cVal" bType="Struct" type="mycVal_279"/>
   <DA fc="CF" name="units" bType="Struct" type="myunits_106"/>
   <DA qchg="true" fc="MX" name="q" bType="Quality"/>
   <DA fc="MX" name="t" bType="Timestamp"/>
   <DA fc="CF" name="db" bType="INT32U"/>
</DOType>
<DOType cdc="SEQ" id="mySeqA_372">
   <SDO type="myphsAB_208" name="c1"/>
   <SDO type="myphsAB_208" name="c2"/>
   <SDO type="myphsAB_208" name="c3"/>
   <DA fc="MX" name="seqT" bType="Enum" type="seqT"/>
</DOType>
<DOType cdc="SPC" id="myLEDRs_430">
   <DA fc="ST" name="origin" bType="Struct" type="myorigin_1"/>
   <DA fc="ST" name="ctlNum" bType="INT8U"/>
   <DA dchg="true" fc="ST" name="stVal" bType="BOOLEAN"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
   <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
   <DA fc="EX" name="cdcNs" bType="VisString255"/>
   <DA fc="CO" name="Oper" bType="Struct" type="SBOwDbpos_0"/>
</DOType>
<DOType cdc="DPC" id="myPos_467">
   <DA fc="ST" name="origin" bType="Struct" type="myorigin_1"/>
   <DA fc="ST" name="ctlNum" bType="INT8U"/>
   <DA dchg="true" fc="ST" name="stVal" bType="Dbpos" type="Dbpos"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
   <DA dchg="true" fc="ST" name="stSeld" bType="BOOLEAN"/>
   <DA fc="CF" name="ctlModel" bType="Enum" type="ctlModel"/>
   <DA fc="CF" name="sboTimeout" bType="INT32U"/>
   <DA fc="CF" name="sboClass" bType="Enum" type="sboClass"/>
   <DA fc="EX" name="cdcNs" bType="VisString255"/>
   <DA fc="CO" name="SBOw" bType="Struct" type="SBOwDbpos_0"/>
   <DA fc="CO" name="Oper" bType="Struct" type="SBOwDbpos_0"/>
   <DA fc="CO" name="Cancel" bType="Struct" type="myCancel_473"/>
</DOType>
<DOType cdc="INS" id="mySwTyp_490">
   <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="SwTyp"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DOType cdc="INS" id="mySwOpCap_491">
   <DA dchg="true" fc="ST" name="stVal" bType="Enum" type="ShOpCap"/>
   <DA qchg="true" fc="ST" name="q" bType="Quality"/>
   <DA fc="ST" name="t" bType="Timestamp"/>
</DOType>
<DAType id="myorigin_1">
   <BDA name="orCat" bType="Enum" type="orCategory"/>
   <BDA name="orIdent" bType="Octet64"/>
</DAType>
<DAType id="SBOwDbpos_0">
   <BDA name="ctlVal" bType="BOOLEAN"/>
   <BDA name="origin" bType="Struct" type="myorigin_1"/>
   <BDA name="ctlNum" bType="INT8U"/>
   <BDA name="T" bType="Timestamp"/>
   <BDA name="Test" bType="BOOLEAN"/>
   <BDA name="Check" bType="Check"/>
</DAType>
<DAType id="myOper_2">
   <BDA name="ctlVal" bType="Enum" type="Beh"/>
   <BDA name="origin" bType="Struct" type="myorigin_1"/>
   <BDA name="ctlNum" bType="INT8U"/>
   <BDA name="T" bType="Timestamp"/>
   <BDA name="Test" bType="BOOLEAN"/>
   <BDA name="Check" bType="Check" type="Check"/>
</DAType>
<DAType id="myunits_106">
   <BDA name="SIUnit" bType="Enum" type="SIUnit"/>
   <BDA name="multiplier" bType="Enum" type="multiplier"/>
</DAType>
<DAType id="myinstMag_188">
   <BDA name="f" bType="FLOAT32"/>
</DAType>
<DAType id="myinstCVal_209">
   <BDA name="mag" bType="Struct" type="myinstMag_188"/>
</DAType>
<DAType id="mycVal_279">
```

```
   <BDA name="mag" bType="Struct" type="myinstMag_188"/>
   <BDA name="ang" bType="Struct" type="myinstMag_188"/>
</DAType>
<DAType id="myCancel_473">
   <BDA name="ctlVal" bType="BOOLEAN"/>
   <BDA name="origin" bType="Struct" type="myorigin_1"/>
   <BDA name="ctlNum" bType="INT8U"/>
   <BDA name="T" bType="Timestamp"/>
   <BDA name="Test" bType="BOOLEAN"/>
</DAType>
<EnumType id="orCategory">
   <EnumVal ord="0">not-supported</EnumVal>
   <EnumVal ord="1">bay-control</EnumVal>
   <EnumVal ord="2">station-control</EnumVal>
   <EnumVal ord="3">remote-control</EnumVal>
   <EnumVal ord="4">automatic-bay</EnumVal>
   <EnumVal ord="5">automatic-station</EnumVal>
   <EnumVal ord="6">automatic-remote</EnumVal>
   <EnumVal ord="7">maintenance</EnumVal>
   <EnumVal ord="8">process</EnumVal>
</EnumType>
<EnumType id="ctlModel">
   <EnumVal ord="0">status-only</EnumVal>
   <EnumVal ord="1">direct-with-normal-security</EnumVal>
   <EnumVal ord="2">sbo-with-normal-security</EnumVal>
   <EnumVal ord="3">direct-with-enhanced-security</EnumVal>
   <EnumVal ord="4">sbo-with-enhanced-security</EnumVal>
</EnumType>
<EnumType id="Beh">
   <EnumVal ord="1">on</EnumVal>
   <EnumVal ord="2">blocked</EnumVal>
   <EnumVal ord="3">test</EnumVal>
   <EnumVal ord="4">test/blocked</EnumVal>
   <EnumVal ord="5">off</EnumVal>
</EnumType>
<EnumType id="Health">
   <EnumVal ord="1">Ok</EnumVal>
   <EnumVal ord="2">Warning</EnumVal>
   <EnumVal ord="3">Alarm</EnumVal>
</EnumType>
<EnumType id="PulseConfigCmdQual">
   <EnumVal ord="0">pulse</EnumVal>
   <EnumVal ord="1">persistent</EnumVal>
</EnumType>
<EnumType id="SIUnit">
   <EnumVal ord="1">none</EnumVal>
   <EnumVal ord="2">m</EnumVal>
   <EnumVal ord="3">kg</EnumVal>
   <EnumVal ord="4">s</EnumVal>
   <EnumVal ord="5">A</EnumVal>
   <EnumVal ord="6">K</EnumVal>
   <EnumVal ord="7">mol</EnumVal>
   <EnumVal ord="8">cd</EnumVal>
   <EnumVal ord="9">deg</EnumVal>
   <EnumVal ord="10">rad</EnumVal>
   <EnumVal ord="11">sr</EnumVal>
   <EnumVal ord="21">Gy</EnumVal>
   <EnumVal ord="22">q</EnumVal>
   <EnumVal ord="23">°C</EnumVal>
   <EnumVal ord="24">Sv</EnumVal>
   <EnumVal ord="25">F</EnumVal>
   <EnumVal ord="26">C</EnumVal>
   <EnumVal ord="27">S</EnumVal>
```

```
      <EnumVal ord="28">H</EnumVal>
      <EnumVal ord="29">V</EnumVal>
      <EnumVal ord="30">ohm</EnumVal>
      <EnumVal ord="31">J</EnumVal>
      <EnumVal ord="32">N</EnumVal>
      <EnumVal ord="33">Hz</EnumVal>
      <EnumVal ord="34">lx</EnumVal>
      <EnumVal ord="35">Lm</EnumVal>
      <EnumVal ord="36">Wb</EnumVal>
      <EnumVal ord="37">T</EnumVal>
      <EnumVal ord="38">W</EnumVal>
      <EnumVal ord="39">Pa</EnumVal>
      <EnumVal ord="41">m²</EnumVal>
      <EnumVal ord="42">m³</EnumVal>
      <EnumVal ord="43">m/s</EnumVal>
      <EnumVal ord="44">m/s²</EnumVal>
      <EnumVal ord="45">m³/s</EnumVal>
      <EnumVal ord="46">m/m³</EnumVal>
      <EnumVal ord="47">M</EnumVal>
      <EnumVal ord="48">kg/m³</EnumVal>
      <EnumVal ord="49">m²/s</EnumVal>
      <EnumVal ord="50">W/m K</EnumVal>
      <EnumVal ord="51">J/K</EnumVal>
      <EnumVal ord="52">ppm</EnumVal>
      <EnumVal ord="53">1/s</EnumVal>
      <EnumVal ord="54">rad/s</EnumVal>
      <EnumVal ord="61">VA</EnumVal>
      <EnumVal ord="62">Watts</EnumVal>
      <EnumVal ord="63">VAr</EnumVal>
      <EnumVal ord="64">phi</EnumVal>
      <EnumVal ord="65">cos_phi</EnumVal>
      <EnumVal ord="66">Vs</EnumVal>
      <EnumVal ord="67">V²</EnumVal>
      <EnumVal ord="68">As</EnumVal>
      <EnumVal ord="69">A²</EnumVal>
      <EnumVal ord="70">A²t</EnumVal>
      <EnumVal ord="71">VAh</EnumVal>
      <EnumVal ord="72">Wh</EnumVal>
      <EnumVal ord="73">VArh</EnumVal>
      <EnumVal ord="74">V/Hz</EnumVal>
   </EnumType>
   <EnumType id="multiplier">
      <EnumVal ord="-24">y</EnumVal>
      <EnumVal ord="-21">z</EnumVal>
      <EnumVal ord="-18">a</EnumVal>
      <EnumVal ord="-15">f</EnumVal>
      <EnumVal ord="-12">p</EnumVal>
      <EnumVal ord="-9">n</EnumVal>
      <EnumVal ord="-6">µ</EnumVal>
      <EnumVal ord="-3">m</EnumVal>
      <EnumVal ord="-2">c</EnumVal>
      <EnumVal ord="-1">d</EnumVal>
      <EnumVal ord="0"></EnumVal>
      <EnumVal ord="1">da</EnumVal>
      <EnumVal ord="2">h</EnumVal>
      <EnumVal ord="3">k</EnumVal>
      <EnumVal ord="6">M</EnumVal>
      <EnumVal ord="9">G</EnumVal>
      <EnumVal ord="12">T</EnumVal>
      <EnumVal ord="15">P</EnumVal>
      <EnumVal ord="18">E</EnumVal>
      <EnumVal ord="21">Z</EnumVal>
      <EnumVal ord="24">Y</EnumVal>
```

```
</EnumType>
<EnumType id="Boolean">
   <EnumVal ord="0">FALSE</EnumVal>
   <EnumVal ord="1">TRUE</EnumVal>
</EnumType>
<EnumType id="sboClass">
   <EnumVal ord="0">operate-once</EnumVal>
   <EnumVal ord="1">operate-many</EnumVal>
</EnumType>
<EnumType id="dir">
   <EnumVal ord="0">unknown</EnumVal>
   <EnumVal ord="1">forward</EnumVal>
   <EnumVal ord="2">backward</EnumVal>
   <EnumVal ord="3">both</EnumVal>
</EnumType>
<EnumType id="dirPhs">
   <EnumVal ord="0">unknown</EnumVal>
   <EnumVal ord="1">forward</EnumVal>
   <EnumVal ord="2">backward</EnumVal>
</EnumType>
<EnumType id="Check">
   <EnumVal ord="0">no-check</EnumVal>
   <EnumVal ord="1">synchrocheck</EnumVal>
   <EnumVal ord="2">interlocking-check</EnumVal>
   <EnumVal ord="3">both</EnumVal>
</EnumType>
<EnumType id="Dbpos">
   <EnumVal ord="0">intermediate</EnumVal>
   <EnumVal ord="1">off</EnumVal>
   <EnumVal ord="2">on</EnumVal>
   <EnumVal ord="3">bad</EnumVal>
</EnumType>
<EnumType id="seqT">
   <EnumVal ord="0">pos-neg-zero</EnumVal>
   <EnumVal ord="1">dir-quad-zero</EnumVal>
</EnumType>
<EnumType id="Tcmd">
   <EnumVal ord="0">stop</EnumVal>
   <EnumVal ord="1">lower</EnumVal>
   <EnumVal ord="2">higher</EnumVal>
   <EnumVal ord="3">reserved</EnumVal>
</EnumType>
<EnumType id="sev">
   <EnumVal ord="0">unknown</EnumVal>
   <EnumVal ord="1">critical</EnumVal>
   <EnumVal ord="2">major</EnumVal>
   <EnumVal ord="3">minor</EnumVal>
   <EnumVal ord="4">warning</EnumVal>
</EnumType>
<EnumType id="range">
   <EnumVal ord="0">normal</EnumVal>
   <EnumVal ord="1">high</EnumVal>
   <EnumVal ord="2">low</EnumVal>
   <EnumVal ord="3">high-high</EnumVal>
   <EnumVal ord="4">low-low</EnumVal>
</EnumType>
<EnumType id="angidCMV">
   <EnumVal ord="0">V</EnumVal>
   <EnumVal ord="1">A</EnumVal>
   <EnumVal ord="2">other</EnumVal>
</EnumType>
<EnumType id="angid">
   <EnumVal ord="0">Va</EnumVal>
   <EnumVal ord="1">Vb</EnumVal>
   <EnumVal ord="2">Vc</EnumVal>
   <EnumVal ord="3">Aa</EnumVal>
   <EnumVal ord="4">Ab</EnumVal>
   <EnumVal ord="5">Ac</EnumVal>
   <EnumVal ord="6">Vab</EnumVal>
   <EnumVal ord="7">Vbc</EnumVal>
   <EnumVal ord="8">Vca</EnumVal>
   <EnumVal ord="9">Vother</EnumVal>
   <EnumVal ord="10">Aother</EnumVal>
</EnumType>
<EnumType id="phsid">
   <EnumVal ord="0">A</EnumVal>
   <EnumVal ord="1">B</EnumVal>
   <EnumVal ord="2">C</EnumVal>
</EnumType>
<EnumType id="hvid">
   <EnumVal ord="0">fundamental</EnumVal>
   <EnumVal ord="1">rms</EnumVal>
   <EnumVal ord="2">absolute</EnumVal>
</EnumType>
<EnumType id="SetCharact">
   <EnumVal ord="1"></EnumVal>
   <EnumVal ord="2"></EnumVal>
   <EnumVal ord="3"></EnumVal>
   <EnumVal ord="4"></EnumVal>
   <EnumVal ord="5"></EnumVal>
   <EnumVal ord="6"></EnumVal>
   <EnumVal ord="7"></EnumVal>
   <EnumVal ord="8"></EnumVal>
   <EnumVal ord="9"></EnumVal>
   <EnumVal ord="10"></EnumVal>
   <EnumVal ord="11"></EnumVal>
   <EnumVal ord="12"></EnumVal>
   <EnumVal ord="13"></EnumVal>
   <EnumVal ord="14"></EnumVal>
   <EnumVal ord="15"></EnumVal>
   <EnumVal ord="16"></EnumVal>
   <EnumVal ord="17"></EnumVal>
   <EnumVal ord="18"></EnumVal>
   <EnumVal ord="19"></EnumVal>
   <EnumVal ord="20"></EnumVal>
   <EnumVal ord="21"></EnumVal>
   <EnumVal ord="22"></EnumVal>
   <EnumVal ord="23"></EnumVal>
   <EnumVal ord="24"></EnumVal>
   <EnumVal ord="25"></EnumVal>
   <EnumVal ord="26"></EnumVal>
   <EnumVal ord="27"></EnumVal>
   <EnumVal ord="28"></EnumVal>
   <EnumVal ord="29"></EnumVal>
   <EnumVal ord="30"></EnumVal>
   <EnumVal ord="31"></EnumVal>
   <EnumVal ord="32"></EnumVal>
   <EnumVal ord="33"></EnumVal>
   <EnumVal ord="34"></EnumVal>
   <EnumVal ord="35"></EnumVal>
   <EnumVal ord="36"></EnumVal>
   <EnumVal ord="37"></EnumVal>
   <EnumVal ord="38"></EnumVal>
   <EnumVal ord="39"></EnumVal>
   <EnumVal ord="40"></EnumVal>
   <EnumVal ord="41"></EnumVal>
   <EnumVal ord="42"></EnumVal>
   <EnumVal ord="43"></EnumVal>
   <EnumVal ord="44"></EnumVal>
   <EnumVal ord="45"></EnumVal>
   <EnumVal ord="46"></EnumVal>
   <EnumVal ord="47"></EnumVal>
   <EnumVal ord="48"></EnumVal>
</EnumType>
<EnumType id="AutoRecSt">
   <EnumVal ord="1">Ready</EnumVal>
   <EnumVal ord="2">InProgress</EnumVal>
   <EnumVal ord="3">Successful</EnumVal>
</EnumType>
<EnumType id="CBOpCap">
   <EnumVal ord="1">None</EnumVal>
   <EnumVal ord="2">Open</EnumVal>
   <EnumVal ord="3">Close-Open</EnumVal>
   <EnumVal ord="4">Open-Close-Open</EnumVal>
   <EnumVal ord="5">Close-Open-Close-Open</EnumVal>
</EnumType>
<EnumType id="DirMod">
   <EnumVal ord="1">NonDirectional</EnumVal>
   <EnumVal ord="2">Forward</EnumVal>
   <EnumVal ord="3">Inverse</EnumVal>
</EnumType>
<EnumType id="FanCtl">
   <EnumVal ord="1">Inactive</EnumVal>
   <EnumVal ord="2">Stage1</EnumVal>
   <EnumVal ord="3">Stage2</EnumVal>
   <EnumVal ord="4">Stage3</EnumVal>
</EnumType>
<EnumType id="FltLoop">
   <EnumVal ord="1">PhaseAtoGround</EnumVal>
   <EnumVal ord="2">PhaseBtoGround</EnumVal>
   <EnumVal ord="3">PhaseCtoGround</EnumVal>
   <EnumVal ord="4">PhaseAtoB</EnumVal>
   <EnumVal ord="5">PhaseBtoC</EnumVal>
   <EnumVal ord="6">PhaseCtoA</EnumVal>
   <EnumVal ord="7">Others</EnumVal>
</EnumType>
<EnumType id="GnSt">
   <EnumVal ord="1">Stopped</EnumVal>
   <EnumVal ord="2">Stopping</EnumVal>
   <EnumVal ord="3">Started</EnumVal>
   <EnumVal ord="4">Starting</EnumVal>
   <EnumVal ord="5">Disabled</EnumVal>
</EnumType>
<EnumType id="POWCap">
   <EnumVal ord="1">None</EnumVal>
   <EnumVal ord="2">Close</EnumVal>
   <EnumVal ord="3">Open</EnumVal>
   <EnumVal ord="4">Close and Open</EnumVal>
</EnumType>
<EnumType id="ShOpCap">
   <EnumVal ord="1">None</EnumVal>
   <EnumVal ord="2">Open</EnumVal>
   <EnumVal ord="3">Close</EnumVal>
   <EnumVal ord="4">Open and Close</EnumVal>
</EnumType>
<EnumType id="SwTyp">
   <EnumVal ord="1">Load Break</EnumVal>
   <EnumVal ord="2">Disconnector</EnumVal>
   <EnumVal ord="3">Earthing Switch</EnumVal>
```

```
         <EnumVal ord="4">High Speed Earthing Switch</EnumVal>
      </EnumType>
   </DataTypeTemplates>
</SCL>
```

APPENDIX E

APPENDIX E: DK60_XCBR VHDL CODE

```
-- ------------------------------------------------------------------------
--------
-- Project: Actuator Node/ DK60-CPLD
-- File:    Actuator Node/ DK60_V101.VHD
-- Version: 1.01
-- Date:      15.March.2012
--
-- The CPLD supports the flexibly to adapt the DK60 evaluation board to the
desired hardware base.
-- Unused CPLD pins and logic functions can be used freely.
-- Hardware logic functions outside of the SC143 are implemented on the
DK60 board with a CPLD:
--  external SRAM chip select logic (FLSSEL, UCSIN#, UCSOUT#)
--  Compact Flash interface
--  8-Bit I/O port
--
-- Project Requirements:
-- Family:   XC9500XL CPLDs
-- Device:   XC95144XL
-- Package:        TQ144
-- Speed Grade: -5
-- Process Properties: Fitting: I/O Termination: Float
--
-- Fitter requirements
-- Set in the "Fit-Properties" the "I/O Pin Termination" entry to "Float"
-- Set in the "Output Slew Rate" entry to "Fast"
--
-- History:      v1.00   - initial implementation S. Perzborn
-- Addaptation: Prof. P. Petev and John Retonda (CPUT- 2011)
--               v1.01   - prevent a bus crash, if PIO2(PCS6) and UCSOUT is
seleced
--               v1.02   - corrected CS_SRAM generation
--
-- ------------------------------------------------------------------------
--------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity DK60 is
    Port ( IO_IN : in std_logic_vector(7 downto 0);    -- 8 Bit in Port
             IO_OUT : out std_logic_vector(7 downto 0);  -- 8 Bit out Port
             unused_CPLD_LED : out std_logic_vector(7 downto 0); --8 bits
LED  routed to the expension header
             PCS6 : in std_logic;                          -- Used I/O-
Chipselect
             A : in std_logic_vector(8 downto 0);          -- SC143 Adress
Bus
             D : inout std_logic_vector(15 downto 0);    -- SC143 Data Bus
             RD : in std_logic;                                   -- SC143
RD# signal
             WR : in std_logic;                                  -- SC143
WR# signal
             RESET : in std_logic;                          -- external
reset signal (Power On Reset, Reset Button)
             FLSSEL : in std_logic;                         -- SC143 FLSSEL
signal (int. Flash / ext. Memory)
```

```
             UCSIN : out std_logic;       -- generated signal for the SC143
UCSIN
             UCSOUT : in std_logic;       -- SC143 UCSOUT signal
             CS_SRAM : out std_logic;     -- generated chip-select signal
for external SRAM
             CSX10 : out std_logic;       -- generated chip-select signal
for external I/O (C10h..C1Fh)
             CSX40 : out std_logic;       -- generated chip-select signal
for external I/O (C40h..C4Fh)
             CSX50 : out std_logic;       -- generated chip-select signal
for external I/O (C50h..C5Fh)
             CSX60 : out std_logic;       -- generated chip-select signal
for external I/O (C60h..C6Fh)
             CSX70 : out std_logic;       -- generated chip-select signal
for external I/O (C70h..C7Fh)
             IDE_RES : out std_logic;     -- generated Reset Signal for
Compact Flash (low-active)
             IDE_D : inout std_logic_vector(15 downto 0); -- generated data
bus for Compact Flash
             IDE_A : out std_logic_vector(2 downto 0);       -- generated
data bus for Compact Flash
             IDE_CS0 : out std_logic;     -- generated Chip-Select signal
for Compact Flash (low active) C20h..C2Fh
             IDE_CS1 : out std_logic;     -- generated Chip-Select signal
for Compact Flash (low active) C30h..C3Fh
             IDE_IOWR : out std_logic;    -- generated IOWR# signal for
Compact Flash (low active)
             IDE_IORD : out std_logic     -- generated IORD# signal for
Compact Flash (low active)
            );
end DK60;

architecture Behavioral of DK60 is
      signal out_latch : std_logic_vector (7 downto 0); -- LED port latch
      signal IDE_CS : std_logic;                           -- access to
C20h..C3Fh
      signal IO_CS : std_logic_vector (7 downto 0);   -- I/O chip selects
0xC00, 0xC10,0xC20 ...
begin

-- Chip Selects
process (PCS6,A,RESET,IO_CS)
      variable cmp_tmp : std_logic_vector (6 downto 0);
      variable tmp_IOCS : std_logic_vector (7 downto 0);
      variable tmp_IDE_CS : std_logic;
begin
      tmp_IDE_CS := '0';
      tmp_IOCS := "11111111";
      cmp_tmp := A(8 downto 4) & PCS6 & RESET;
      case cmp_tmp is
        when "0000001" => tmp_IOCS(0) := '0'; -- 0xC00 & PCS6 & NOT RESET
        when "0000101" => tmp_IOCS(1) := '0'; -- 0xC10 & PCS6 & NOT RESET
        when "0001001" => tmp_IOCS(2) := '0'; -- CF 0xC20 & PCS6 & NOT
RESET
                              tmp_IDE_CS := '1';
        when "0001101" => tmp_IOCS(3) := '0'; -- CF 0xC30 & PCS6 & NOT
RESET
                              tmp_IDE_CS := '1';
        when "0010001" => tmp_IOCS(4) := '0'; -- 0xC40 & PCS6 & NOT RESET
        when "0010101" => tmp_IOCS(5) := '0'; -- 0xC50 & PCS6 & NOT RESET
        when "0011001" => tmp_IOCS(6) := '0'; -- 0xC60 & PCS6 & NOT RESET
        when "0011101" => tmp_IOCS(7) := '0'; -- 0xC70 & PCS6 & NOT RESET
        when others => NULL;
      end case;

      -- CF adress
      if tmp_IDE_CS = '1' then
            IDE_A (2 downto 0) <= A(2 downto 0);
      else
            IDE_A (2 downto 0) <= "000";
      end if;
      IO_CS <= tmp_IOCS;
      IDE_CS <= tmp_IDE_CS;
end process; -- Chip Selects

-- write logic
process (A,PCS6,RESET,WR,IDE_CS,D,IO_CS)
      variable tmp_cmp : std_logic_vector (9 downto 0);
      variable tmp_IDE_D : std_logic_vector (15 downto 0);
      variable tmp_IOWR : std_logic;
begin
  tmp_IDE_D := "ZZZZZZZZZZZZZZZZ";
  tmp_cmp := A(8 downto 0) & PCS6;
  tmp_IOWR :='1';

  -- LED port output
  if rising_edge(WR) then
      if IO_CS(0) = '0' then
            if A(3 downto 0) = "0000" then
                  out_latch(7 downto 0) <= D (7 downto 0);
            end if;
      end if;
  end if;

  -- CF output
  if WR = '0' then
      if IDE_CS = '1' then
            tmp_IOWR := '0';
            if A(0) = '0' then
                  tmp_IDE_D := D;
            else
                  tmp_IDE_D(7 downto 0) := D(15 downto 8);
            end if;
      end if;
  end if;
  IDE_IOWR <= tmp_IOWR;
  if RESET = '0' then
    out_latch <= "00000000";
  end if;
  IDE_D <= tmp_IDE_D;
end process;


-- Flashselect logic
-- NOTE: Without this function, the SC143 on the DK60 is not able to Boot,
--       because the internal SC143 Flash-Memory needs a low signal on
UCSIN#
--       when the SC143 performs an internal flash access.
process (FLSSEL, UCSOUT, RESET, PCS6)
begin
      if PCS6 = '1' and RESET = '1' then
            if FLSSEL = '1' then
                  UCSIN <= UCSOUT;
                  CS_SRAM <= '1';
            else -- FLSSEL = '0'
                  UCSIN <= '1';
                  CS_SRAM <= UCSOUT;
            end if;
      else
            UCSIN <= '1';
            CS_SRAM <= '1';
      end if;
end process;

-- read logic
process (RD,A,IDE_CS, IO_CS,IO_IN,out_latch,IDE_D,RESET)
  variable tmp_d : std_logic_vector (15 downto 0);
  variable tmp_IORD : std_logic;
begin
  tmp_d := "ZZZZZZZZZZZZZZZZ";
  tmp_IORD := '1';
  -- read I/O port
  if RD = '0' then
    if IO_CS(0) = '0' then
       case A(1 downto 0) is
        when "00" =>
           tmp_d(15 downto 8) := out_latch(7 downto 0); -- read out port
(high byte)
                 tmp_d(7 downto 0) := IO_IN(7 downto 0); -- read DIP SW
(low byte)
            --    tmp_d(7 downto 0) := unused_CPLD_DIP(7 downto 0);
        when "01" =>
          tmp_d(15 downto 8) := out_latch(7 downto 0); -- read out port
        when others =>
          tmp_d(15 downto 0) :=  X"0003"; -- CPLD Code Version
      end case;
    end if;

    -- read IDE
    if IDE_CS = '1' then
      tmp_IORD := '0';
      if A(0) = '0' then
         tmp_d := IDE_D;
       else
         tmp_d(15 downto 8) := IDE_D(7 downto 0);
       end if;
    end if;
  end if;
  D <= tmp_d;
  IDE_IORD <= tmp_IORD;
end process;

  IDE_RES <= RESET;
  IO_OUT <= out_latch;
  unused_CPLD_LED <= out_latch; -- affectation of internal Latch signal to
the
                                      -- Unsed CPLD are at header [S10]
  IDE_CS0 <= IO_CS(2);
  IDE_CS1 <= IO_CS(3);
  CSX10 <= IO_CS(1);
  CSX40 <= IO_CS(4);
  CSX50 <= IO_CS(5);
  CSX60 <= IO_CS(6);
  CSX70 <= IO_CS(7);
end Behavioral;
```

APPENDIX F

APPENDIX F: GA61850 Software Application code

```
/*****************************************************************************/
/*! \file           ga61850.c
 *  \brief      Source code for an IEC 61850 Circuit Breaker Actuator
 *  \author     John-Charly Retonda-Modiya
 *  \version    2.0
 *  \date       29 June 2012
 *  \bug
 *  \warning    None
 *  \to do      None
 *  \par        2012. Cape Peninsula University of Technology
 *                (With the cooperation of Systemcorp and Beck IPC)
 *
 *              Phone   : +27 762 755 393 (RSA)
 *              Email   : retonda7@hotmail.com
 */
/*****************************************************************************/

/******************************************************
 This is the Source code for an IEC 61850 Circuit Breaker Actuator developed
 at CPUT by John-Charly Retonda-Modiya. An emulated breaker is used. The CPLD
 have been reprogrammed to use the I/O interface [S10]on the DK 60 board
 All I/O are routed to extension header S10 (Unused CPLD pins). The Position
 Control of the Circuit breaker is issued via unused CPLD_3 (LED_1/Output_1).
 The Position status of the circuit breaker is read/feedback via unused CPLD_1
 and CPLD_2. Please refer to VHDL code section and the present thesis for more details.
 CB=> Circuit breaker.

PIS10 Version Used:  V1.36 (http://www.systemcorp.com.au/)
To run

GA61850.EXE <FILENAME_Server.CID> <FILENAME_Client.SCD> [NTP Server]

e.g GA61850 DK60SV1.CID  DK60CL1.SCD
******************************************************/
```

```
/******************************************************************************
* Includes
******************************************************************************/
#include <clib.h>
#include <stdio.h>
#include <mem.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <time.h>


/* Include IEC 61850 API */
#include "IEC61850API.h"

/******************************************************************************
* Constants
******************************************************************************/
//#define DEBUG_SV61850 1
/* Maximum Object types */
#define OBJECT_TYPES    2
/* Total Objects in each object type */
#define OBJECTS             3
/* SC143 Input Output Address Location */
#define IO_ADDR             0xC00

/* Input Output Handler Task Priority */
#define IOHANDLER_PRIO  120
/* Input Output Handler Task Stack Size */
#define TASK_STACKSIZE  2048

#define UPDATEMAX       32
#define BECK_TIMEZONE_FUDGE_FACTOR      18000
#define UPDATELIMIT     10

#define STRING_ITEM     1
```

```
/* Object Types */
enum
{
    DIGITAL_INPUT       = 1,        // Digital Input (CB Position status )
    DIGITAL_OUTPUT  = 2,        // Digital Output (LED/ CB control )
    STRING_VALUE        = 3,        // String Read and Write
}eObjectTypes;

enum
{
    DIGINPUT_INDEX  = 0,        // Digital Input
    DIGOUTPUT_INDEX     = 1,        // Digital Output
}eObjectIndex;

/* Object Information Index */
enum
{
    VALUE_INDEX         = 1,        // Value Index
    QUALITY_INDEX       = 2,        // Quality Index
    TIME_STAMP_INDEX    = 3,        // Time Stamp Index
}eObjectInfoIndex;

#define DFLTTIMEZONEHOUR    0
#define DFLTTIMEZONEMIN 0

#ifndef BOOL
    #define BOOL   unsigned char
#endif

/*Double point status added in PIS10 V1.36 */
#ifndef DBPOS
    #define DBPOS   IEC61850_DATATYPE_INT8U
#endif
/* Define fALSE/TRUE  for INPUT switches */
#ifndef FALSE
  #define FALSE  0
  #define TRUE   1
```

```
#endif

/*! \typedef enum eErrorCodes
     *  \brief Error Codes
     */
    typedef enum
    {
        SNTP_TIMEZONE_INVALID                           =   -25000,
        SNTP_SOCKET_OPEN_ERROR,
        SNTP_REQUEST_SEND_ERROR,
        SNTP_RESPONSE_RECEIVE_ERROR,
        SNTP_CLOCK_SYNC_ERROR,
    } eErrorCodes;

/******************************************************************************
* Function Prototypes
******************************************************************************/
/* Read Callback function */
int MyReadFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptObjectID, struct IEC61850_DataAttributeData *
ptReturnedValue);
/* Write Callback function */
int MyWriteFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptObjectID, const struct
IEC61850_DataAttributeData * ptNewValue);
/*! Operate Callback function */
int MyOperateFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptControlID, const struct
IEC61850_DataAttributeData * ptOperateValue, int iSyncroCheck, int iInterlockCheck);
/* Update callback function for GOOSE */
int MyUpdateFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptObjectID, const struct
IEC61850_DataAttributeData * ptNewValue);

/* Create Local Database */
void CreateLocalDatabase(void);
/* Get NTP Time */
extern int getTime(char *remoteIp,  signed int iTimeZoneHour,  unsigned int iTimeZoneMinutes,
              struct IEC61850_TimeStamp *ptIEC61850TimeStamp, int *errVal);
/* Input Handler Function */
void huge InputHandler(void);
```

```
/******************************************************************************
* Type definitions
******************************************************************************/

/* Input Handler Task Stack */
static unsigned int InHandler_stack[TASK_STACKSIZE];

/* Input Handler Task Definition Block */
static TaskDefBlock  InHandlerTask =
{
    InputHandler,                           // Task Function
    {'I','P','H','L'},                                  // a name: 4 chars
    &InHandler_stack[TASK_STACKSIZE],       // top of stack
    TASK_STACKSIZE*sizeof(int),             // size of stack
    0,                                      // attributes, not supported now
    IOHANDLER_PRIO,                         // priority 20(high) ... 127(low)
    0,                                      // time slice (if any), not supported now
    0,0,0,0                                 // mailbox depth, not supported now
};

/* Objects */
typedef struct tag_DK61Object
{
    unsigned char               ucObjectNo;         /* Object Number */
    unsigned char               ucObjectType;       /* Object Type */
    unsigned char               ucObjectValue;      /* Object Value */
    unsigned short int              usiObjectQuality;   /* Object Quality */
    struct IEC61850_TimeStamp   tObjectTime;        /* Obect Time */
}tDK61Object;

/* Full Quality */
typedef struct tag_FullQuality
{
    unsigned char               ucvalidity;         /* validity  */
    unsigned char               ucdetailQual;       /* detail quality */
    unsigned char               ucsource;           /* source */
    unsigned char                   uctest;             /* testing ? */
    unsigned char                   ucoperatorBlocked;  /* operator blocked */
}tFullQuality;


/* Local Database */
tDK61Object atObj[OBJECT_TYPES][OBJECTS];

/* IEC61850 Server */
IEC61850    myServer                            = 0;

/* IEC61850 Client */
IEC61850 myClient      = 1;

/* Remote NTP Server IP address */
char acRemoteIP[255] = {0};

char acDescString[255] = {0};

/* Flag for local Condition on Close after Trip*/
unsigned char    ucCDCLOSE;

/* Global variable to store specific field number for "local database" */
unsigned    int                             uiField;
unsigned    int                             uiFieldN;
/* Global INPUT variable */
unsigned    int                             uiDip_1;
unsigned    int                             uiDip_2;


/******************************************************************************
*   Functions Definitions
******************************************************************************/
/**
 * @brief Main Function
 *
 * @param argc - Argument Count
 * @param argv - Argument Vector
 *
 * @return 0
 */
 int main(int argc, char *argv[])
{
    struct IEC61850_Parameters  tServerParam    = {0};          // Server Parameters
    int                         taskID          = -1;                   // Task ID
    char                        *month[]        = { "---", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug",
"Sep", "Oct", "Nov", "Dec" };
    struct date                     date            = {0};
    struct time                     time            = {0};
    struct IEC61850_TimeStamp       tIEC61850Time   = {0};
    int                             errorval            = 0;
    enum IEC61850_ErrorCodes        error           = IEC61850_ERROR_NONE;


  printf("Library Version number: %d Build : %s", IEC61850_GetLibraryVersion(), IEC61850_GetLibraryBuildTime());

    /* Enable Programmable Chip Select for SC143 */
    pfe_enable_pcs( 6 );

    /* ICD file as parameter */
    if(argc < 3)
    {
        printf("\r\n Usage : GA61850 <FILENAME_Server.CID> <FILENAME_Client.SCD> [NTP Server]");
        return 0;
    }

    if(argc > 3)
    {
        strcpy(acRemoteIP, argv[3]);
    }
    else
    {
        strcpy(acRemoteIP, "203.161.12.165");
  }


  printf("\r\n NTP Remote IP : %s", acRemoteIP);

  // Set timezone environment variable to UTC
  putenv("TZ=UTC0");
 // putenv("TZ=UTC+2");   // can be tasted later if direct access to a time server is provided
  tzset();


  printf("\r\n Set Time Quality");
  error = IEC61850_SetTimeQuality(1, 0, 1, -1);
  if(error != IEC61850_ERROR_NONE)
  {
      printf("Failed %i : %s", error, IEC61850_ErrorString(error));
  }

  // Get NTP time
error = getTime( acRemoteIP, DFLTTIMEZONEHOUR, DFLTTIMEZONEMIN,  &tIEC61850Time, &errorval);

  if( error != API_ENOERROR)
  {
      printf("\r\n Get Time error : %i", error);
      /* If NTP Time failed Than Start Time is 1/1/2011 00:00:00 Hours */
      date.da_year = 11;
      date.da_mon = 1;
      date.da_day = 1;
      time.ti_hour = 00;
      time.ti_min = 00;
      time.ti_sec= 00;
      time.ti_hund = 00;
      tIEC61850Time.u32Seconds = dostounix(&date, &time); // Calculate Number of second
      tIEC61850Time.u32FractionsOfSecond = 0;
      printf("\r\n Set Time is 1/1/2011 00:00:00");
      error = IEC61850_SetTime(&tIEC61850Time);
```

```
    if(error != IEC61850_ERROR_NONE)
    {
        printf("Failed %i : %s", error, IEC61850_ErrorString(error));
    }
}
else
{
    unixtodos( tIEC61850Time.u32Seconds, &date, &time );
    /* NTP time get successful Set System Clock */
    printf( "\r\nDate: %s %u. %u", month[date.da_mon], date.da_day, date.da_year );
    printf( "\r\nTime: %u:%02u:%02u", time.ti_hour, time.ti_min, time.ti_sec);
    // Set system clock
    printf("\r\n Set Time");
    IEC61850_SetTime(&tIEC61850Time);
    if(error != IEC61850_ERROR_NONE)
    {
        printf("Failed %i : %s", error, IEC61850_ErrorString(error));
    }
    printf("\r\n Set Time Quality");
    IEC61850_SetTimeQuality(1, 0, 0, 10);
    if(error != IEC61850_ERROR_NONE)
    {
        printf("Failed %i : %s", error, IEC61850_ErrorString(error));
    }

}

/* Set all outputs to Off */
outportb(IO_ADDR, 0);

do
{

    tServerParam.ClientServerFlag   = IEC61850_SERVER;   // This is a Server
    tServerParam.uiOptions          = IEC61850_OPTION_NONE;  // Reserved for future use
    tServerParam.ptReadCallback     = MyReadFunction;        // Assign Read Callback function
    tServerParam.ptWriteCallback    = MyWriteFunction;       // Assign Write Callback function
```

```
        tServerParam.ptUpdateCallback   = NULL;              // No Update callback for Server
        tServerParam.ptSelectCallback   = NULL;                      // Ignoring Select commands
        tServerParam.ptOperateCallback  = MyOperateFunction; // Assign Operate Callback function (to work with
IEDScout)
        tServerParam.ptCancelCallback   = NULL;                  // Ignoring Cancel commands

        printf("\r\n Server Create");

        myServer = IEC61850_Create(&tServerParam,&error);  //Create a server
        if(myServer == NULL)
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
            break;
        }

        printf("\r\n Server Load SCL");
        error = IEC61850_LoadSCLFile(myServer,argv[1]); // Load in ICD file
        if(error != IEC61850_ERROR_NONE)
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
            break;
        }

        /* Create Local Database */
        CreateLocalDatabase();

        printf("\r\n Server Start");
        error = IEC61850_Start(myServer); // Starts myServer
        if(error != IEC61850_ERROR_NONE)
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
            break;
        }

      //code client start here

      else
      {
       enum IEC61850_ErrorCodes error = 0;   //Integer for error checking     (eErrorCodeClient)
       IEC61850 myClient;           //IEC61850 object
       struct IEC61850_Parameters tClientParam = {0};  //IEC61850 object parameters. This includes server/client type
and links to any callbacks to use

      tClientParam.ClientServerFlag     = IEC61850_CLIENT;  // This is a client
        tClientParam.uiOptions          = IEC61850_OPTION_NONE; // Reserved for future use
        tClientParam.ptReadCallback     = NULL;                 // Ignoring Read Callback function
        tClientParam.ptWriteCallback    = NULL;                      // Ignoring Write Callback function
        tClientParam.ptUpdateCallback = MyUpdateFunction;  // Update callback for Client/Subscriber
        tClientParam.ptSelectCallback   = NULL;                     // Ignoring Select commands
        tClientParam.ptOperateCallback  = NULL;                  // Ignoring Operate Callback function
        tClientParam.ptCancelCallback   = NULL;                 // Ignoring Cancel commands

        printf("\r\n Client Create");

        myClient = IEC61850_Create(&tClientParam,&error);  //Create a client
        if(myClient == NULL)
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
            break;
        }

        printf("\r\n Client Load SCL");
        error = IEC61850_LoadSCLFile(myClient,argv[2]); // Load in CID file
        if(error != IEC61850_ERROR_NONE)
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
            break;
        }
        else
        {
            outportb(IO_ADDR,1);   // set LED_1 to ON
            RTX_Sleep_Time(100);   // wait Breaker_settling time
        }
```

```
    printf("\r\n Client Start");
    error = IEC61850_Start(myClient); // Starts myServer
    if(error != IEC61850_ERROR_NONE)
    {
        printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
        break;
    }

  }  //End Else for Client start

}while(0); /* Dummy while loop to avoid nested If's */


/* Check for Any Error */
if(error == IEC61850_ERROR_NONE)
{
    /* Create and Start IO Handler Task */
    error = RTX_Create_Task(&taskID , &InHandlerTask);
    if(error != 0)
    {
        printf("\r\n IO Handler task Create Failed : %i", error);
    }

    /* Do not Exit */
    while(1)
    {
        RTX_Sleep_Time(30000);
    }
}
else
{
    /* If any errors in Create and Starting the Server or client */
    /* Stop the Server */
    printf("\r\n  Server and Client Stop");
    error = IEC61850_Stop(myServer);
    if(error != IEC61850_ERROR_NONE)
    {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
        }

        /* Free all Memory */
        printf("\r\n  Server  Free");
        IEC61850_Free(myServer);

      /*  If any errors in Create and Starting the Client */
        /* Stop the Client */
      error = IEC61850_Stop(myClient);
        if(error != IEC61850_ERROR_NONE)
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));
        }

        /* Free all Memory */
        printf("\r\n  Client  Free");
        IEC61850_Free(myClient);
     }

    return 0;
} // End of main


/**
 * @brief Function to create local database
 *
 */
void CreateLocalDatabase(void)
{
    unsigned char                           ucObjTypeCnt                    = 0;
    unsigned char                           ucObjects                       = 0;
    struct IEC61850_DataAttributeData           atUpdateValue[UPDATEMAX];               // Value to send on Change
    struct IEC61850_DataAttributeID_Generic     atObject[UPDATEMAX];                    // ID of the Object
    unsigned char                               nucUpdateCnt                    = 0;
    enum IEC61850_ErrorCodes                    error           = IEC61850_ERROR_NONE;
    /* Initalise memory */
    memset(&atUpdateValue[0], 0, (sizeof(struct IEC61850_DataAttributeData) * UPDATEMAX));
    memset(&atObject[0], 0, (sizeof(struct IEC61850_DataAttributeID_Generic) * UPDATEMAX));


    /* For all Object Types */
    for(ucObjTypeCnt = 0; ucObjTypeCnt < OBJECT_TYPES; ucObjTypeCnt++)
    {

        /* Each Object within Object Type */
        for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
        {
            /* Assign Object Number */
            atObj[ucObjTypeCnt][ucObjects].ucObjectNo   = ucObjects + 1;
            /* Assign Object Type DIP Switch (Input): 1 , LED (Output) : 2 */
            atObj[ucObjTypeCnt][ucObjects].ucObjectType     = ucObjTypeCnt + 1;

            /* Initialise Value to 0 */
            atObj[ucObjTypeCnt][ucObjects].ucObjectValue = 0;

            atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
            atObject[nucUpdateCnt].uiField1 = ucObjects + 1;                // Object Number
            atObject[nucUpdateCnt].uiField2 = ucObjTypeCnt + 1;             // Object Type
            atObject[nucUpdateCnt].uiField3 = VALUE_INDEX;
         atUpdateValue[nucUpdateCnt].pvData = &atObj[ucObjTypeCnt][ucObjects].ucObjectValue;   /*Set the value to
update*/

         /* uiField and uiFieldN to store Field 2 and 1 content*/
         uiField = atObject[nucUpdateCnt].uiField2;
         uiFieldN = atObject[nucUpdateCnt].uiField1;

          /* Handling of type: Make the 3rd entry point a double point status */
         if ((uiField == DIGITAL_INPUT)&&(uiFieldN == 3))
         {
         atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_DBPOS;

         }
         else
         {
           atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_BOOLEAN;
         } // End of type handleling section

            atUpdateValue[nucUpdateCnt].uiBitLength = 8;

            nucUpdateCnt++;

            if(uiField == DIGITAL_INPUT)
            {
                /* Initialise Quality */
                atObj[ucObjTypeCnt][ucObjects].usiObjectQuality = (IEC61850_QUALITY_FAILURE | IEC61850_QUALITY_INVALID
|
                                                                                  IEC61850_QUALITY_OLDDATA |
IEC61850_QUALITY_QUESTIONABLE );
                atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
                atObject[nucUpdateCnt].uiField1 = ucObjects + 1;               // Object Number
                atObject[nucUpdateCnt].uiField2 = ucObjTypeCnt + 1;            // Object Type
                atObject[nucUpdateCnt].uiField3 = QUALITY_INDEX;
                atUpdateValue[nucUpdateCnt].pvData = &atObj[ucObjTypeCnt][ucObjects].usiObjectQuality;
                atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_QUALITY;
                atUpdateValue[nucUpdateCnt].uiBitLength = IEC61850_QUALITY_BITSIZE;
                nucUpdateCnt++;
                /* Initialise Time */
                IEC61850_GetTime(&atObj[ucObjTypeCnt][ucObjects].tObjectTime);
                atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
                atObject[nucUpdateCnt].uiField1 = ucObjects + 1;               // Object Number
                atObject[nucUpdateCnt].uiField2 = ucObjTypeCnt + 1;            // Object Type
                atObject[nucUpdateCnt].uiField3 = TIME_STAMP_INDEX;
                atUpdateValue[nucUpdateCnt].pvData = &atObj[ucObjTypeCnt][ucObjects].tObjectTime;
                atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_TIMESTAMP;
                atUpdateValue[nucUpdateCnt].uiBitLength = IEC61850_TIMESTAMP_BITSIZE;
                nucUpdateCnt++;

            }
        }
```

```
    }

    if(nucUpdateCnt != 0)
    {
        /* Send Local Value to IEC 61850 Stack */
        printf("\r\n Server Update");
        error = IEC61850_Update(myServer, (struct IEC61850_DataAttributeID *)&atObject[0], &atUpdateValue[0],
nucUpdateCnt);
        if(error == -10)
        {
        printf("\r\nWarning : %i : %s\r\n John has reserved 2 Objects ID for future use.", error,
IEC61850_ErrorString(error));
        }
        if((error != IEC61850_ERROR_NONE)& (error != -10))
        {
            printf(" Failed : %i : %s", error, IEC61850_ErrorString(error));

        }
        nucUpdateCnt = 0;
    }


}

}

/**
 * @brief Write callback
 *
 * @return 0
 */
int MyWriteFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptObjectID, const struct
IEC61850_DataAttributeData * ptNewValue)
{
    struct IEC61850_DataAttributeID_Generic     *ptGenObjID     = NULL;
    enum IEC61850_ErrorCodes                error           = IEC61850_ERROR_NONE;
    // Nothing to write
    (void)ptUserData;   // If not used avoid compiler warning


    if(ptObjectID->daid_type == IEC61850_DAID_GENERIC)
    {

        /* Each Object within Object type */
        /* Check if the Field matches */
        if((ptGenObjID->uiField1 == STRING_ITEM) &&
            (ptGenObjID->uiField2 == STRING_VALUE))
        {

            if(ptGenObjID->uiField3 == VALUE_INDEX)
            {
                /* Return Value */
                memset(acDescString, 0, 255);
            }
        }
    }
   return error;
}

/**
 * @brief Operate callback
 *
 * @return 0
 */
int MyOperateFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptControlID, const struct
IEC61850_DataAttributeData * ptOperateValue, int iSyncroCheck, int iInterlockCheck)
{
    unsigned char                           ucLED           = 0;
    unsigned char                       ucObjects       = 0;
    unsigned char                           blLEDChange     = 0;
    unsigned char                           ucFound         = 0;
    struct IEC61850_DataAttributeID_Generic     *ptGenObjID     = NULL;

    (void)ptUserData;   // If not used avoid compiler warning
```

```
(void)iSyncroCheck;     // If not used avoid compiler warning
(void)iInterlockCheck;      // If not used avoid compiler warning

if(ptControlID->daid_type == IEC61850_DAID_GENERIC)
{
    ptGenObjID = (struct IEC61850_DataAttributeID_Generic   *)ptControlID;

    /* Each Object within Object type */
    for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
    {
        /* Check if the Field matches */
        if((ptGenObjID->uiField1 == atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectNo) &&
        ((ptGenObjID->uiField2 == atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectType)))
        {

            if(ptGenObjID->uiField3 == VALUE_INDEX)
            {
                /* Get the Value of the */
                memcpy(&ucLED, ptOperateValue->pvData, sizeof(unsigned char));
                if(ucLED != 0)
                {
                    ucLED = 1;

                }
                /* Check if the LED has changed */
                if(atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue != ucLED)
                {
                    /* Set the Value */
                    atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue = ucLED;
                    blLEDChange = 1;
                    ucFound = 1;
              ucCDCLOSE = 0; /* Ready to trip flag*/
        //    printf("\r\n GOOSE recieved -->");
                }
            }
        }
```

```
            if(ucFound) break;
        }

        /* LED Changed */
        if(blLEDChange)
        {
            ucLED = 0;
            /* Get all the values of the LED */
            for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
            {
                /* Form Byte to Output */
                ucLED = (ucLED | (atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue << ucObjects));
            }
#ifdef DEBUG_SV61850
            printf("\r\n Write LED : %X ", ucLED);
#endif
            /* Output the LED */
            outportb(IO_ADDR,ucLED);
        }
    }
    return IEC61850_COMMAND_ERROR_NONE;
}

/**
 * Brief Update function call back by the stack when a GOOSE
 * is recieved
 */

  int MyUpdateFunction(void * ptUserData, struct IEC61850_DataAttributeID * ptObjectID, const struct
IEC61850_DataAttributeData * ptNewValue)
{
   unsigned char                            ucLED           = 0;
    unsigned char                       ucObjects       = 0;
    unsigned char                           blLEDChange     = 0;
    unsigned char                           ucFound         = 0;

   struct IEC61850_DataAttributeID_Generic  *ptGenObjID     = NULL;
   int returnvalue = IEC61850_CB_ERROR_NONE;


   IEC61850 myClient = (IEC61850)ptUserData;

        if(ptObjectID->daid_type == IEC61850_DAID_GENERIC)
    {

        ptGenObjID = (struct IEC61850_DataAttributeID_Generic   *)ptObjectID;

        /* Each Object within Object type */
        for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
        {
            /* Check if the Field matches */
            if((ptGenObjID->uiField1 == atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectNo) &&
            ((ptGenObjID->uiField2 == atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectType)))
            {

                if(ptGenObjID->uiField3 == VALUE_INDEX)
                {
                    /* Get the Value of the */
                    memcpy(&ucLED, ptNewValue->pvData, sizeof(unsigned char));
                    if(ucLED != 0)
                    {
                        ucLED = 1;
                    }

                  if (ucCDCLOSE == 0)    /* test the Ready to trip flag*/
                   {

                    /* Check if the LED has changed */
                     if(atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue == ucLED) //  verify if the GOOSE value is true
                     {
                        /* Set the Value */
                        atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue = !ucLED; // If GOOSE=1 then swith off
correspondind LED
                        blLEDChange = 1;
                    ucFound = 1;
                 }

           }

            }
        }

        if(ucFound) break;
    }

    /* LED Changed */
    if(blLEDChange)
    {
        ucLED = 0;
        /* Get all the values of the LED */
        for(ucObjects = 0; ucObjects < OBJECTS;  ucObjects++)
        {
            /* Form Byte to Output */
            ucLED = (ucLED | (atObj[DIGOUTPUT_INDEX][ucObjects].ucObjectValue << ucObjects));
        }

        /* Output the LED */
        outportb(IO_ADDR,ucLED);

    }

  /*If breaker has been closed by hand operation*/
  if((blLEDChange == 0) && (ucCDCLOSE == 0))
  {
    /*Force breaker to trip */
        outportb(IO_ADDR,0);
  }

}

return(returnvalue);
```

```
}

/**
 * @brief Input Handler Task.
 *
 */
void huge InputHandler(void)
{
    unsigned char                               ucDIPValue                      = 0;        // DIP Switch Value
    unsigned char                               ucPrevDIPValue                  = 0;        // Previous DIP Switch
Value
    unsigned char                               nucObjects                      = 0;        // Total Objects
    struct IEC61850_DataAttributeData           atUpdateValue[UPDATEMAX];       // Value to send on Change
    struct IEC61850_DataAttributeID_Generic     atObject[UPDATEMAX];            // ID of the Object
    unsigned char                           ucObjectVal                         = 0;        // Local Object Value
    unsigned short int                          usiQuality                      = 0;    // Local Quality
    unsigned char                           nucUpdateCnt                    = 0;
    int                                     iReturnErrCode                  = API_ENOERROR;
    int                                         errorval                            = 0;
    struct IEC61850_TimeStamp                   tIEC61850Time                   = {0};
    enum IEC61850_ErrorCodes                    error                           = IEC61850_ERROR_NONE;


    while(1)    // Indefinite Loop
    {
        iReturnErrCode = getTime( acRemoteIP, DFLTTIMEZONEHOUR, DFLTTIMEZONEMIN,  &tIEC61850Time, &errorval);
        if(iReturnErrCode  == API_ENOERROR)
        {
            if(errorval == SNTP_CLOCK_SYNC_ERROR)
            {
                error = IEC61850_SetTimeQuality(0, 0, 1, 10 );
                if(error != IEC61850_ERROR_NONE)
                {
                    printf(" Set Time Quality Failed : %i : %s ",error,  IEC61850_ErrorString(error));
                }

            }
            else
            {
                error = IEC61850_SetTimeQuality(0, 0, 0, 10 );
                if(error != IEC61850_ERROR_NONE)
                {
                    printf(" Set Time Quality Failed : %i : %s ",error,  IEC61850_ErrorString(error));
                }
                error = IEC61850_SetTime(&tIEC61850Time);
                if(error != IEC61850_ERROR_NONE)
                {
                    printf(" Set Time Failed : %i : %s ",error,  IEC61850_ErrorString(error));
                }
            }

        }
        else
        {

        // printf("\r\n NTP Time Error : %i : %i", iReturnErrCode, errorval);
            error = IEC61850_SetTimeQuality(0, 0, 1, 10);
            if(error != IEC61850_ERROR_NONE)
            {
                printf(" Set Time Quality Failed : %i : %s ",error,  IEC61850_ErrorString(error));
            }
        }

        /* Read INPUT Switch Value */
        ucDIPValue = inportb(IO_ADDR);

        /* If previous value does not match current value */
        if(ucPrevDIPValue != ucDIPValue)
        {
            nucUpdateCnt = 0;
            memset(&atUpdateValue[0], 0, (sizeof(struct IEC61850_DataAttributeData) * UPDATEMAX));
            memset(&atObject[0], 0, (sizeof(struct IEC61850_DataAttributeID_Generic) * UPDATEMAX));

#ifdef DEBUG_SV61850
```

```
            printf("\r\n DIP Value : %X", ucDIPValue);
#endif

            /* Check which INPUT  has changed (Dip_1 or Dip_2?) */
         //for(nucObjects = 0; nucObjects < OBJECTS; nucObjects++)
        for(nucObjects = 0; nucObjects < 2; nucObjects++)
            {
                /* Get the Values which have changed */
                ucObjectVal = ((ucDIPValue & (1 << nucObjects)) >> nucObjects);
                if(ucObjectVal != atObj[0][nucObjects].ucObjectValue)
                {
#ifdef DEBUG_SV61850
                    printf("\r\n  %u : %X",  atObj[0][nucObjects].ucObjectNo, ucObjectVal);
#endif

                    /* Object Value */
                    /* Object Value Index */
                    atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
                    atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
                    atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
                    atObject[nucUpdateCnt].uiField3 = VALUE_INDEX;
                    /* Update Value in the database */
                    atObj[0][nucObjects].ucObjectValue       = ucObjectVal;
                    atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].ucObjectValue;
                   atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_BOOLEAN;

                   /* uiField and uiFieldN to store Field 2 and 1 content*/
                   uiField = atObject[nucUpdateCnt].uiField2;
                   uiFieldN = atObject[nucUpdateCnt].uiField1;

                    atUpdateValue[nucUpdateCnt].uiBitLength = 8;

                    /* Send Update for Value */
                    nucUpdateCnt++;

                  /* Object Quality  */
                    usiQuality = 0;
```

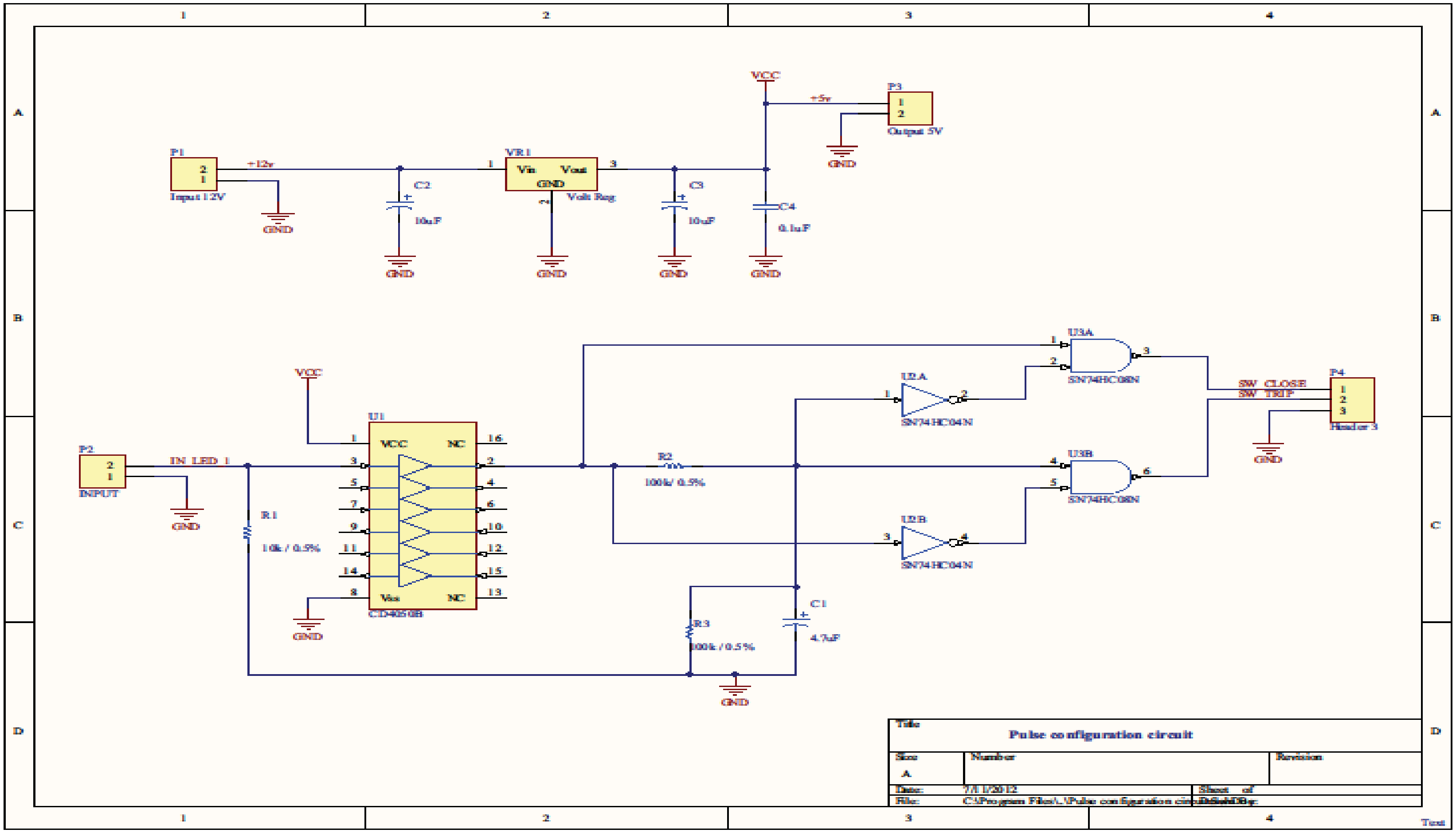
```
  /* No way to determine if DIP Switch failed so */
  atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
  atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
  atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
  atObject[nucUpdateCnt].uiField3 = QUALITY_INDEX;
  /* Update Quality in the database */
  atObj[0][nucObjects].usiObjectQuality    = usiQuality;
  atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].usiObjectQuality;
  atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_QUALITY;
  atUpdateValue[nucUpdateCnt].uiBitLength = IEC61850_QUALITY_BITSIZE;

  /* Send Update for Quality */
  nucUpdateCnt++;

  /* Send Time */
  /* Convert to 61850 Time */

  IEC61850_GetTime(&tIEC61850Time);
  atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
  atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
  atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
  atObject[nucUpdateCnt].uiField3 = TIME_STAMP_INDEX;
  /* Update Time in the database */
  memcpy(&atObj[0][nucObjects].tObjectTime, &tIEC61850Time, sizeof(struct IEC61850_TimeStamp));
  atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].tObjectTime;
  atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_TIMESTAMP;
  atUpdateValue[nucUpdateCnt].uiBitLength = IEC61850_TIMESTAMP_BITSIZE;

  /* Send Update for Time Stamp */
  nucUpdateCnt++;

/* Conditional  affectation of a new value for Dip_1 */
  if ((uiField == DIGITAL_INPUT)&&(uiFieldN == 1))
  {
  uiDip_1 = atObj[0][nucObjects].ucObjectValue;
  }
```

```
           /* Conditional  affectation of a new value for Dip_2 */
           else if ((uiField == DIGITAL_INPUT)&&(uiFieldN == 2))
           {
           uiDip_2 = atObj[0][nucObjects].ucObjectValue;
           }

           else    //do no thing
           {
           }
           /*End conditional  affectation of new value for Dip_1 and Dip_2 */

       }  //End IF

   }  //End For  loop to read dip_1 and Dip_2

/* For Loop to Update XCBR/Pos/stVal in the local Data base */
for(nucObjects = 2; nucObjects < 3; nucObjects++)
{
  /* Get the Values which have changed */
   /* Conditional Biding of Dip_1 and Dip_2 */
   / * to load new value for XCBR/Pos/stVal */
            if((uiDip_2 == FALSE) && (uiDip_1 == FALSE))
           {
          ucObjectVal = IEC61850_DB_POS_OFF;   // Double Position is in transitionnal state
           }

           else if((uiDip_2 == FALSE) && (uiDip_1 == TRUE))
           {
          ucObjectVal = IEC61850_DB_POS_FALSE; // Double Position is false/open
          outportb(IO_ADDR,0);      /*As per hand operation*/
          ucCDCLOSE = 1; /* Prevent from a direct reclosur after fault*/

           }

           else if((uiDip_2 == TRUE) && (uiDip_1 == FALSE))
           {
          ucObjectVal = IEC61850_DB_POS_TRUE;  // Double Position is close/true
   outportb(IO_ADDR,1);      /*As per hand operation*/
   ucCDCLOSE = 0; /* Reset the ready to  trip flag */
    }

    else
    {
   ucObjectVal = IEC61850_DB_POS_INVALID;   // Double Position is in a invalid state

    }
    //End IF selector based on change Dip_1 or Dip_2

if(ucObjectVal != atObj[0][nucObjects].ucObjectValue)
    {

    /* Object Value */
    /* Object Value Index */
    atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
    atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
    atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
    atObject[nucUpdateCnt].uiField3 = VALUE_INDEX;
    /* Update Value in the database */
    atObj[0][nucObjects].ucObjectValue       = ucObjectVal;
    atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].ucObjectValue;
   atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_DBPOS;

    atUpdateValue[nucUpdateCnt].uiBitLength = 8;

    /* Send Update for Value */
    nucUpdateCnt++;

 /* Object Quality  */
    usiQuality = 0;

    /* No way to determine if DIP Switch failed so */
    atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
    atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
    atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
```

```
                    atObject[nucUpdateCnt].uiField3 = QUALITY_INDEX;
                    /* Update Quality in the database */
                    atObj[0][nucObjects].usiObjectQuality    = usiQuality;
                    atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].usiObjectQuality;
                    atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_QUALITY;
                    atUpdateValue[nucUpdateCnt].uiBitLength = IEC61850_QUALITY_BITSIZE;

                    /* Send Update for Quality */
                    nucUpdateCnt++;

                    /* Send Time */
                    /* Convert to 61850 Time */

                    IEC61850_GetTime(&tIEC61850Time);
                    atObject[nucUpdateCnt].Generic_type = IEC61850_DAID_GENERIC; // Set Object Type to Generic
                    atObject[nucUpdateCnt].uiField1 = nucObjects + 1; // Object Number
                    atObject[nucUpdateCnt].uiField2 = DIGITAL_INPUT;  // Object Type
                    atObject[nucUpdateCnt].uiField3 = TIME_STAMP_INDEX;
                    /* Update Time in the database */
                    atUpdateValue[nucUpdateCnt].pvData = &atObj[0][nucObjects].tObjectTime;
                    atUpdateValue[nucUpdateCnt].ucType = IEC61850_DATATYPE_TIMESTAMP;
                    atUpdateValue[nucUpdateCnt].uiBitLength = IEC61850_TIMESTAMP_BITSIZE;

                    /* Send Update for Time Stamp */
                    nucUpdateCnt++;

                    }  //End IF

        } // ENd for loop to update XCBR/Pos/stVal in the data base

          /*General update */
            if(nucUpdateCnt != 0)
            {
                error = IEC61850_Update(myServer, (struct IEC61850_DataAttributeID *)&atObject[0], &atUpdateValue[0],
nucUpdateCnt);
                 if(error == -10)
                {// Do nothing
                }
                if((error != IEC61850_ERROR_NONE)& (error != -10))
                {
                    printf(" Update Failed : %i : %s ",error,  IEC61850_ErrorString(error));
                }
                nucUpdateCnt = 0;
            } //end General Update

            ucPrevDIPValue =   ucDIPValue;
        } //end If
    //  RTX_Sleep_Time((unsigned int)50); /* Increase the GOOSE Transfer time*/


    }  // End While
}     // End huge InputHandler
```

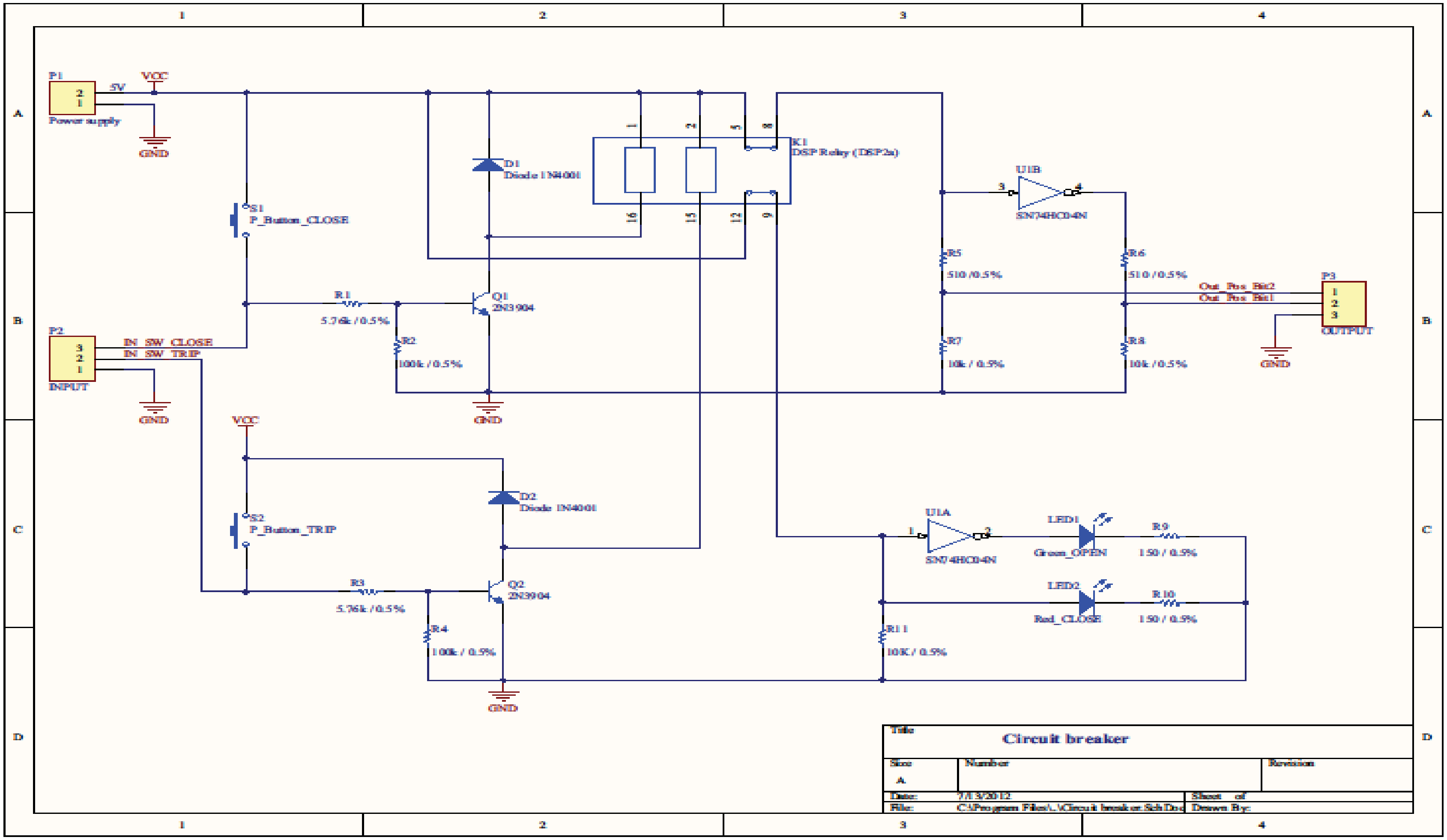
APPENDIX G

APPENDIX G: Pulse Control Logic Diagram



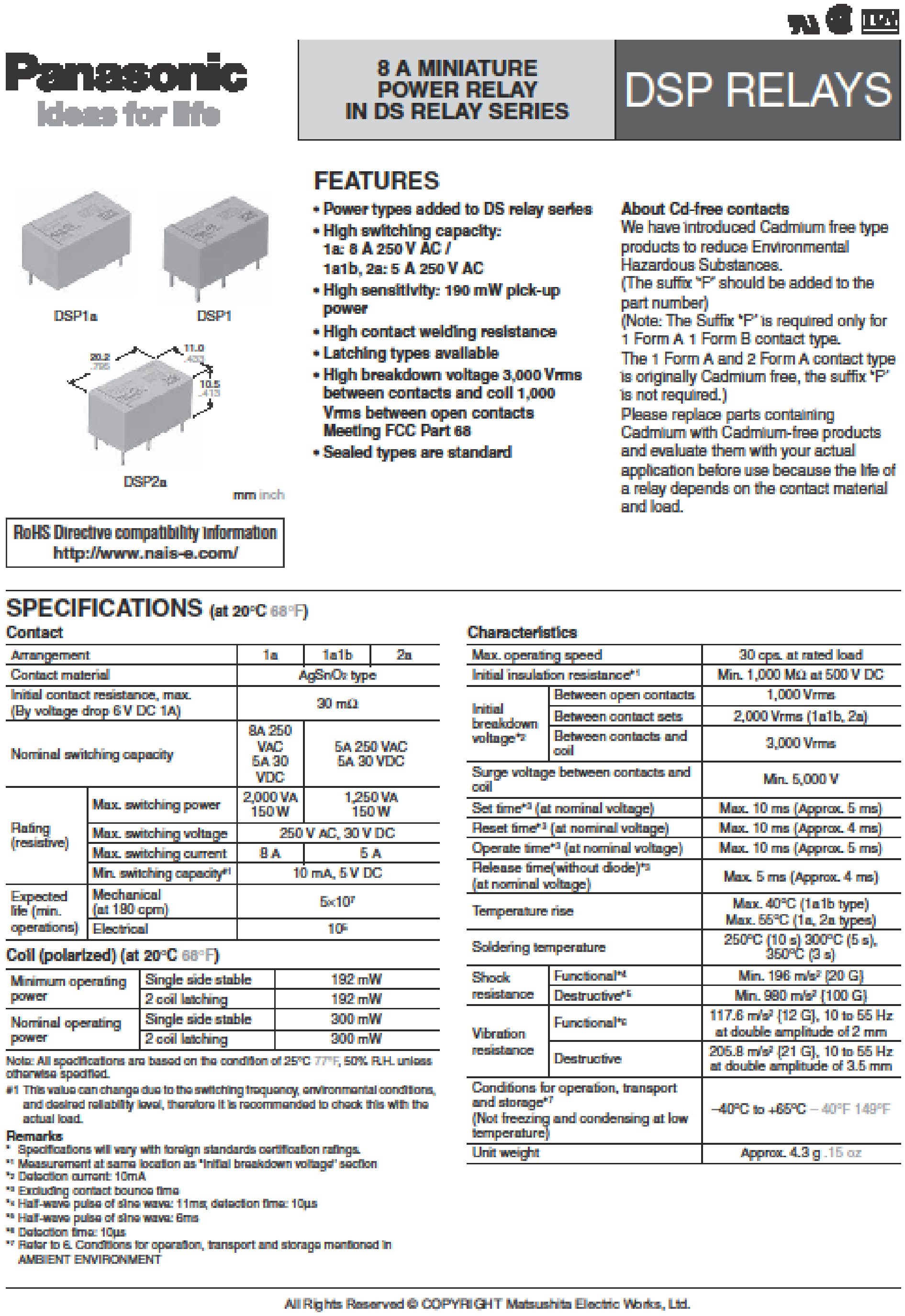
APPENDIX H

APPENDIX H: Emulated Circuit Breaker Diagram



APPENDIX I

APPENDIX I: DSP2a-PSL2 Data Sheet (use for emulated Circuit Breaker)



DSP1a
DSP1
20.2
.795
11.0
.433
10.5
.413
DSP2a
mm inch

Panasonic
ideas for life

8 A MINIATURE
POWER RELAY
IN DS RELAY SERIES

DSP RELAYS

FEATURES

- Power types added to DS relay series
- High switching capacity:
  1a: 8 A 250 V AC /
  1a1b, 2a: 5 A 250 V AC
- High sensitivity: 190 mW pick-up power
- High contact welding resistance
- Latching types available
- High breakdown voltage 3,000 Vrms between contacts and coil 1,000 Vrms between open contacts Meeting FCC Part 68
- Sealed types are standard

About Cd-free contacts
We have introduced Cadmium free type products to reduce Environmental Hazardous Substances.
(The suffix "F" should be added to the part number)
(Note: The Suffix "F" is required only for 1 Form A 1 Form B contact type.
The 1 Form A and 2 Form A contact type is originally Cadmium free, the suffix "F" is not required.)
Please replace parts containing Cadmium with Cadmium-free products and evaluate them with your actual application before use because the life of a relay depends on the contact material and load.

RoHS Directive compatibility information
http://www.nais-e.com/

SPECIFICATIONS (at 20°C 68°F)

Contact

		1a	1a1b	2a
Arrangement		1a	1a1b	2a
Contact material		AgSnO₂ type		
Initial contact resistance, max. (By voltage drop 6 V DC 1A)		30 mΩ		
Nominal switching capacity		8A 250 VAC 5A 30 VDC	5A 250 VAC 5A 30 VDC	
Rating (resistive)	Max. switching power	2,000 VA 150 W	1,250 VA 150 W	
	Max. switching voltage	250 V AC, 30 V DC		
	Max. switching current	8 A	5 A	
	Min. switching capacity[#1]	10 mA, 5 V DC		
Expected life (min. operations)	Mechanical (at 180 cpm)	$5{\times}10^7$		
	Electrical	10^5		

Coil (polarized) (at 20°C 68°F)

Minimum operating power	Single side stable	192 mW
	2 coil latching	192 mW
Nominal operating power	Single side stable	300 mW
	2 coil latching	300 mW

Note: All specifications are based on the condition of 25°C 77°F, 50% R.H. unless otherwise specified.

[#1] This value can change due to the switching frequency, environmental conditions, and desired reliability level, therefore it is recommended to check this with the actual load.

Characteristics

Max. operating speed		30 cps. at rated load
Initial insulation resistance[*1]		Min. 1,000 MΩ at 500 V DC
Initial breakdown voltage[*2]	Between open contacts	1,000 Vrms
	Between contact sets	2,000 Vrms (1a1b, 2a)
	Between contacts and coil	3,000 Vrms
Surge voltage between contacts and coil		Min. 5,000 V
Set time[*3] (at nominal voltage)		Max. 10 ms (Approx. 5 ms)
Reset time[*3] (at nominal voltage)		Max. 10 ms (Approx. 4 ms)
Operate time[*3] (at nominal voltage)		Max. 10 ms (Approx. 5 ms)
Release time(without diode)[*3] (at nominal voltage)		Max. 5 ms (Approx. 4 ms)
Temperature rise		Max. 40°C (1a1b type) Max. 55°C (1a, 2a types)
Soldering temperature		250°C (10 s) 300°C (5 s), 350°C (3 s)
Shock resistance	Functional[*4]	Min. 196 m/s² {20 G}
	Destructive[*5]	Min. 980 m/s² {100 G}
Vibration resistance	Functional[*6]	117.6 m/s² {12 G}, 10 to 55 Hz at double amplitude of 2 mm
	Destructive	205.8 m/s² {21 G}, 10 to 55 Hz at double amplitude of 3.5 mm
Conditions for operation, transport and storage[*7] (Not freezing and condensing at low temperature)		–40°C to +65°C –40°F 149°F
Unit weight		Approx. 4.3 g .15 oz

Remarks

- * Specifications will vary with foreign standards certification ratings.
- [*1] Measurement at same location as "Initial breakdown voltage" section
- [*2] Detection current: 10mA
- [*3] Excluding contact bounce time
- [*4] Half-wave pulse of sine wave: 11ms; detection time: 10µs
- [*5] Half-wave pulse of sine wave: 6ms
- [*6] Detection time: 10µs
- [*7] Refer to 6. Conditions for operation, transport and storage mentioned in AMBIENT ENVIRONMENT

Full Datasheet available at:

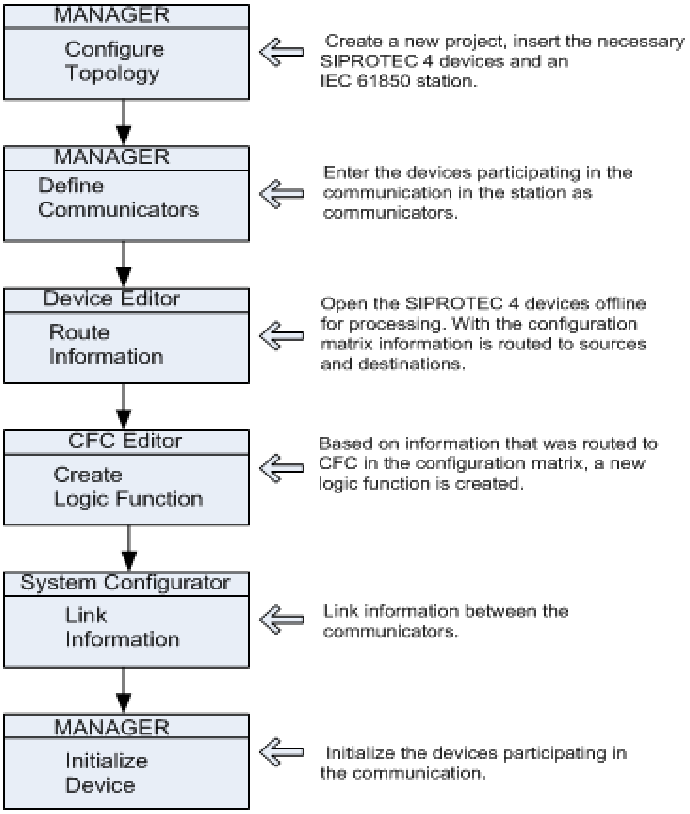
http://www.rlocman.ru/i/File/dat/Panasonic/Power_General_Purpose/DSP1_DC24V_F.pdf

APPENDIX J

APPENDIX J: Equipment and environment set up for an Overcurrent test

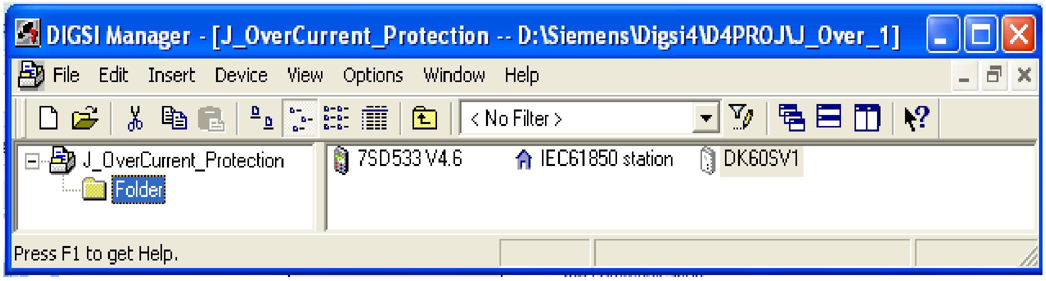
J1: SIEMENS relay set up

Preparing a SIPROTEC device for a current injection test in order to simulate a fault is straight forward. Due to the number of configuration steps and to not lose the focus of what this chapter is about, the configuration description has been reduced to a minimum. Figure_A J.1 outlines the steps required for the configuration of a SIPROTEC device.


MANAGER
Configure Topology
Create a new project, insert the necessary SIPROTEC 4 devices and an IEC 61850 station.
MANAGER
Define Communicators
Enter the devices participating in the communication in the station as communicators.
Device Editor
Route Information
Open the SIPROTEC 4 devices offline for processing. With the configuration matrix information is routed to sources and destinations.
CFC Editor
Create Logic Function
Based on information that was routed to CFC in the configuration matrix, a new logic function is created.
System Configurator
Link Information
Link information between the communicators.
MANAGER
Initialize Device
Initialize the devices participating in the communication.

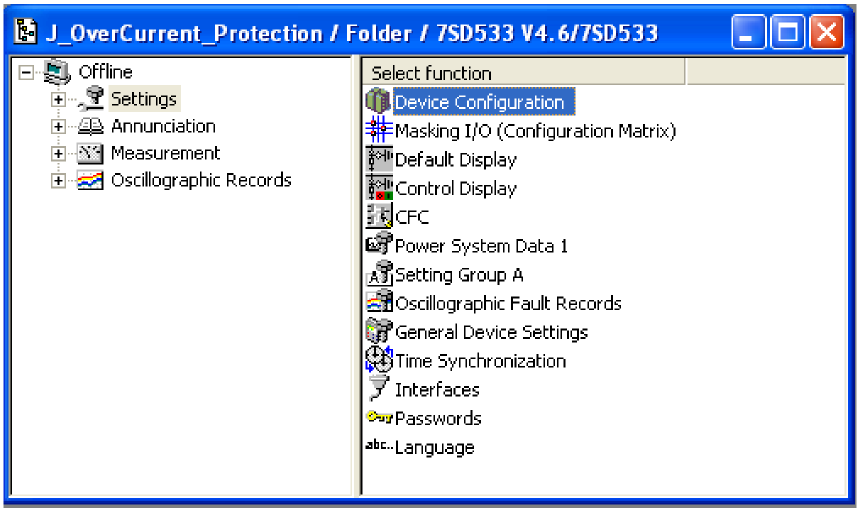
Figure_A J.1: Configuration device steps in DIGSI 4
(Siemens, 2009)

SIEMENS has developed a unique platform that serves as both for configuration of device settings and also for communication parameter definition. This platform is the DGISI Manager as illustrated in Figure_A J.2.



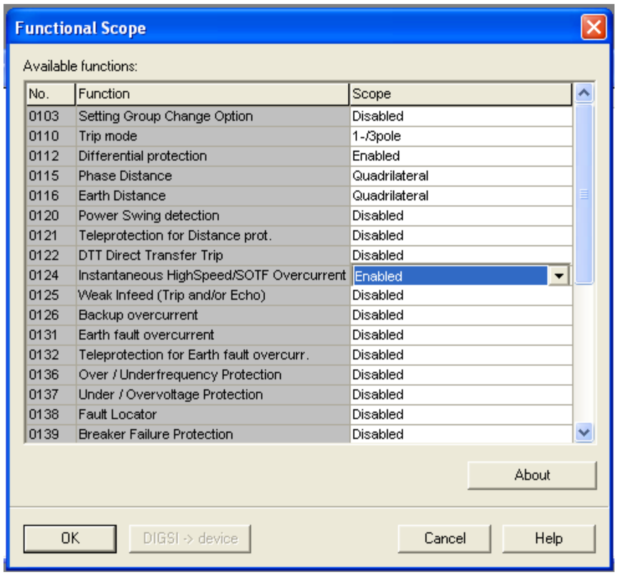
Figure_A J.2: Main project view of devices in DIGSI Manager

In order to configure the device, one needs to connect to it in an offline mode via Ethernet or RS-232 connection. From the device library add the specific device to the project. To connect, double-click on the desired SIPROTEC device. On the screen that appears, select the mode of connection that is appropriate, and continues until the windows as illustrated in Figure_A J.3 appears.



Figure_A J.3: Offline connection to IED_1

To view the diverse functions that are available, double-click on “Device Configuration” (see Figure_A J.3). The Functional Scope window Figure_A J.4 below appears as shown. Please make sure that the “Instantaneous HighSpeedSOTF Overcurrent” function is enabled as by default it is usually disabled.

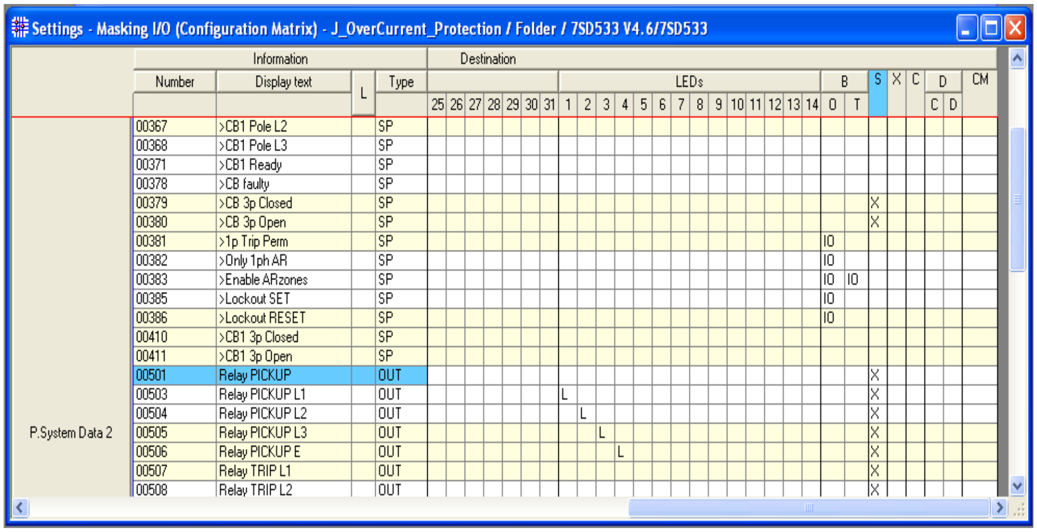


Available functions:

No.	Function	Scope
0103	Setting Group Change Option	Disabled
0110	Trip mode	1-/3pole
0112	Differential protection	Enabled
0115	Phase Distance	Quadrilateral
0116	Earth Distance	Quadrilateral
0120	Power Swing detection	Disabled
0121	Teleprotection for Distance prot.	Disabled
0122	DTT Direct Transfer Trip	Disabled
0124	Instantaneous HighSpeed/SOTF Overcurrent	Enabled
0125	Weak Infeed (Trip and/or Echo)	Disabled
0126	Backup overcurrent	Disabled
0131	Earth fault overcurrent	Disabled
0132	Teleprotection for Earth fault overcurr.	Disabled
0136	Over / Underfrequency Protection	Disabled
0137	Under / Overvoltage Protection	Disabled
0138	Fault Locator	Disabled
0139	Breaker Failure Protection	Disabled

Figure_A J.4: Device configuration Panel view

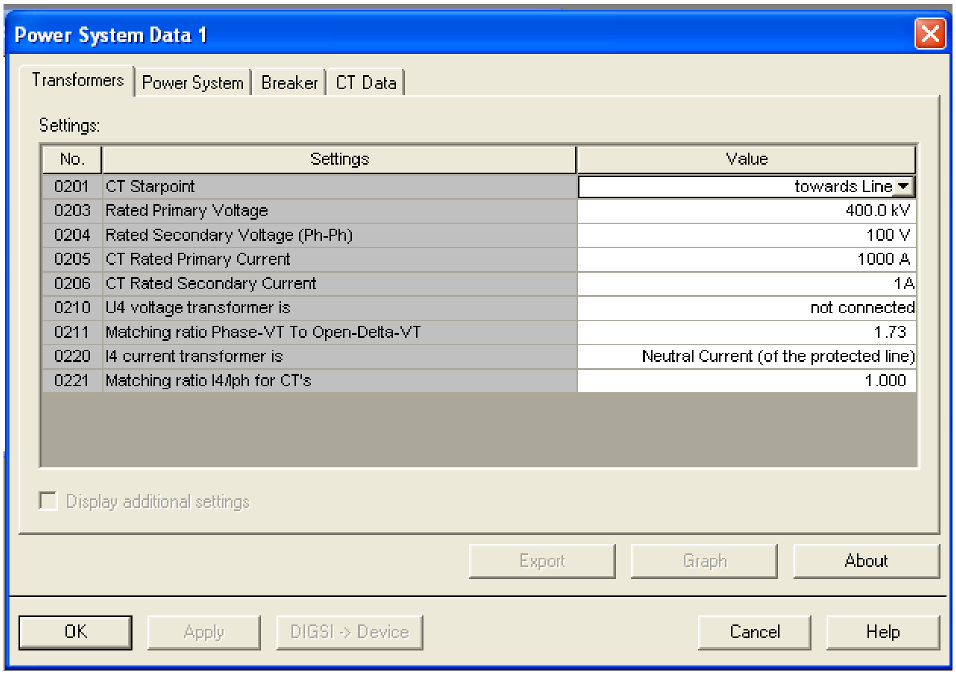
From Figure_A J.3, the Input/ Output Configuration Matrix (Figure_A J.5 below) can also be accessed. To proceed, double-click on the “Masking I/O” menu. Figure_A J.5 shows the destination routing of the “Relay PICKUP” which has been routed to the system interface in order to be available for the GOOSE mapping procedure.


Settings - Masking I/O (Configuration Matrix) - J_OverCurrent_Protection / Folder / 7SD533 V4.6/7SD533

	Information				Destination																												
	Number	Display text	L	Type								LEDs														B		S	X	C	D		CM
					25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	O	T				C	D	
	00367	>CB1 Pole L2		SP																													
	00368	>CB1 Pole L3		SP																													
	00371	>CB1 Ready		SP																													
	00378	>CB faulty		SP																													
	00379	>CB 3p Closed		SP																								X					
	00380	>CB 3p Open		SP																								X					
	00381	>1p Trip Perm		SP																						IO							
	00382	>Only 1ph AR		SP																						IO							
	00383	>Enable ARzones		SP																						IO	IO						
	00385	>Lockout SET		SP																						IO							
	00386	>Lockout RESET		SP																						IO							
	00410	>CB1 3p Closed		SP																													
	00411	>CB1 3p Open		SP																													
	00501	Relay PICKUP		OUT																								X					
	00503	Relay PICKUP L1		OUT								L																X					
	00504	Relay PICKUP L2		OUT									L															X					
P.System Data 2	00505	Relay PICKUP L3		OUT										L														X					
	00506	Relay PICKUP E		OUT											L													X					
	00507	Relay TRIP L1		OUT																								X					
	00508	Relay TRIP L2		OUT																								X					

Figure_A J.5: Input/ Output configuration matrix for Relay Pickup

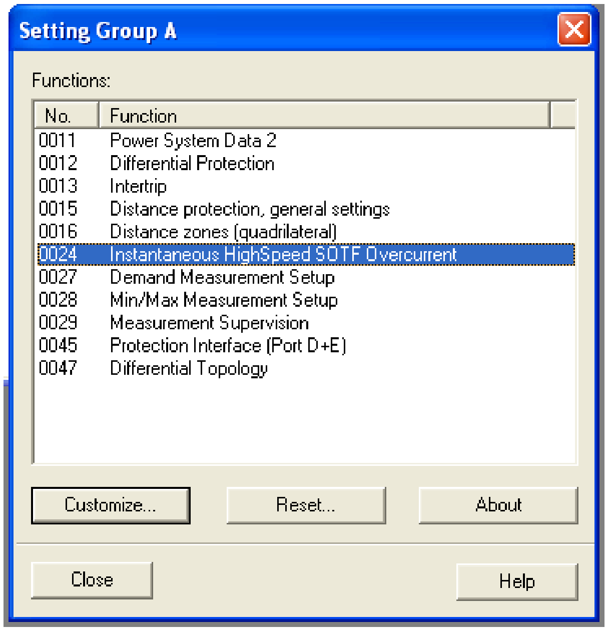
From Figure_A J.3, you may also review the Power System Data settings by a double click on “Power System Data 1”. A window as depicted in Figure_A J.6 appears.



No.	Settings	Value
0201	CT Starpoint	towards Line
0203	Rated Primary Voltage	400.0 kV
0204	Rated Secondary Voltage (Ph-Ph)	100 V
0205	CT Rated Primary Current	1000 A
0206	CT Rated Secondary Current	1A
0210	U4 voltage transformer is	not connected
0211	Matching ratio Phase-VT To Open-Delta-VT	1.73
0220	I4 current transformer is	Neutral Current (of the protected line)
0221	Matching ratio I4/Iph for CT's	1.000

Figure_A J.6: Configuration of Power system Data 1

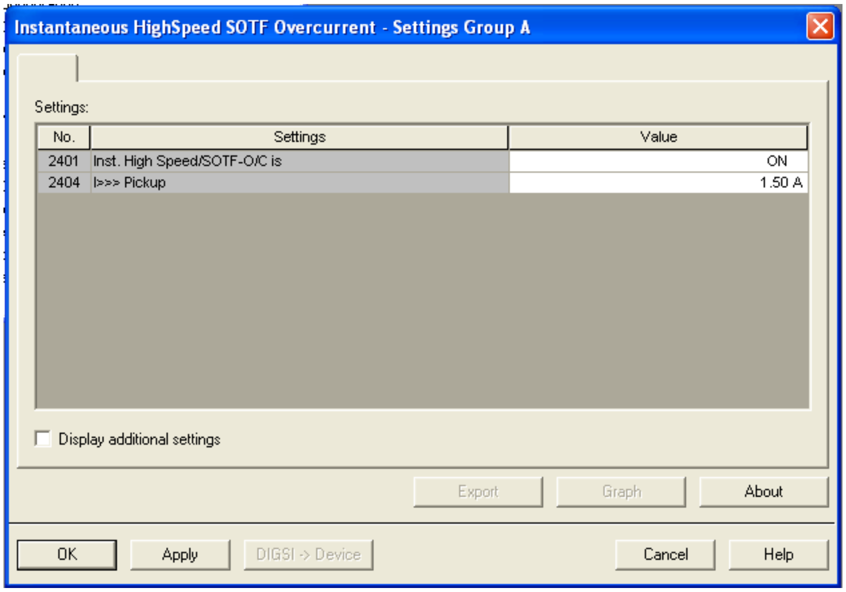
From the main menu of the device setting (see Figure_A J.3), double click on “Setting Group A” then select the “Instantaneous HighSpeed SOTF Overcurrent” and customise the relevant settings to the requirements.



No.	Function
0011	Power System Data 2
0012	Differential Protection
0013	Intertrip
0015	Distance protection, general settings
0016	Distance zones (quadrilateral)
0024	Instantaneous HighSpeed SOTF Overcurrent
0027	Demand Measurement Setup
0028	Min/Max Measurement Setup
0029	Measurement Supervision
0045	Protection Interface (Port D+E)
0047	Differential Topology

Figure_A J.7: Main view of the Setting Group A

For the current project, the “Inst. High Speed/SOTF O/C” has been set to ON as illustrated in Figure_A J.8.



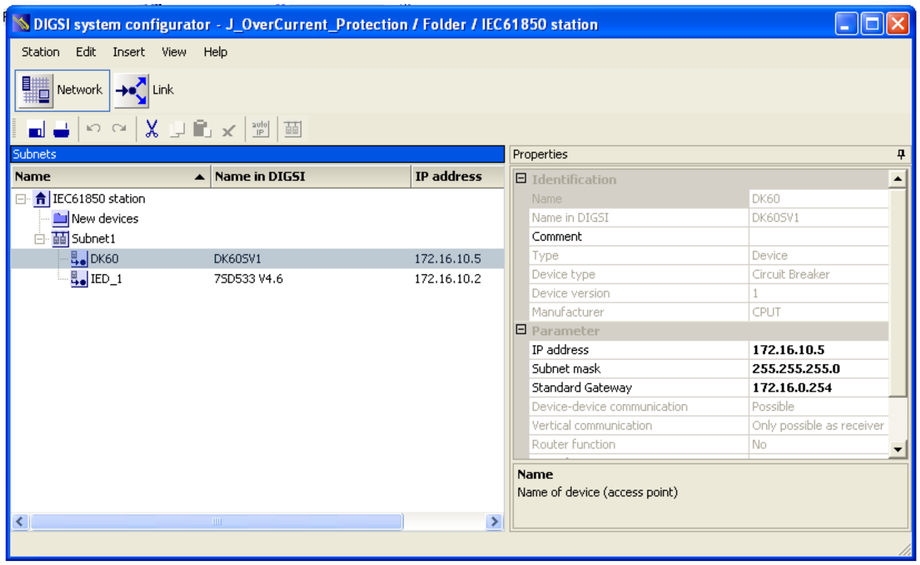
Settings:

No.	Settings	Value
2401	Inst. High Speed/SOTF-O/C is	ON
2404	I>>> Pickup	1.50 A

□ Display additional settings

Figure_A J.8: Configuration for HighSpeed Overcurrent module

After the configuration of the basic functions of the device, proceed to the “IEC61850 substation” communication design. From the DIGSI Manager main view, double click on “IEC 61850 station”. The screen presented at Figure_A J.9 appears.


DIGSI system configurator - J_OverCurrent_Protection / Folder / IEC61850 station
Station
Edit
Insert
View
Help
Network
Link
Subnets
Properties

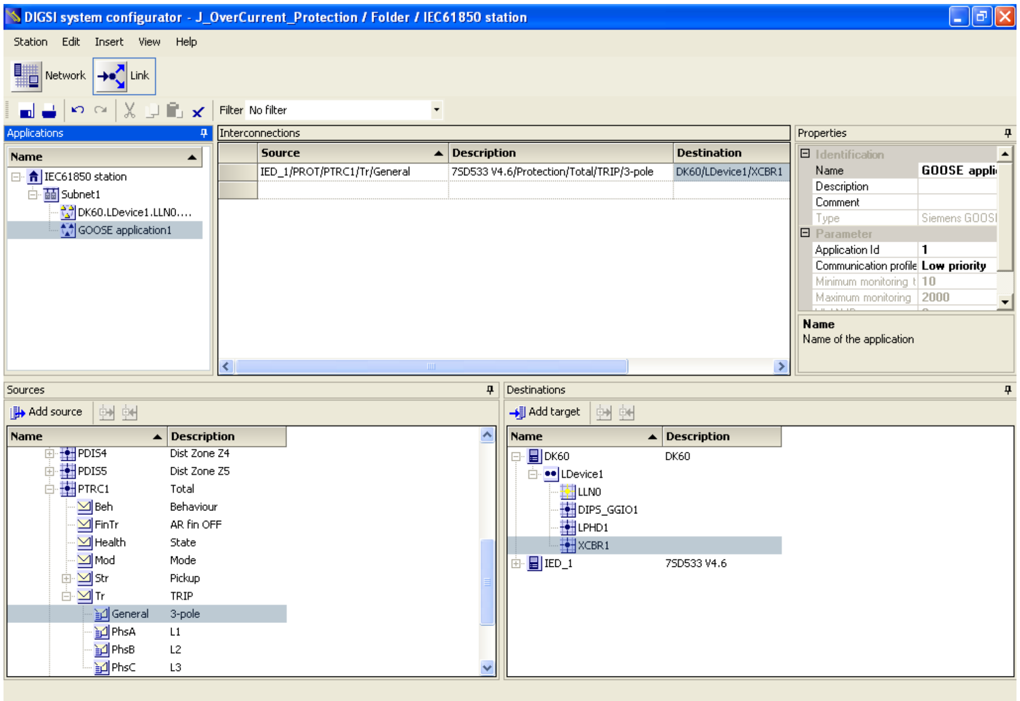
Name	Name in DIGSI	IP address
IEC61850 station		
New devices		
Subnet1		
DK60	DK60SV1	172.16.10.5
IED_1	7SD533 V4.6	172.16.10.2

Identification	
Name	DK60
Name in DIGSI	DK60SV1
Comment	
Type	Device
Device type	Circuit Breaker
Device version	1
Manufacturer	CPUT
Parameter	
IP address	**172.16.10.5**
Subnet mask	**255.255.255.0**
Standard Gateway	**172.16.0.254**
Device-device communication	Possible
Vertical communication	Only possible as receiver
Router function	No

Name
Name of device (access point)

Figure_A J.9: Network topology for the IEC 61850 station

To access the GOOSE mapping area click on “Link” as illustrated in Figure_A J.10 below.



Filter: No filter

Applications

Name
IEC61850 station
Subnet1
DK60.LDevice1.LLN0....
GOOSE application1

Interconnections

Source	Description	Destination
IED_1/PROT/PTRC1/Tr/General	7SD533 V4.6/Protection/Total/TRIP/3-pole	DK60/LDevice1/XCBR1

Properties

Identification	
Name	GOOSE appli
Description	
Comment	
Type	Siemens GOOSI
Parameter	
Application Id	1
Communication profile	Low priority
Minimum monitoring t	10
Maximum monitoring	2000

Name
Name of the application

Sources

Name	Description
PDIS4	Dist Zone Z4
PDIS5	Dist Zone Z5
PTRC1	Total
Beh	Behaviour
FinTr	AR fin OFF
Health	State
Mod	Mode
Str	Pickup
Tr	TRIP
General	3-pole
PhsA	L1
PhsB	L2
PhsC	L3

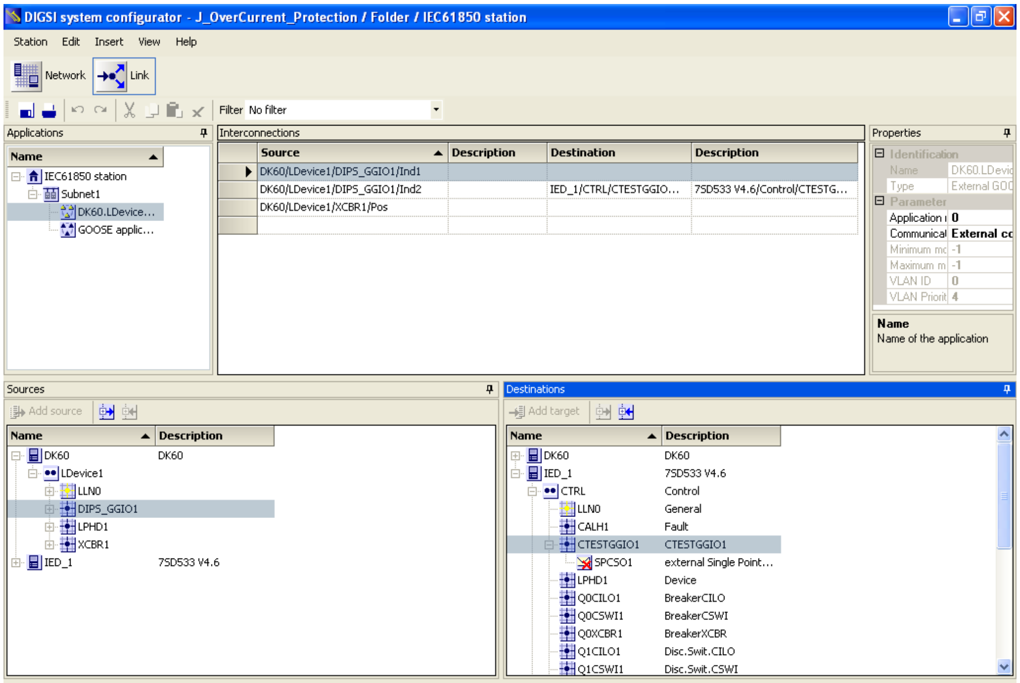
Destinations

Name	Description
DK60	DK60
LDevice1	
LLN0	
DIPS_GGIO1	
LPHD1	
XCBR1	
IED_1	7SD533 V4.6

Figure_A J.10: Trip configuration via GOOSE

As, it is observed, the general trip data information “IED_1.PROT.PTRC1.Tr.General”, is mapped to the “ctVal” of the XCBR1 Logical Node. This is important to do in order to allow the SIPROTEC device to publish a GOOSE message with the required Dataset.

Figure_A J.11 below shows the mapping of ~~a~~ Generic Input Output (GGIO) Data from the Dk60 board (DK60SV1.CID) back to the SIPROTECT device. This was necessary in order to ensure that not only could the Dk60 board with the PIS-10 stack subscribe to a GOOSE message generated by a commercial IED, but also be capable of publishing a GOOSE message that can be subscribed to by a commercial IED for interoperability purposes. Please review section 6.4.3 for further comment on interoperability issues in this project.



Name
IEC61850 station
Subnet1
DK60.LDevice…
GOOSE applic…

Source	Description	Destination	Description
DK60/LDevice1/DIPS_GGIO1/Ind1			
DK60/LDevice1/DIPS_GGIO1/Ind2		IED_1/CTRL/CTESTGGIO…	7SD533 V4.6/Control/CTESTG…
DK60/LDevice1/XCBR1/Pos			

Identification	
Name	DK60.LDevic
Type	External GOC
Parameter	
Application i	0
Communicat	External cc
Minimum mc	-1
Maximum m	-1
VLAN ID	0
VLAN Priorit	4

Name
Name of the application

Name	Description
DK60	DK60
LDevice1	
LLN0	
DIPS_GGIO1	
LPHD1	
XCBR1	
IED_1	7SD533 V4.6

Name	Description
DK60	DK60
IED_1	7SD533 V4.6
CTRL	Control
LLN0	General
CALH1	Fault
CTESTGGIO1	CTESTGGIO1
SPCSO1	external Single Point…
LPHD1	Device
Q0CILO1	BreakerCILO
Q0CSWI1	BreakerCSWI
Q0XCBR1	BreakerXCBR
Q1CILO1	Disc.Swit.CILO
Q1CSWI1	Disc.Swit.CSWI

Figure_A J.11: Mapping of Breaker status back to IED1

J2: CMC 256 configuration set up

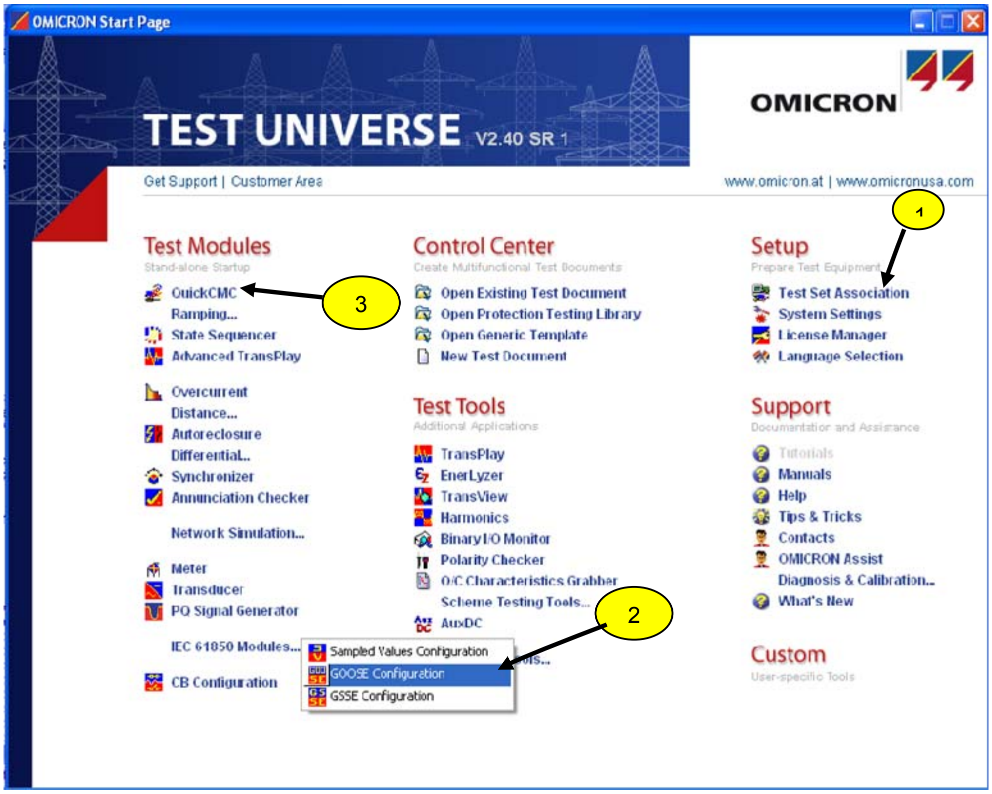
The CMC 256 is a device used by protection engineers to inject test signals into/onto the protection device (DUT – Device Under Test) so as to assess its ability to respond appropriately should such a fault condition occur on a “live” power system.

In this thesis, the OMICRON test injection (CMC 256) is used as a fault emulator to a commercial protective relay. This is done not to test the protection characteristics of the considered relay (System A) but to induce protection behaviour in a second device (System B) by stimulating the generation of a GOOSE message in response to the “fault” event.

The detection of a fault condition by the “System A” generates an alarm. This alarm is published on the network by means of a GOOSE message. The “System B” which is subscribing to the “System A” GOOSE message is playing the role of an “Intelligent Circuit Breaker Controller”. The “System C” serves as the emulated Circuit Breaker.

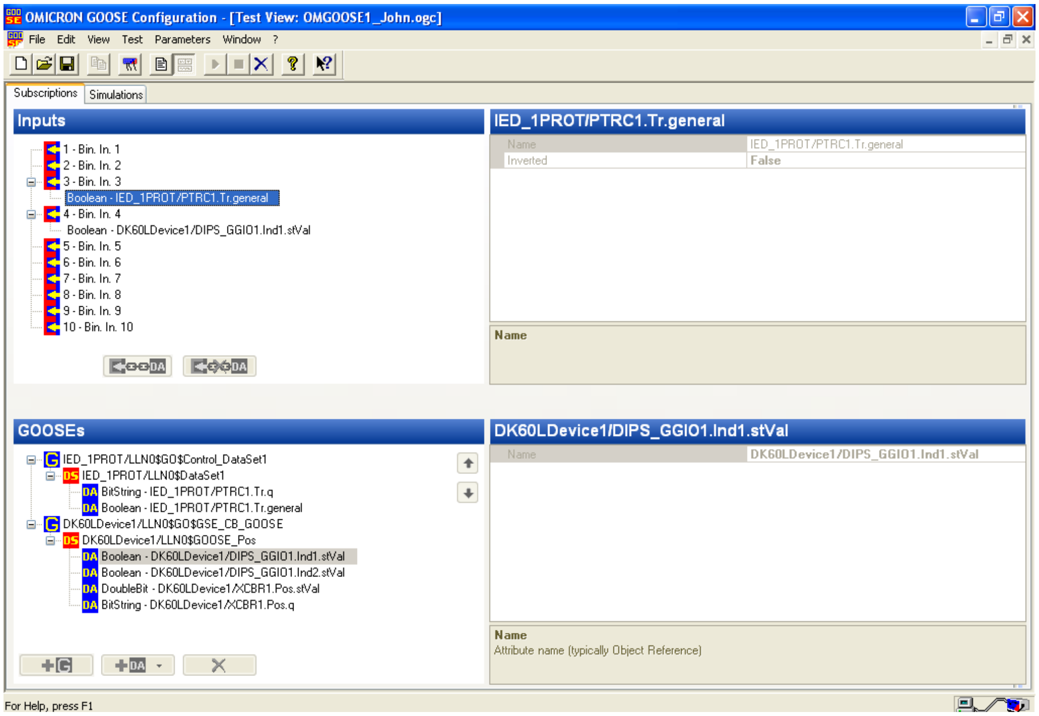
Below, are the basics steps needed to prepare the test set once proper hardwired connections have been made between the SIPROTEC device and the CMC 256.

Figure_A J.12, below shows the Test Universe modules that have been used in order to configure the Test Set.



Figure_A J.12: Test Universe main window

First, with reference to step 1 Figure_A J.12 above, associate the computer to the “Test set” during the set up procedure. The GOOSE configuration module (2) is used to allow the “Test set” to subscribe to specific GOOSE messages on the network. Those GOOSE messages are mapped into dedicated virtual inputs as shown in Figure_A J.13.



OMICRON GOOSE Configuration - [Test View: OMGOOSE1_John.ogc]

Inputs

- 1 - Bin. In. 1
- 2 - Bin. In. 2
- 3 - Bin. In. 3
  - Boolean - IED_1PROT/PTRC1.Tr.general
- 4 - Bin. In. 4
  - Boolean - DK60LDevice1/DIPS_GGIO1.Ind1.stVal
- 5 - Bin. In. 5
- 6 - Bin. In. 6
- 7 - Bin. In. 7
- 8 - Bin. In. 8
- 9 - Bin. In. 9
- 10 - Bin. In. 10

IED_1PROT/PTRC1.Tr.general

Name	IED_1PROT/PTRC1.Tr.general
Inverted	False

Name

GOOSEs

- IED_1PROT/LLN0$GO$Control_DataSet1
  - IED_1PROT/LLN0$DataSet1
    - BitString - IED_1PROT/PTRC1.Tr.q
    - Boolean - IED_1PROT/PTRC1.Tr.general
- DK60LDevice1/LLN0$GO$GSE_CB_GOOSE
  - DK60LDevice1/LLN0$GOOSE_Pos
    - Boolean - DK60LDevice1/DIPS_GGIO1.Ind1.stVal
    - Boolean - DK60LDevice1/DIPS_GGIO1.Ind2.stVal
    - DoubleBit - DK60LDevice1/XCBR1.Pos.stVal
    - BitString - DK60LDevice1/XCBR1.Pos.q

DK60LDevice1/DIPS_GGIO1.Ind1.stVal

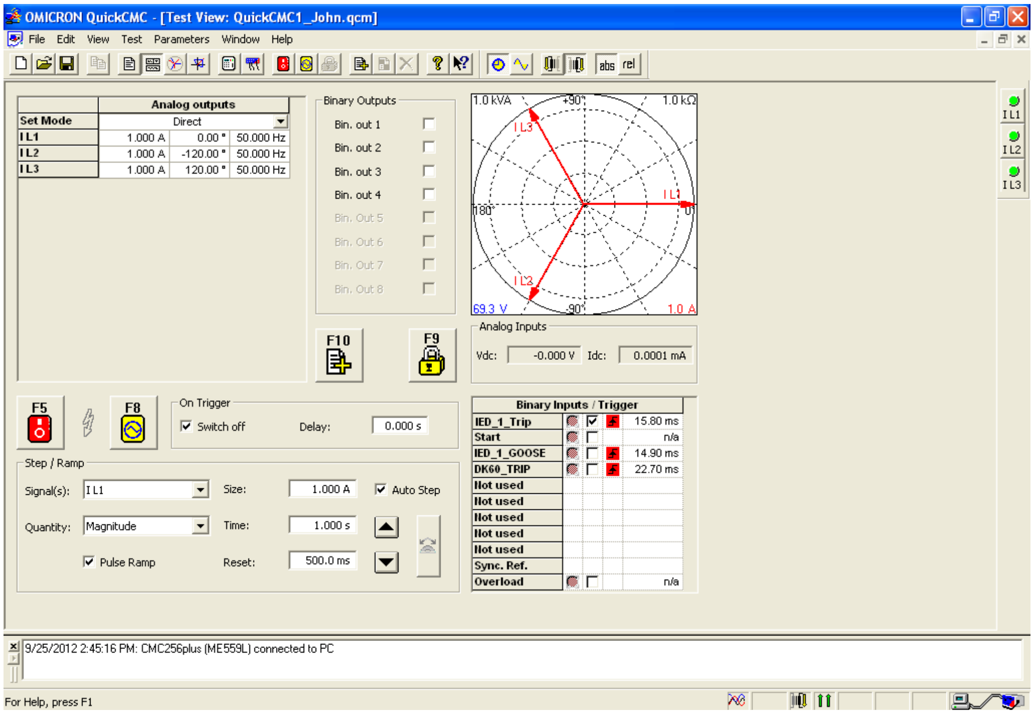
Name	DK60LDevice1/DIPS_GGIO1.Ind1.stVal

Name
Attribute name (typically Object Reference)

For Help, press F1

Figure_A J.13: GOOSE mapping into virtual Input of the CMC 256

After the GOOSE configuration step, open the QuickCMC (step 3 on Figure_A J.12). Figure_A J.14 presents the main view of the QuickCMC.



OMICRON QuickCMC - [Test View: QuickCMC1_John.qcm]

	Analog outputs		
Set Mode	Direct		
I L1	1.000 A	0.00 °	50.000 Hz
I L2	1.000 A	-120.00 °	50.000 Hz
I L3	1.000 A	120.00 °	50.000 Hz

Binary Outputs
- Bin. out 1 □
- Bin. out 2 □
- Bin. out 3 □
- Bin. out 4 □
- Bin. Out 5 □
- Bin. Out 6 □
- Bin. Out 7 □
- Bin. Out 8 □

Analog Inputs
Vdc: -0.000 V Idc: 0.0001 mA

On Trigger
☑ Switch off Delay: 0.000 s

Step / Ramp
Signal(s): I L1 Size: 1.000 A ☑ Auto Step
Quantity: Magnitude Time: 1.000 s
☑ Pulse Ramp Reset: 500.0 ms

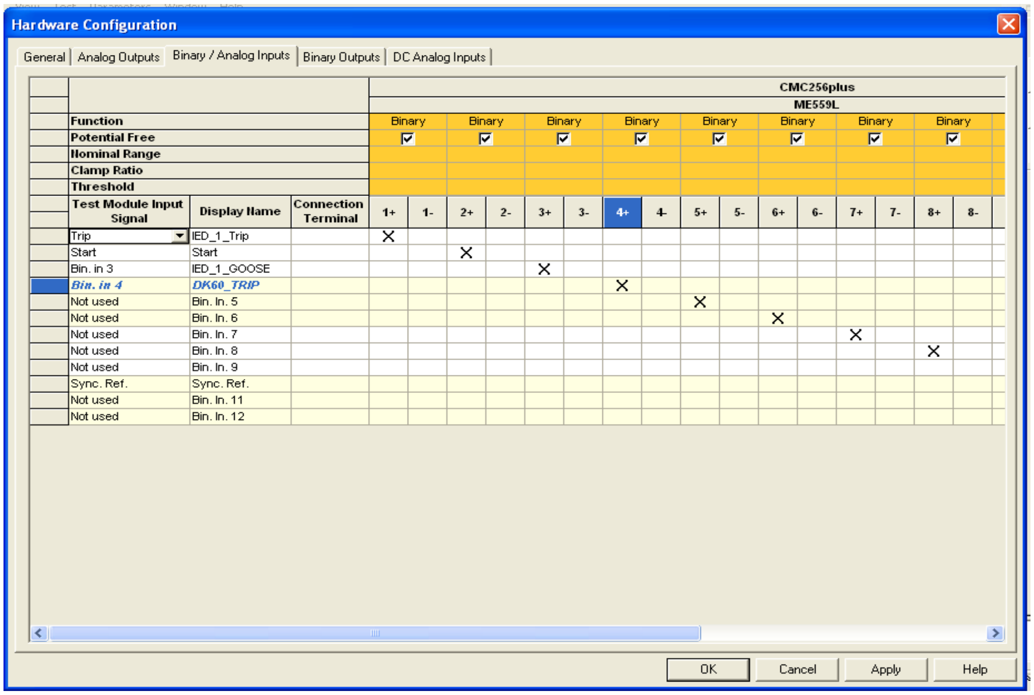
Binary Inputs / Trigger				
IED_1_Trip	●	☑	⚡	15.80 ms
Start	●	□		n/a
IED_1_GOOSE	●	□	⚡	14.90 ms
DK60_TRIP	●	□	⚡	22.70 ms
Not used				
Not used				
Not used				
Not used				
Not used				
Sync. Ref.				
Overload	●	□		n/a

```
9/25/2012 2:45:16 PM: CMC256plus (ME559L) connected to PC
```

For Help, press F1

Figure_A J.14: Quick CMC main view

Configure now the hardware according to the GOOSE message subscription done previously within the “GOOSE Configuration” module. Figure_A J.15 shows the hardware configuration used for this test.



			CMC256plus															
			ME559L															
Function			Binary		Binary		Binary		Binary		Binary		Binary		Binary		Binary	
Potential Free			☑		☑		☑		☑		☑		☑		☑		☑	
Nominal Range																		
Clamp Ratio																		
Threshold																		
Test Module Input Signal	Display Name	Connection Terminal	1+	1-	2+	2-	3+	3-	4+	4-	5+	5-	6+	6-	7+	7-	8+	8-
Trip	IED_1_Trip		X															
Start	Start				X													
Bin. in 3	IED_1_GOOSE						X											
Bin. in 4	*DK60_TRIP*								X									
Not used	Bin. In. 5										X							
Not used	Bin. In. 6												X					
Not used	Bin. In. 7														X			
Not used	Bin. In. 8																X	
Not used	Bin. In. 9																	
Sync. Ref.	Sync. Ref.																	
Not used	Bin. In. 11																	
Not used	Bin. In. 12																	

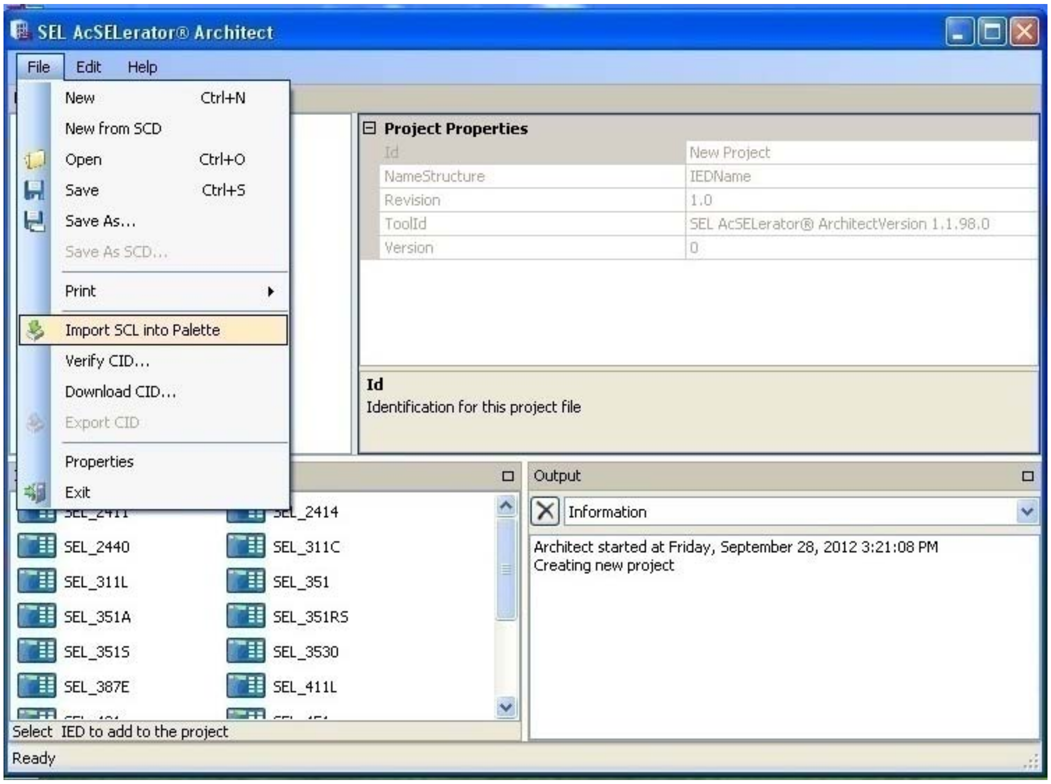
Figure_A J.15: Quick CMC hardware configuration

As can be seen, the GOOSE signal published by the SIPROTEC device (IED_1) which contains the information for a Trip is mapped to the binary input 3. The GOOSE message containing the information of the Circuit Breaker (the adjacent bit) is mapped to the binary input 4.

APPENDIX K

APPENDIX K: SEL file error verification and correction method

To check any SCL file for consistency with “SEL Architect” when it has been previously opened, proceed to the File menu and click on “Import SCL into Palette” as illustrated in Figure_A K.1 below.



Project Properties	
Id	New Project
NameStructure	IEDName
Revision	1.0
ToolId	SEL AcSELerator® ArchitectVersion 1.1.98.0
Version	0

Id
Identification for this project file

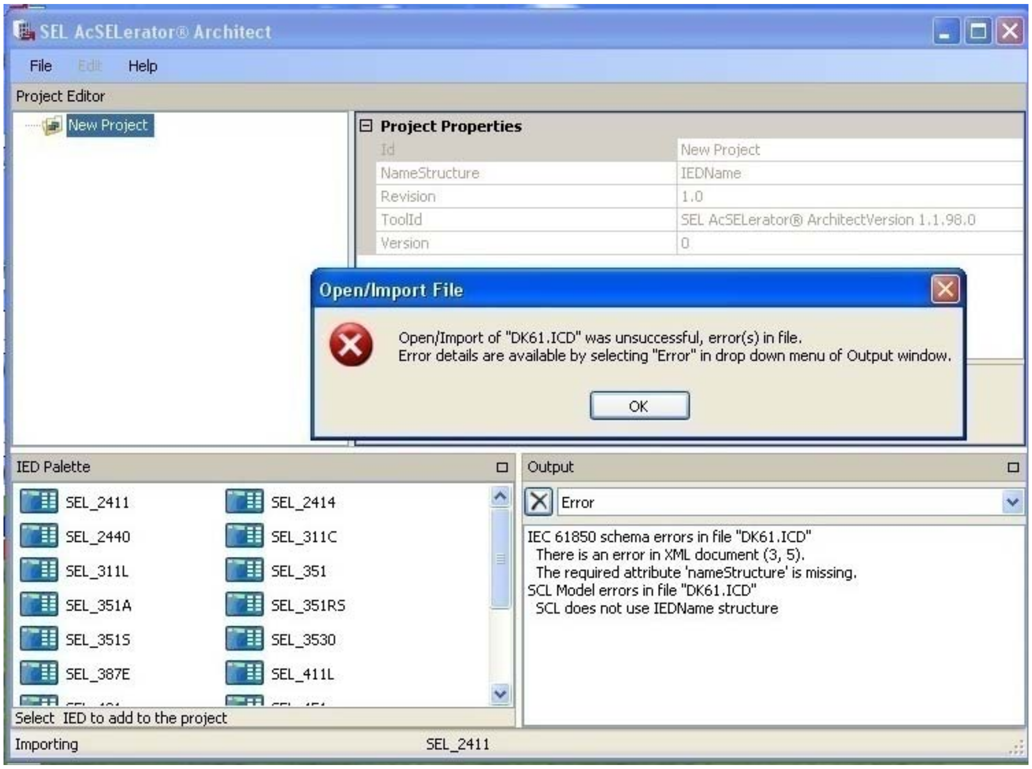
Architect started at Friday, September 28, 2012 3:21:08 PM
Creating new project

Select IED to add to the project

Ready

Figure_A K.1: Import of SCL file for structure verification

If there is an error in the file, you will get the screen presented at Figure_A K.2 with a clear warning.



Project Properties	
Id	New Project
NameStructure	IEDName
Revision	1.0
ToolId	SEL AcSELerator® ArchitectVersion 1.1.98.0
Version	0

Open/Import File

Open/Import of "DK61.ICD" was unsuccessful, error(s) in file.
Error details are available by selecting "Error" in drop down menu of Output window.

Output: Error

```
IEC 61850 schema errors in file "DK61.ICD"
  There is an error in XML document (3, 5).
  The required attribute 'nameStructure' is missing.
SCL Model errors in file "DK61.ICD"
  SCL does not use IEDName structure
```

Figure_A K.2: Message alert for error in SCL importation

Information on what causes the error to occur will be displayed in the output section of the "SEL Architect" main view with precise information on line and column where the error is located and the syntax that causes the problem.

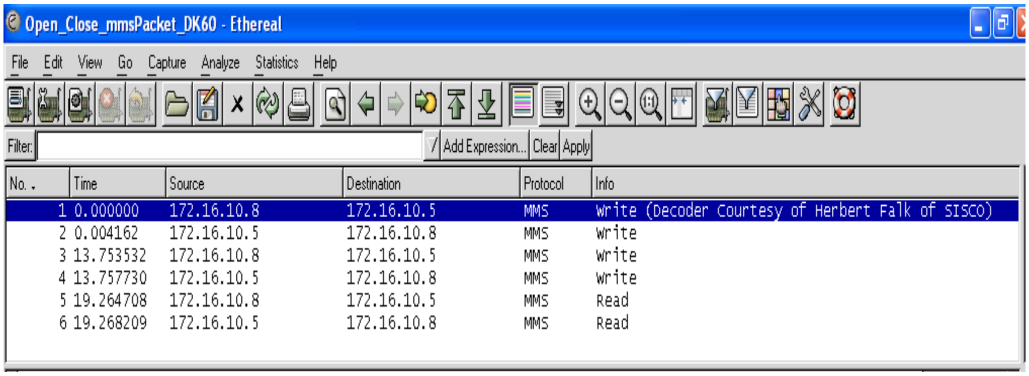
Normally fixing the error requires examination of part six (6) of the IEC 61850 standard and looking for specific information on syntax based on keyword errors provided to you by "SEL Architect". Modify the SCL file manually using XML marker or Notepad. Load the file back into the SEL Architect-XML file checker. Repeat until error-free. Subsequent to this labour-intensive iterative process, loading the file onto another manufacturer's, platform such as DIGSI 4 was normally successful.

APPENDIX L

APPENDIX L: Breaker control based MMS over TCP/IP

The MMS captures in this section have been realized using “MMS-Ethereal” network analyser. Be aware that this is not a detailed study of the MMS message structure like the one performed earlier for the structure of GOOSE message (see section).

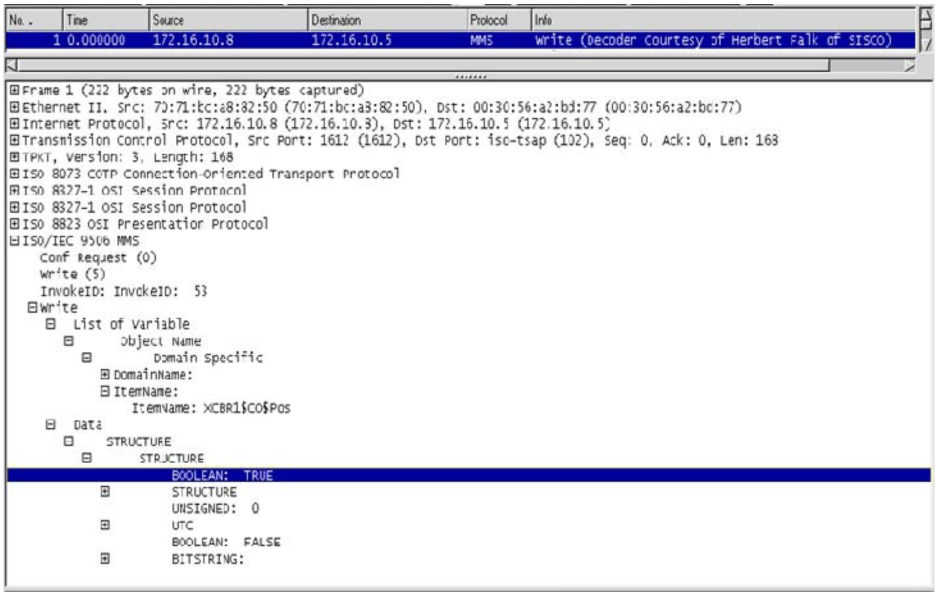
Figure_A L.1 show the main view of **MMS-Ethereal** with a specific set of packet just being captured (you may also use Wireshark, but you will have to do some work around). The rest of pictures that will be shown depict the communication exchange between the universal Client (IEDScout) and the Server object created by the “GA61850.exe” (Cf. Figure 4.34, and Application code on page 273).


Open_Close_mmsPacket_DK60 - Ethereal
File Edit View Go Capture Analyze Statistics Help
Filter:
Add Expression...
Clear
Apply

```
No. .  Time        Source        Destination   Protocol  Info
    1  0.000000    172.16.10.8   172.16.10.5   MMS       Write (Decoder Courtesy of Herbert Falk of SISCO)
    2  0.004162    172.16.10.5   172.16.10.8   MMS       Write
    3  13.753532   172.16.10.8   172.16.10.5   MMS       Write
    4  13.757730   172.16.10.5   172.16.10.8   MMS       Write
    5  19.264708   172.16.10.8   172.16.10.5   MMS       Read
    6  19.268209   172.16.10.5   172.16.10.8   MMS       Read
```

Figure_A L.1: Partial view of captured MMS transaction with MMS-Ethereal

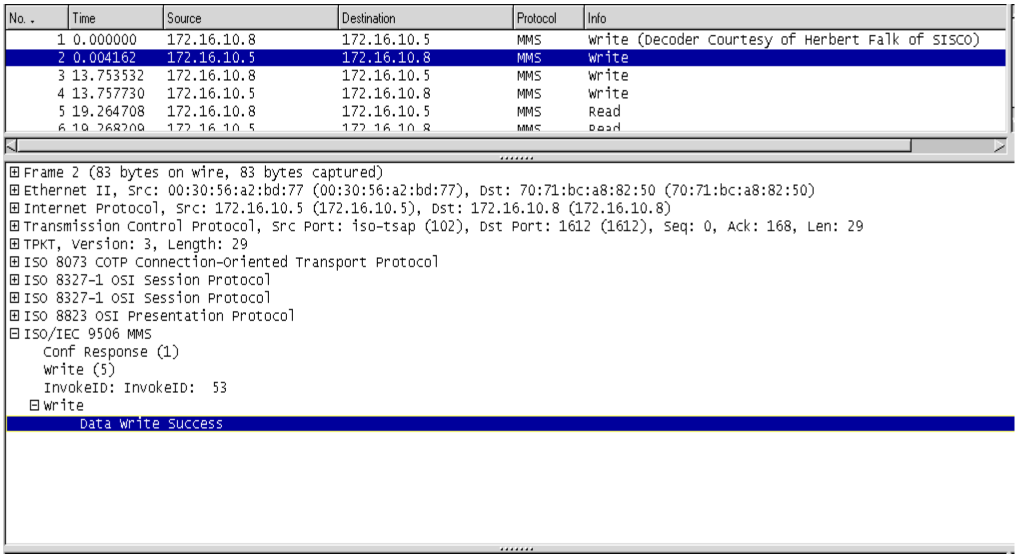
Figure_A L.2 shows the MMS transaction corresponding to a “Close” command (of type Write) sent from IEDScout.



```
No. .  Time        Source        Destination   Protocol  Info
    1  0.000000    172.16.10.8   172.16.10.5   MMS       Write (Decoder Courtesy of Herbert Falk of SISCO)

⊞Frame 1 (222 bytes on wire, 222 bytes captured)
⊞Ethernet II, Src: 70:71:bc:a8:82:50 (70:71:bc:a8:82:50), Dst: 00:30:56:a2:bd:77 (00:30:56:a2:bd:77)
⊞Internet Protocol, Src: 172.16.10.8 (172.16.10.8), Dst: 172.16.10.5 (172.16.10.5)
⊞Transmission Control Protocol, Src Port: 1612 (1612), Dst Port: iso-tsap (102), Seq: 0, Ack: 0, Len: 168
⊞TPKT, Version: 3, Length: 168
⊞ISO 8073 COTP Connection-Oriented Transport Protocol
⊞ISO 8327-1 OSI Session Protocol
⊞ISO 8327-1 OSI Session Protocol
⊞ISO 8823 OSI Presentation Protocol
⊟ISO/IEC 9506 MMS
    Conf Request (0)
    Write (5)
    InvokeID: InvokeID:  53
  ⊟Write
    ⊟  List of Variable
      ⊟       Object Name
        ⊟         Domain Specific
          ⊞DomainName:
          ⊟ItemName:
              ItemName: XCBR1$CO$Pos
    ⊟  Data
      ⊟      STRUCTURE
        ⊟        STRUCTURE
                    BOOLEAN:  TRUE
          ⊞         STRUCTURE
                    UNSIGNED:  0
          ⊞         UTC
                    BOOLEAN:  FALSE
          ⊞         BITSTRING:
```

Figure_A L.2: Transmission of MMS “Close” command

Figure_A L.3, show the acknowledgement message sends back by the Server object of the “GA6185.exe”.

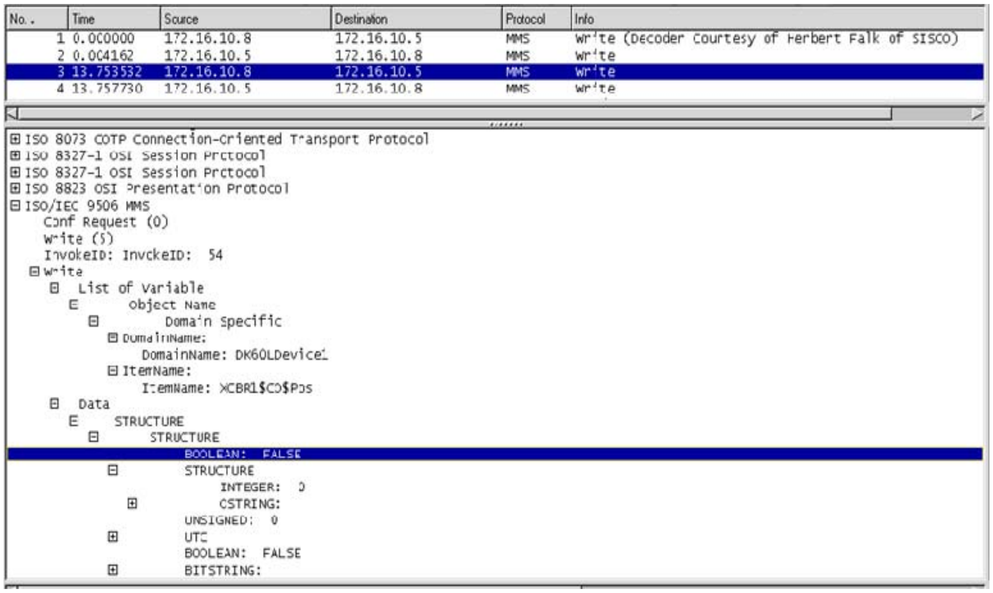


```
No. .  Time         Source        Destination   Protocol  Info
     1 0.000000     172.16.10.8   172.16.10.5   MMS       Write (Decoder Courtesy of Herbert Falk of SISCO)
     2 0.004162     172.16.10.5   172.16.10.8   MMS       Write
     3 13.753532    172.16.10.8   172.16.10.5   MMS       Write
     4 13.757730    172.16.10.5   172.16.10.8   MMS       Write
     5 19.264708    172.16.10.8   172.16.10.5   MMS       Read
     6 19.268209    172.16.10.5   172.16.10.8   MMS       Read
```

```
⊞ Frame 2 (83 bytes on wire, 83 bytes captured)
⊞ Ethernet II, Src: 00:30:56:a2:bd:77 (00:30:56:a2:bd:77), Dst: 70:71:bc:a8:82:50 (70:71:bc:a8:82:50)
⊞ Internet Protocol, Src: 172.16.10.5 (172.16.10.5), Dst: 172.16.10.8 (172.16.10.8)
⊞ Transmission Control Protocol, Src Port: iso-tsap (102), Dst Port: 1612 (1612), Seq: 0, Ack: 168, Len: 29
⊞ TPKT, Version: 3, Length: 29
⊞ ISO 8073 COTP Connection-Oriented Transport Protocol
⊞ ISO 8327-1 OSI Session Protocol
⊞ ISO 8327-1 OSI Session Protocol
⊞ ISO 8823 OSI Presentation Protocol
⊟ ISO/IEC 9506 MMS
    Conf Response (1)
    Write (5)
    InvokeID: InvokeID:  53
  ⊟ Write
        Data Write Success
```

Figure_A L.3: Close command acknowledged

Figure_A L.4 shows the MMS transaction corresponding to an “Open” command (of type Write) sent from IEDScout.

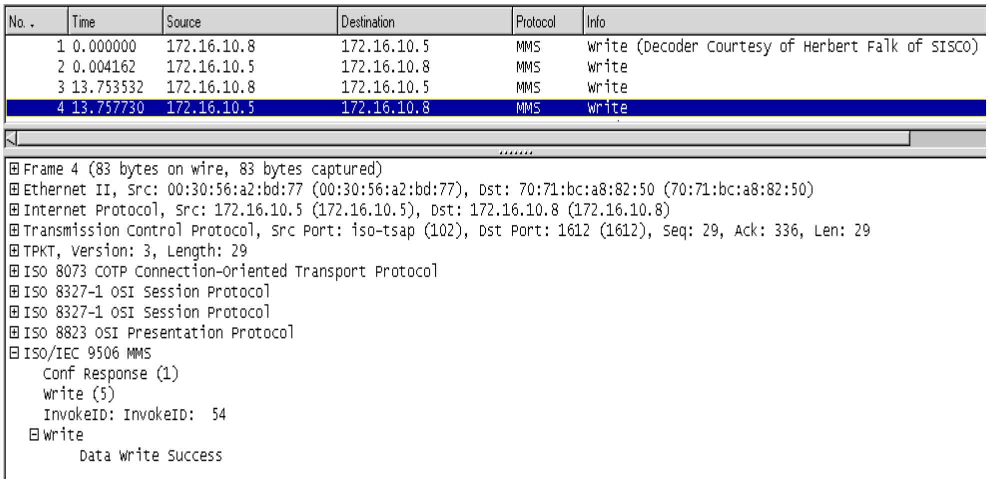


```
No. .  Time         Source        Destination   Protocol  Info
     1 0.000000     172.16.10.8   172.16.10.5   MMS       Write (Decoder Courtesy of Herbert Falk of SISCO)
     2 0.004162     172.16.10.5   172.16.10.8   MMS       Write
     3 13.753532    172.16.10.8   172.16.10.5   MMS       Write
     4 13.757730    172.16.10.5   172.16.10.8   MMS       Write
```

```
⊞ ISO 8073 COTP Connection-Oriented Transport Protocol
⊞ ISO 8327-1 OSI Session Protocol
⊞ ISO 8327-1 OSI Session Protocol
⊞ ISO 8823 OSI Presentation Protocol
⊟ ISO/IEC 9506 MMS
    Conf Request (0)
    Write (5)
    InvokeID: InvokeID:  54
  ⊟ Write
    ⊟  List of Variable
      ⊟      Object Name
        ⊟          Domain Specific
          ⊟ DomainName:
              DomainName: DK60LDevice1
          ⊟ ItemName:
              ItemName: XCBR1$CO$Pos
    ⊟  Data
      ⊟     STRUCTURE
        ⊟        STRUCTURE
                     BOOLEAN:  FALSE
          ⊟          STRUCTURE
                         INTEGER:  0
            ⊞            CSTRING:
                     UNSIGNED:  0
          ⊞          UTC
                     BOOLEAN:  FALSE
          ⊞          BITSTRING:
```

Figure_A L.4: Transmission of MMS “Open” command

Figure_A L.5 shows the acknowledgement message send back by the Server object of the “GA6185.exe”.

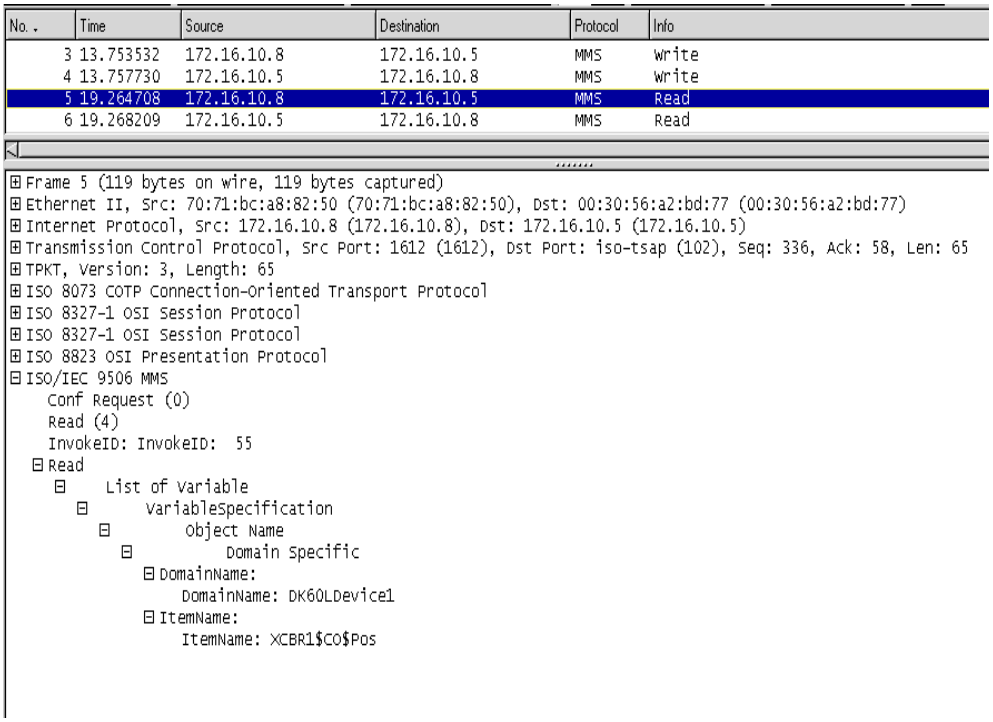


```
No. .  Time          Source          Destination     Protocol  Info
     1 0.000000      172.16.10.8     172.16.10.5     MMS       Write (Decoder Courtesy of Herbert Falk of SISCO)
     2 0.004162      172.16.10.5     172.16.10.8     MMS       Write
     3 13.753532     172.16.10.8     172.16.10.5     MMS       Write
     4 13.757730     172.16.10.5     172.16.10.8     MMS       Write
```

```
⊞Frame 4 (83 bytes on wire, 83 bytes captured)
⊞Ethernet II, Src: 00:30:56:a2:bd:77 (00:30:56:a2:bd:77), Dst: 70:71:bc:a8:82:50 (70:71:bc:a8:82:50)
⊞Internet Protocol, Src: 172.16.10.5 (172.16.10.5), Dst: 172.16.10.8 (172.16.10.8)
⊞Transmission Control Protocol, Src Port: iso-tsap (102), Dst Port: 1612 (1612), Seq: 29, Ack: 336, Len: 29
⊞TPKT, Version: 3, Length: 29
⊞ISO 8073 COTP Connection-Oriented Transport Protocol
⊞ISO 8327-1 OSI Session Protocol
⊞ISO 8327-1 OSI Session Protocol
⊞ISO 8823 OSI Presentation Protocol
⊟ISO/IEC 9506 MMS
   Conf Response (1)
   Write (5)
   InvokeID: InvokeID:  54
  ⊟Write
       Data Write Success
```

Figure_A L.5: Open command acknowledged

Figure_A L.6 shows the MMS transaction corresponding to a “Read” command issue from IEDScout.

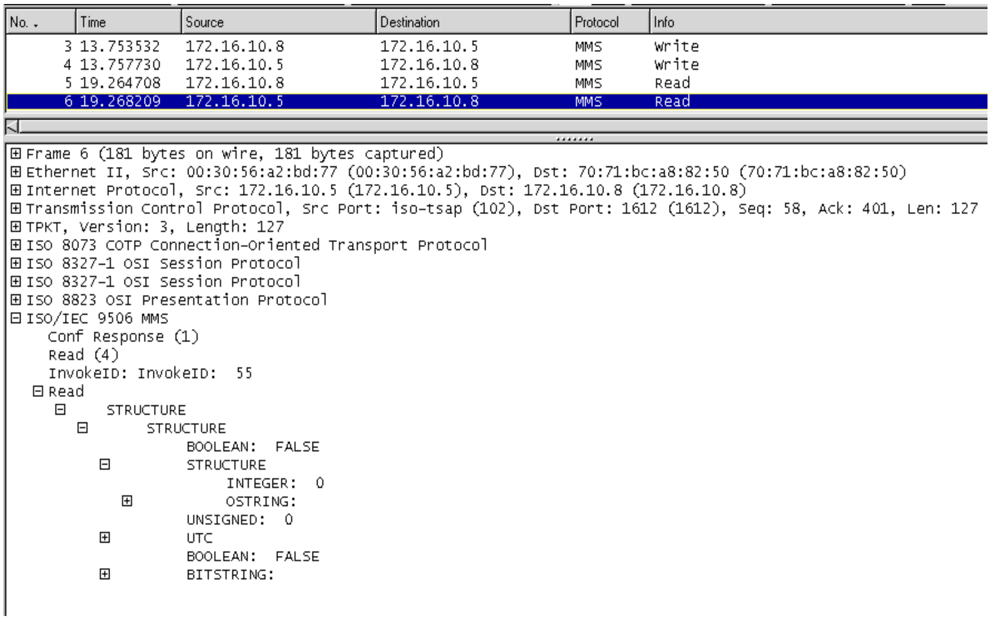


```
No. .  Time          Source          Destination     Protocol  Info
     3 13.753532     172.16.10.8     172.16.10.5     MMS       Write
     4 13.757730     172.16.10.5     172.16.10.8     MMS       Write
     5 19.264708     172.16.10.8     172.16.10.5     MMS       Read
     6 19.268209     172.16.10.5     172.16.10.8     MMS       Read
```

```
⊞Frame 5 (119 bytes on wire, 119 bytes captured)
⊞Ethernet II, Src: 70:71:bc:a8:82:50 (70:71:bc:a8:82:50), Dst: 00:30:56:a2:bd:77 (00:30:56:a2:bd:77)
⊞Internet Protocol, Src: 172.16.10.8 (172.16.10.8), Dst: 172.16.10.5 (172.16.10.5)
⊞Transmission Control Protocol, Src Port: 1612 (1612), Dst Port: iso-tsap (102), Seq: 336, Ack: 58, Len: 65
⊞TPKT, Version: 3, Length: 65
⊞ISO 8073 COTP Connection-Oriented Transport Protocol
⊞ISO 8327-1 OSI Session Protocol
⊞ISO 8327-1 OSI Session Protocol
⊞ISO 8823 OSI Presentation Protocol
⊟ISO/IEC 9506 MMS
   Conf Request (0)
   Read (4)
   InvokeID: InvokeID:  55
  ⊟Read
    ⊟    List of Variable
      ⊟      VariableSpecification
        ⊟        Object Name
          ⊟          Domain Specific
            ⊟DomainName:
                DomainName: DK60LDevice1
            ⊟ItemName:
                ItemName: XCBR1$CO$Pos
```

Figure_A L.6: Transmission of MMS “Read” command

Figure_A L.7 shows the acknowledgement message send back by the Server object of the “GA6185.exe” after a read request.



```
No. .   Time        Source        Destination   Protocol  Info
      3 13.753532   172.16.10.8   172.16.10.5   MMS       Write
      4 13.757730   172.16.10.5   172.16.10.8   MMS       Write
      5 19.264708   172.16.10.8   172.16.10.5   MMS       Read
      6 19.268209   172.16.10.5   172.16.10.8   MMS       Read
```

```
⊞ Frame 6 (181 bytes on wire, 181 bytes captured)
⊞ Ethernet II, Src: 00:30:56:a2:bd:77 (00:30:56:a2:bd:77), Dst: 70:71:bc:a8:82:50 (70:71:bc:a8:82:50)
⊞ Internet Protocol, Src: 172.16.10.5 (172.16.10.5), Dst: 172.16.10.8 (172.16.10.8)
⊞ Transmission Control Protocol, Src Port: iso-tsap (102), Dst Port: 1612 (1612), Seq: 58, Ack: 401, Len: 127
⊞ TPKT, Version: 3, Length: 127
⊞ ISO 8073 COTP Connection-Oriented Transport Protocol
⊞ ISO 8327-1 OSI Session Protocol
⊞ ISO 8327-1 OSI Session Protocol
⊞ ISO 8823 OSI Presentation Protocol
⊟ ISO/IEC 9506 MMS
    Conf Response (1)
    Read (4)
    InvokeID: InvokeID:  55
  ⊟ Read
    ⊟      STRUCTURE
      ⊟        STRUCTURE
                   BOOLEAN:  FALSE
        ⊟          STRUCTURE
                       INTEGER:  0
          ⊞            OSTRING:
                   UNSIGNED:  0
        ⊞          UTC
                   BOOLEAN:  FALSE
        ⊞          BITSTRING:
```

Figure_A L.7: Read command acknowledged

More information on MMS can be found on internet. A recommend link is the following: http://www.nettedautomation.com/qanda/mms/

APPENDIX M

APPENDIX M: DK61 Kit additional description

Figure_A M.1 shows the DK61 kit accessories.

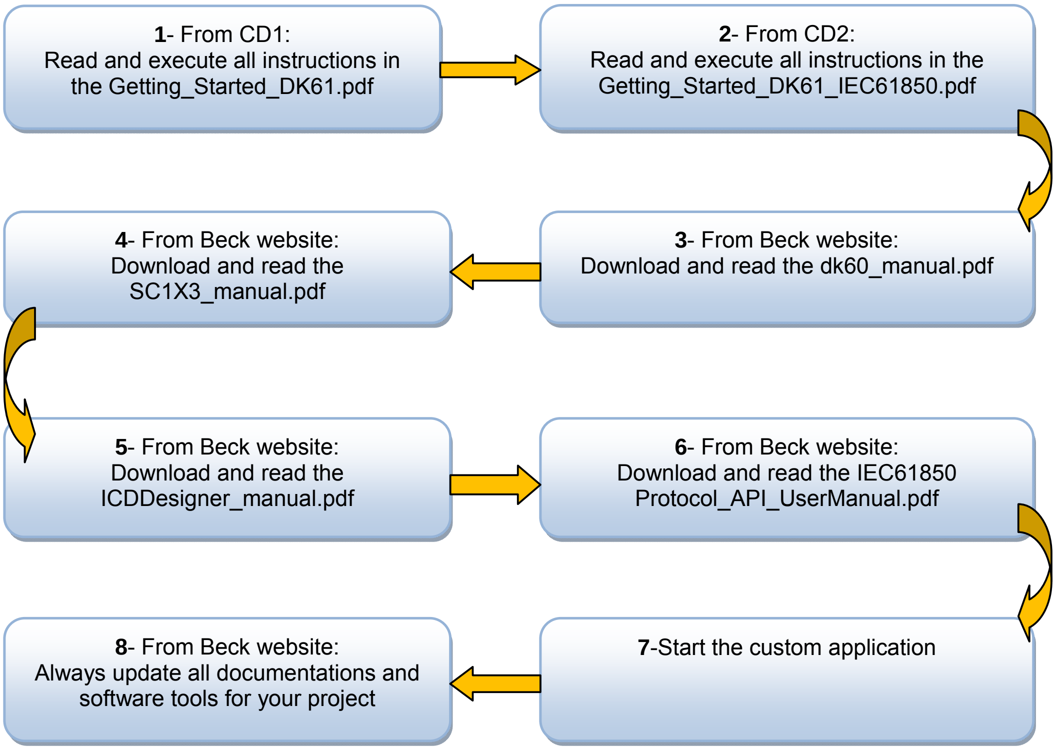
Figure_A M.1: DK61 Development Kit
(Beck IPC-GmBH, 2010)

The DK61 Kit is a composition of the following hardware and software component:

- **DK60*** project board (with adapters for international use);
- Paradigm C/C++ Compiler* (Beck IPC edition for IPC@CHIP®);
- RTOS Remote Debugger and other tools;
- CoDeSys IEC61131-3 SDK for SC123/SC143;
- **IEC 61850 stack*** (Client/Server, GOOSE) and a configuration Tool based on SCL;

The following section provides more information about specific components of the DK61 kit.

There are normally three CDs delivered with the DK61 Kit (CD1: Paradigm C/C++_For Beck IPC, CD2: IEC61850_BECK_Li and CD3: CoDeSys). It is strongly recommended that the latest updates be downloaded from the Beck IPC website. The steps to follow for using the DK61 kit for this project have been chronologically organised in Figure_A M.2:


1- From CDI:
Read and execute all instructions in the Getting_Started_DK61.pdf
2- From CD2:
Read and execute all instructions in the Getting_Started_DK61_IEC61850.pdf
3- From Beck website:
Download and read the dk60_manual.pdf
4- From Beck website:
Download and read the SC1X3_manual.pdf
5- From Beck website:
Download and read the ICDDesigner_manual.pdf
6- From Beck website:
Download and read the IEC61850 Protocol_API_UserManual.pdf
7-Start the custom application
8- From Beck website:
Always update all documentations and software tools for your project

Figure_A M.2: Proposed work flow organisation to start using the DK61 Kit

All documentation related to the DK 61 kit can be freely downloaded from Beck IPC website at http://www.beck-ipc.com/en/download/index.asp.

REFERENCES

ABB AB, 2010. *Live Tank Circuit Breakers*. Application Guide. LUDVIKA (SWEDEN): ABB Marketing & Seles Department.

ABB AG, 2005. *Vacuum circuit-breaker with magnetic actuator mechanism*. User guide. Ratingen-Germany: ABB AG Calor Emag Medium Voltage Products.

ABB- KIRRMANN H., 2004. *Introduction to IEC 61850 Substation communication standard*. User guide. ABBCH-RD.

ABB review, 2010. *Special Report IEC 61850*. Technical journal. Zürich: ABB Group R&D and ABB.

ABB, 2011. *The evolution of substations*. [Online] Available at: http://www.mena.abb.com/ [Accessed 4 October 2011].

Adamiak, M., Baigent, D. and Mackiewicz, R., 2012. *IEC 61850 Communication Networks and Systems In Substations: An Overview for Users*. [Online] GE Digital Energy & SISCO Available at: http://www.gedigitalenergy.com/multilin/journals/issues/spring09/iec61850.pdf [Accessed 28 August 2012].

Adaptive Digital, 2012. *Adaptive Digital Protocol Stack Algorithm Index*. [Online] Available at: http://www.adaptivedigital.com/product/protocol_stacks/protocol_stk_index.htm [Accessed 21 October 2012].

Admiak, M. and Baigent, D., 2000. "Practical consideration in application of UCA GOOSE". In Gb, ed. *Georgia Tech Relay Conference*. Prague, 2000. GE. pp.1-2.

Aggarwal, S., 2011. "IEC 61850 prototype design". In *Innovative Smart Grid Technologies (ISGT)*. Hilton Anaheim, CA, 2011. IEEE PES. p.1.

Aggarwal, S., 2011. "IEC 61850 prototype design". In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*. Hilton Anaheim, CA, 2011. IEEE. pp.1 - 4.

Ali, I. and Thomas, M.S., 2011. "GOOSE based protection scheme implementation & testing in laboratory". In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*. Hilton Anaheim, CA, 2011. IEEE. pp.1 - 7.

Ankatross, J.O., 2007. *IEC 61850 testing*. [Online] OMICRON electronics GmbH: OMICRON Available at: http://www.omicron.at/en/products/pro/communication-protocols/ [Accessed 20 july 2010].

Apostolov, A., 2010. *IEC 61850 Fundamentals, Application and Benefits*. Training Course. Cape Town: CPUT.

Apostolov, A.P., 2012. "Impact of IEC 61850 on the engineering of protection schemes". In OMICRON electronics, U., ed. *Developments in Power Systems Protection, 2012. DPSP 2012. 11th International Conference*. Birmingham, UK, 2012. IEEE. pp.1 - 6.

Apostolov, A. and Vandiver, B., 2010. "IEC 61850 process bus - principles, applications and benefits". In *Protective Relay Engineers, 2010 63rd Annual Conference*. College Station, TX, 2010. IEEE. pp.1 - 6.

Apostolov, A. and Vandiver, B., 2011. "IEC 61850 GOOSE applications to distribution protection schemes". In *Protective Relay Engineers, 2011 64th Annual Conference*. College Station, TX, 2011. IEEE. pp.178 - 184.

Appostolov, A., 2011. *IEC 61850 Systems and their Components*. Seminar Note. CAPE TOWN: CPUT CPUT.

AREVA T&D, 2002. Relay Technology. In F. Espace, ed. *Network Protection & Automation Guide*. Fisrt Edition ed. Paris: AREVA T&D. pp.99-110.

AREVA, 2005. Fundamentals of Protection Practice. In F. Espace, ed. *Network Protection and Automation Guide*. First Edition ed. Paris, France: AREVA T&D. Ch. 2. p.7.

Baigent, D. and Lebenhaft, E., 1991. "Microprocessor-based protection relays: Design and application examples". In *Petroleum and Chemical Industry Conference, 1991, Record of Conference Papers., Industry Applications Society 38th Annual*. Toronto, Ont, 1991. IEEE. pp.237- 244.

Beck IPC-GmBH, 2010. *beck-ipc*. [Online] Available at: http://www.beck-ipc.com.

BECK-IPC, 2011. *Multitasking with @Chip-RTOS*. [Online] (07.11.2011) Available at: http://www.beck-ipc.com/files/api/scxxx/multitasking.htm [Accessed 15 December 2011].

Bonetti, A. and Douib, R., 2010. "Transfer time measurement for protection relay applications with the IEC 61850 standard". In *Electrical Insulation (ISEI), Conference Record of the 2010 IEEE International Symposium*. San Diego, CA, 2010. IEEE. pp.1 - 5.

Brand, K. and Wimmer, W., 2003. *Modeling interoperable protection and control devices for substation automation according to IEC 61850*. Technichal repport. Bruggerstr: Utility Automation Systems ABB Switzerland Ltd.

CESI RICERCA, 2006. *Valutazione delle tempistche associate ai messaggi di tipo GOOSE nell'ambito del protocol IEC 61850*. Project repport. Milano: TTD Tecnologie T&D.

Chaturvedi, M., 2002. "Substation IED communications". In *Power Engineering Society Winter Meeting*., 2002. IEEE. p.509 Vol.1.

Dick, A.J., 1996. "Review of communications standards for distribution automation applications". In *Methods of Substation Automation, Half Day Colloquium*. Manchester, 1996. IEEE. pp.1/1 - 1/5.

Digital Bond, 2010. *IEC 61850*. [Online] Available at: http://www.digitalbond.com/wiki/index.php/IEC_61850 [Accessed 26 July 2010].

Edward, A.L., 2002. Embedded Software. In M. Zelkowitz, ed. *Advances in Computers*. 56th ed. London: Academic Press. p.1.

ESA Board, 1991. ESA PSS-05-0 Issue 2 *Software Engineering Standard Software Standardisation and Control*. Standars. PARIS: ESA Board for Software Standardisation and Control (BSSC).

Goraj, M. and Herrmann, J., 2007. *Experience in IEC 61850: And possible improvements of SCL Language*. [Online] sisconet Available at: http://www.sisconet.com/downloads/GE_E_2007.pdf [Accessed 20 October 2012].

Hammer, E. and Sivertsen, E., 2008. *Analysis and implementation of the IEC 61850 standard*. Thesis. Technical University of Denmark.

Hamrén, R., 2007. *Using IEC 61850 for remote disturbance analysis*. Master Thesis. Östersund, Sweden: Mälardalen University.

Hou, D. and Dolezilek, D., 2012. *IEC 61850 – What It Can and Cannot Offer to Traditional Protection Schemes*. [Online] Schweitzer Engineering Laboratories, Inc Available at: http://www.selinc.com/WorkArea/DownloadAsset.aspx?id=3546 [Accessed 20 October 2012].

IEC 61850-1, 2003. *Communication networks and systems in substations – Part 1: Introduction and overview*. Standard. IEC.

IEC 61850, 2010. *IEC Main web page*. [Online] Available at: http://www.iec.ch/.

IEC 61850-2, 2003. *Communication networks and systems- Part 2: Glossary*. Standard. IEC.

IEC 61850-2, 2004. *Communication networks and systems in substations – Part 9-2:Specific Communication Service Mapping (SCSM) –Sampled values over ISO/IEC 8802-3*. Standard. IEC.

IEC 61850-5, 2003. *Communication networks and systems in substations- Part 5: Communication requirements for functions and device models*. Standard. IEC.

IEC 61850-6, 2004. *Communication networks and systems in substations – Part 6: Configuration description language for communication in electrical substations related to IEDs*. Standard. IEC.

IEC 61850-7-1, 2003. *Communication networks and systems in substations – Part 7-1: Basic communication structure for substation and feeder equipment – Principles and models*. Standard. IEC.

IEC 61850-7-2, 2003. *Communication networks and systems in substations – Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI)*. Standard. IEC.

IEC 61850-7-4, 2003. *Communication networks and systems in substations – Part 7-4: Basic communication structure for substation and feeder equipment – Compatible logical node classes and data classes*. Standard. IEC.

IEC 61850-7-500-DTR_Ed1, 2011. *Basic information and communication structure - Use of logical nodes for modelling applications and related concepts and guidelines for Substations*. Standard. IEC.

IEC 61850-8-1, 2004. *Communication networks and systems in substations – Part 8-1: Specific Communication Service Mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*. Standard. IEC.

IEC 61850-9-2, 2004. *Communication networks and systems in substations – Part 9-2: Specific Communication Service Mapping (SCSM) – Sampled values over ISO/IEC 8802-3*. Standard. IEC.

IEEE Canadian Region, 2000. Links With the World. In W.H. Prevey, ed. *Electricity: The Magic Medium*. 2nd ed. IEEE Canadian Region.

In, E. et al., 2011. "Design of one chip communication stack processor and MMS communication stack library based on IEC 61850". In *Networked Embedded Systems for Enterprise Applications (NESEA), 2011 IEEE 2nd International Conference on*. Fremantle, WA, 2011. IEEE. pp.1 - 6.

ITU-T, 2002. ISO/IEC 8825-1; X.690-027 *Abstract Syntax Notation One (ASN.1): Specification of Basic Encoding Rules (BER)*. Standard. Geneva: International Telecommunication Union.

Kim, G.-S. and Lee, H.-H., 2005. "A study on IEC 61850 based communication for intelligent electronic devices". In *Science and Technology, 2005. KORUS 2005. Proceedings. The 9th Russian-Korean International Symposium*. Ulsan, 2005. IEEE. pp.765 - 770.

Kostic, T. and Frei, C., 2007. "Modelling and using IEC 61850-7-2 (ACSI) as an API". In *Power Tech, 2007 IEEE Lausanne*. Lausanne, 2007. pp.713 - 719.

Lundqvist, B., 2003. *100 years of relay protection, the Swedish ABB relay history*. White paper. Sweden: ABB Automation Products.

Makadam, M.A., 2008. *Case study and evaluation of an IEC 61850*. Unpublished Project repport. Cape Town: Cape Peninsula University of Technology.

Mansour, M.M. and Swift, G.W., 1986. Design and testing of a multi-microprocessor travelling wave relay. *Power Delivery, IEEE Transactions on*, 1(4), pp.74- 82.

Martin, P. and Nguyen, V.T., 2004. SEG001 *Système ALCID*. Standard implementation guideline. Hydro Québec.

McGhee, J. and Goraj, M., 2010. "Smart High Voltage Substation based on IEC 61850 Process Bus and IEEE 1588 Time Synchronization". In *International Conference on Smart Grid Communications (SmartGridComm)*. Gaithersburg, MD, 2010. IEEE. pp.490-94.

Montignies, P., Angays, P. and Guise, L., 2011a. IEC 61850 in the oil and gas industries. *IEEE Industry Applications Magazine*, 17(1), pp.36 - 46.

Montignies, P., Angays, P. and Guise, L., 2011. IEC 61850 in the oil and gas industries. *IEEE Industry Applications Magazine*, p.37.

Nasrallah, E., Brikci, F. and Perron, S., 2007. *Zensol*. [Online] Available at: http://www.zensol.com/en/make-break-contacts-in-power-circuit-breakers [Accessed 20 January 2012].

National Instruments, 2011. *NI Developer Zone: Differences Between Multithreading and Multitasking for Programmers*. [Online] Available at: http://www.ni.com/white-paper/6424/en [Accessed 15 March 2012].

Nelson, J., 2005. *Automation solution: Why automate*. White paper. General Electric Company.

NETGEAR, 2012. *Netgear corporation website*. [Online] Available at: http://support.netgear.com/app/answers/detail/a_id/1070/~/defining-the-terms-driver,-firmware,-hardware,-software,-and-utility.

Nguyen-Dinh, N., Kim, G.-S. and Lee, H.-H., 2007. "A study on GOOSE communication based on IEC 61850 using MMS ease lite". In *Control, Automation and Systems, 2007. ICCAS '07. International Conference*. Seoul, 2007. IEEE. pp.1873 - 1877.

Nordel, D.E., 2005. *Substation Communication History and Pratice*. IT course. Minneapolis: Xcel Energy.

Nordell, D.E., 2003. *Substation Communication History and Practice*. [Online] Michigan Technological University: Michigan Tech. University Available at: http://www.ece.mtu.edu/faculty/bamork/EE5223/proceedi.pdf [Accessed 11 July 2012].

Nuclear Power Training, 2012. *Integrated Publishing*. [Online] Available at: http://nuclearpowertraining.tpub.com/h1011v1/css/h1011v1_110.htm [Accessed 08 October 2012].

Olovsson, H.-E. and Lejdeby, S.-A., 2008. *Substation evolution: Substation design in the 1900s and modern substations today*. White paper. ABB Power Systems.

OMG UML, 2012. *UML 2.2 Diagrams Overview*. [Online] Available at: http://www.uml-diagrams.org/uml-22-diagrams.html [Accessed 15 January 2012].

OMICRON-IEDScout, 2012. *OMICRON IEDScout 2.1*. [Online] Available at: http://omicron-iedscout.software.informer.com/2.1/ [Accessed 10 October 2012].

Oracle, 2012. *The Java Tutorials*. [Online] Available at: http://docs.oracle.com/javase/tutorial/java/concepts/index.html.

OSI Model-org, 2012. *OSI Model*. [Online] Available at: http://www.osimodel.org/ [Accessed 15 January 2012].

Ozansoy, C.R., 2006. *Design & Implementation of a Universal Communications Processor for Substation Integration, Automation and Protection*. Unpublished Phd Thesis. Australia: Victoria University.

Ozansoy, C.R., Zeyegh, A. and Akhtar, K., 2009. The Application-View Model of the International Standard IEC 61850. *Power Delivery, IEEE Transactions* , 24(3), pp.1132 - 1139.

Paradigm, 2010. *PARADIGM C++ PROFESSIONAL IDE*. [Online] Available at: http://www.devtools.com/default.htm.

Park, J. et al., 2012. "IEC 61850 Standard Based MMS Communication Stack Design Using OOP". In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. Fukuoka, 2012. IEEE. pp.329 - 332.

Proudfoot, D., 2002. *UCA and 61850 For DUMMIES*. Technichal repport. Siemens Power Transmission & Distribution.

Richards, S., 2006. "Building Very High Reliability into the Design and Manufacture of Relays". In T&D, A., ed. *Advances in Power Protection: Learning the Lessons from Major Disturbances. The IEE Seminar on (Ref. No. 2006/11426).*, 2006. IEEE Conference Publications. pp.94-104.

Ruggedcom Russia, 2010. [Online] Available at: http://www.ruggedcom.ru.

Sahai, A. and Pandya, D.J., 2011. "A Novel Design and Development of a Microprocessor-based Current Protection Relay". In *Recent Advancements in Electrical, Electronics and Control Engineering (ICONRAEeCE), 2011 International Conference on*. Sivakasi, 2011. IEEE. pp.309- 313.

Schwarz, K., 2012. *IEC 61850 in the U.S. – A Personal View of IEC 61850*. [Online] NettedAutomation Available at: http://blog.iec61850.com/2012/01/iec-61850-in-us-personal-view-of-iec.html [Accessed 15 November 2012].

Schwarz, K., 2012. *IEC 61850-5 Edition 2 FDIS Published for Ballot*. [Online] NettedAutomation Available at: http://blog.iec61850.com/2012/10/iec-61850-5-edition-2-fdis-published.html [Accessed 15 November 2012].

Shujian, Z., Ming, S. and Gaofei, G., 2011. "The research based on IEC61850 intelligent terminal of substation". In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference*. Xi'an, 2011. IEEE. pp.304 - 307.

Sidhu, T.S., Injeti, S., Kanabar, M.G. and Parikh, P.P., 2010. "Packet scheduling of GOOSE messages in IEC 61850 based substation intelligent electronic devices (IEDs)". In *Power and Energy Society General Meeting, 2010 IEEE*. Minneapolis, MN, 2010. IEEE. pp.1 - 8.

Sidhu, T.S., Kanabar, M.G. and Dadash Zadeh, M.R., 2011a. *IEC 61850-9-2 Based Process Bus*. [Online] University of Western Ontario Available at: http://www.pacw.org/issue/december_2010_issue/process_bus.html [Accessed 27 October 2012].

Sidhu, T., Kanabar, M. and Parikh, P., 2011b. "Configuration and performance testing of IEC 61850 GOOSE". In *Advanced Power System Automation and Protection (APAP), 2011 International Conference*. Beijing IEEE. pp.1384 - 1389.

Siemens AG, 2010. "Aspects on IEC 61850 Edition 2.0 & Current Activities". In *IEEE/PES Transmission & Distribution- Latin America, 9th IEEE/IAS International Conference on Industry Applications - INDUSCON*. São Paulo, 2010. IEEE/PES. pp.1-23.

Siemens PLM Software, 2010. *Role of PLM in the software lifecycle*. White paper. Granite Parkway (USA): Siemens PLM Software Siemens PLM Software.

Siemens, 2009. *Ethernet & IEC 61850: Start Up*. [Online] Siemens Available at: http://siemens.siprotec.de/download_neu/software/DIGSI_4/Manual/IEC61850_STARTUP_A3_EN.pdf [Accessed 15 September 2010].

SISCO, 1996. *MMS and ASN.1 Encodings*. Project repport. Sterling Heights: Systems Integration Specialists Company (SISCO).

SourceMaking, 2012. *History of UML: Methods and Notations*. [Online] Available at: http://sourcemaking.com/uml/basic-principles-and-background/history-of-uml-methods-and-notations [Accessed 15 January 2012].

Steinhauser, F., Schossig, T., Klien, A. and Geiger, S., 2010. *Performance Measurements for IEC 61850 IEDs and Systems*. [Online] OMICRON electronics Available at: http://www.omicron.at/fileadmin/user_upload/files/pdf/en/2010-PotM-Performance-Measurement-IEC61850.pdf [Accessed 20 July 2012].

SUITTIO, A., 2010. *TESTING IEC 61850 IN MULTI-VENDOR SUBSTATION AUTOMATION SYSTEM*. Master Thesis. TAMPERE UNIVERSITY OF TECHNOLOGY.

SystemCorp_a, 2011. *Getting Started IPC@CHIP Embedded Web Controller Family IEC 61850 Basics*. [Online] Systemcorp Embedded Technology (1.01) Available at: http://www.systemcorp.com.au/images/stories/manuals/Getting_Started_DK61_IEC61850.pdf [Accessed 08 May 2011].

SystemCorp_b, 2010. *PIS10 API User Manual*. [Online] Available at: http://www.systemcorp.com.au/news/42-iec-61850-protocol-stack-pis-10-.html [Accessed 15 January 2011].

SystemCorp_c, 2012. *New IEC 61850 Beck DK-61 Library Release*. [Online] Systemcorp Available at: https://www.systemcorp.com.au/news/83-iec-6185-beck-dk-61-library-release.html [Accessed 15 August 2012].

Ting, W., Xuejin, G., Ping, W. and Juan, L., 2011. "Research of Relaying Protection System of Digital Substation". In *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference*. Shenzhen, Guangdong, 2011. IEEE. pp.684 - 688.

TOSHIBA, 2012. *TOSHIBA: Leading Innovation*. [Online] Available at: http://www.toshiba.co.jp/sis/en/tands/switch/gis4.htm [Accessed 19 May 2012].

US-Department of Labor, 2010. *Substation Equipment and function*. [Online] Available at: http://www.osha.gov/SLTC/etools/electric_power/illustrated_glossary/substation.html [Accessed 22 February 2010].

Uzair, M., 2008. *COMMUNICATION METHODS (PROTOCOLS, FORMAT & LANGUAGE) FOR THE SUBSTATION AUTOMATION & CONTROL*. [Online] Western Engineering Available at: http://www.eng.uwo.ca/people/tsidhu/Documents/project%20report%20Uzair.pdf [Accessed 15 July 2012].

Walter, S., 2007. Protection history: the Start of Protection. *PAC World Magazine*, AUTUMN. p.69.

Walter, S., 2009. Protection History: Generator Protection, The Beginning. *PAC World Magazine*, SPRING. p.71.

Wolf, G., 2009b. *Penton Media. n.d Digital Substation: Evolution or Revolution*. [Online] Available at: http://tdworld.com/distribution_management_systems/upgrading-electrical-substations-digital-200901/index.html [Accessed 15 september 2009].

Wolf, G., 2009a. *Penton Media. n.d Digital Substation: Evolution or Revolution*. [Online] Available at: http://tdworld.com/distribution_management_systems/upgrading-electrical-substations-digital-200901/index1.html [Accessed 15 september 2009].

Yang, C., Vyatkin, V., Nair, N.-K.C. and Chouinard, J., 2011. "Programmable Logic for IEC 61850 Logical Nodes by means of IEC 61499". In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*. Melbourne, VIC, 2011. IEEE. pp.2717 - 2723.

Yongli, Z., Dewen, W., Yan, W. and Wenquing, Z., 2009. "Study on interoperable exchange of IEC 61850 data model". In *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference*. Xi'an, 2009. IEEE. pp.2724 - 2728.

Zhang, J. and Gunter, C.A., 2009. *IEC 61850- Communication Networks and Systems in Substation: An Overview of Computer Science*. IT course. Urbana-Champaign: University of Illinois.

Zhang, J., Guo, C., Xu, L. and Cao, Y., 2008. "The design of an Object-Oriented Embedded Platform for Substation Data Integration". In *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*. Pittsburgh, PA, 2008. IEEE. pp.1 - 6.

Zhang, L.X. and Nair, N.-K., 2008. "Testing protective relays in IEC 61850 framework". In *Power Engineering Conference, 2008. AUPEC '08.*. Sydney, NSW, 2008. Australasian Universities. pp.1 - 6.

Zhu, T., Xie, X., Zhu, D. and Cui, W., 2006. "Generating Detailed Software Models of Microprocessor-Based Relays". In *Power System Technology, 2006. PowerCon 2006. International Conference on*. Chongqing, 2006. IEEE. pp.1 - 6.

Reference evidence

Related to Figure 2.1: Publication rate for “IEC 61850 Software” topic, page 13 of this thesis

Apostolov, A., Brunner, C. & Clinard, K. 2003, "Use of IEC 61850 object models for power system quality/security data exchange", *Quality and Security of Electric Power Delivery Systems, 2003. CIGRE/PES 2003. CIGRE/IEEE PES International Symposium*, pp. 155.

Apostolov, A. & Ingleson, J. 2004, "Verification of models in protection related analysis programs", *Protective Relay Engineers, 2004 57th Annual Conference for*, pp. 339.

Barron, D.A. & Holliday, P. 2010, "Implementation of IEC 61850 process bus. A Utility view on design, installation, testing and commissioning, and lifetime issues.” *Developments in Power System Protection (DPSP 2010). Managing the Change, 10th IET International Conference on*, pp. 1.

Bin Duan, Nian Liu and Shenglong Huang 2006, "Study on Substation Control Interlocking Combined with PKI/PMI Based Access Security Method", *Power System Technology, 2006. PowerCon 2006. International Conference on*, pp. 1.

Bose, S., Liu, Y., Bahei-Eldin, K., de Bedout, J. and Adamiak, M. 2007, "Tieline controls in microgrid applications", *Bulk Power System Dynamics and Control - VII. Revitalizing Operational Reliability, 2007 iREP Symposium*, pp. 1.

Brédillet, P., Lambert, E. & Schultz, E. 2010, "CIM, 61850, COSEM Standards Used in a Model Driven Integration Approach to Build the Smart Grid Service Oriented Architecture", *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pp. 467.

Brent, D. & Self, H. 2006, ""Applications and Advantages for Protection schemes using IEC 61850 Standard"", Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources, 2006. PS '06, pp. 63.

Chen Ailin, Dai Zemei, Ding Jie, Huang Haifeng, Wang Yan & He Ming-yi 2010, "Application of IEC 61850 proxy in seamless communication between digital substation and control centre", *Electricity Distribution (CICED), 2010 China International Conference on*, pp. 1.

Davidson, E.M., McArthur, S., Dolan, M.J. & McDonald, J.R. 2009, "Exploiting intelligent systems techniques within an autonomous regional active network management system", *Power & Energy Society General Meeting, 2009. PES '09. IEEE*, pp. 1.

Dewen Wang, Yongli Zhu, Peipei Liu, Jiancai Huang & Wenqing Zhao 2008, "Research on distributed transmission of power telecontrol information based on ACSI/MMS", *Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on*, pp. 670.

Ferrari, P., Flammini, A., Rinaldi, S. & Prytz, G. 2011, "Applying PTP-to-SNTP time-gateway to IEC61850 systems", *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pp. 1.

Ferrari, P., Flammini, A., Rinaldi, S. & Prytz, G. 2011, "Time synchronization concerns in substation automation system", *Applied Measurements for Power Systems (AMPS), 2011 IEEE International Workshop on*, pp. 112.

Gao Cong, Zhou Hong, Deng Qijun & Zhang Ziyang 2010, "The research and implementation of IEC 61850 communication protocol for smart grid", *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pp. 2463.

Hadeli, H., Schierholz, R., Braendle, M. & Tuduce, C. 2009, "Generating configuration for missing traffic detector and security measures in industrial control systems based on the system description files", *Technologies for Homeland Security, 2009. HST '09. IEEE Conference on*, pp. 503.

Haffar, M., Thiriet, J.M. & El-Nachar, M. 2010, "Software and hardware in the loop component for an IEC 61850 co-simulation platform", *Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 International Multiconference on*, pp. 817.

Hui Hou, Tianqi Xu, Dahai You, Xianggen Yin, Wei Chen, Bo Wang & Xiongkai He 2007, "Research on Protection Equipment with Digital Interface Based on IEC-61850", *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1.

Hyo-Sik Yang, Hyuk-Soo Jang, Yong-Won Kim, Un-Sig Song, Sang-Sig Kim, Byung-Tae Jang & Byung-Seok Park 2007, "Communication Networks for Interoperability and Reliable Service in Substation Automation System", *Software Engineering Research, Management & Applications, 2007. SERA 2007. 5th ACIS International Conference on*, pp. 160.

Ingram, M. & Ehlers, R. 2007. Toward effective substation automation. *Power and Energy Magazine, IEEE,* 5(3):67-73.

Jaloudi, S., Ortjohann, E., Schmelter, A., Sinsukthavorn, W., Wirasanti, P., Alamin, N. & Morton, D. 2011, "Integration of distributed energy resources in the grid by applying international standards to the inverter as a multifunctional grid interface", *PowerTech, 2011 IEEE Trondheim*, pp. 1.

Jinjiang Zhang, Chuangxin Guo, Lizhong Xu & Yijia Cao 2008, "The design of an Object-Oriented Embedded Platform for Substation Data Integration", *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pp. 1.

Jo YongHwan, Kim TaeWan, Choi MyeonSong, Lee SeungJae, Lee ByeongHo, Kim HyusSung & Cho ChulHee 2011, "A software engine for HMI of IED based on IEC 61850", *Advanced Power System Automation and Protection (APAP), 2011 International Conference on*, pp. 1312.

Johnsson, A., Söderström, J.E., Norberg, P. & Fogelberg, A. 2010, "Standard platform for integrated soft protection and control", *Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES*, pp. 1.

Kanabar, P.M., Kanabar, M.G., El-Khattam, W., Sidhu, T.S. & Shami, A. 2009, "Evaluation of communication technologies for IEC 61850 based distribution automation system with distributed energy resources", *Power & Energy Society General Meeting, 2009. PES '09. IEEE*, pp. 1.

Kasztenny, B., Whatley, J., Urden, E.A., Burger, J., Finney, D. & Adamiak, M. 2006, "JEC 61850 - A Practical Application Primer for Protection Engineers", *Power Systems Conference: Advanced Metering, Protection, Control, Communication, and Distributed Resources, 2006. PS '06*, pp. 18.

Kezunovic, M., Guo, C., Guan, Y. & Ghavami, M. 2010, "New concept and solution for monitoring and control system for the 21st century substation", *Power System Technology (POWERCON), 2010 International Conference on*, pp. 1.

Klein, S.A. 2009, "A Secure IEC-61850 Toolkit for Utility Automation", Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology, pp. 245.

Klein, S.A. 2008, "An open source IEC-61850 Toolkit for utility automation and wind power applications", *Transmission and Distribution Conference and Exposition, 2008. T&D. IEEE/PES*, pp. 1.

Klein, S.A. 2006, "An open source SCADA toolkit", *Power Engineering Society General Meeting, 2006. IEEE*, pp. 2 pp.

König, J., Nordström, L. & Ekstedt, M. 2010, "Probabilistic Relational Models for assessment of reliability of active distribution management systems", *Probabilistic Methods Applied to Power Systems (PMAPS), 2010 IEEE 11th International Conference on*, pp. 454.

Kostic, T. & Frei, C. 2007, "Modelling and using IEC 61850-7-2 (ACSI) as an API", *Power Tech, 2007 IEEE Lausanne*, pp. 713.

Kuffel, R., Ouellette, D. & Forsyth, P. 2010, "Real time simulation and testing using IEC 61850", *Modern Electric Power Systems (MEPS), 2010 Proceedings of the International Symposium*, pp. 1.

Kwon, S., Chu, C., Cho, J. & Yoon, S. 2012, "Development of IEC 61850 based Feeder IEDs for self-healing operation in distribution network", *Integration of Renewables into the Distribution Grid, CIRED 2012 Workshop*, pp. 1.

Laaksonen, H.J. 2010. Protection Principles for Future Microgrids. *Power Electronics, IEEE Transactions on*, 25(12):2910-2918.

Lehnhoff, S., Rohjans, S., Uslar, M. & Mahnke, W. 2012, "OPC Unified Architecture: A Service-oriented Architecture for Smart Grids", *Software Engineering for the Smart Grid (SE4SG), 2012 International Workshop on*, pp. 1.

Liang Du & Qun-ying Liu 2012, "The Design of Communication System on the Real-Time Relay Protection Based on GOOSE", *Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific*, pp. 1.

Lima, C., Gomes, V., Lima, J., Martins, J.F., Barata, J., Ribeiro, L. & Candido, G. 2011, "A standard-based software infrastructure to support energy efficiency using renewable energy sources", *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, pp. 1175.

Lima, J., Lima, C., Gomes, V., Martins, J.F., Barata, J., Ribeiro, L. & Candido, G. 2011, "DPWS as Specific Communication Service Mapping for IEC 61850", *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pp. 193.

Lin Zhu, Dongyuan Shi & Xianzhong Duan 2011. Standard Function Blocks for Flexible IED in IEC 61850-Based Substation Automation. *Power Delivery, IEEE Transactions on*, 26(2):1101-1110.

Liuhong Wei, Wenxin Guo, Fushuan Wen, Ledwich, G., Zhiwei Liao & Jianbo Xin 2011. An Online Intelligent Alarm-Processing System for Digital Substations. *Power Delivery, IEEE Transactions on*, 26(3):1615-1624.

Lorimer, M. & Zhang, R. 2008, "A Software Substation Interlocking Design Based on IEC 61850", *Developments in Power System Protection, 2008. DPSP 2008. IET 9th International Conference on*, pp. 35.

Ma WenJun, Mu LongHua, Ao Pei & Zhang Xin 2011, "New type merging unit for digital substation", *Advanced Power System Automation and Protection (APAP), 2011 International Conference on*, pp. 1598.

Maffezzini, I. & Premiana, A. 2006, "E-Learning whilst eliciting and working IEC 61850 Substation, a case study", *E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on*, pp. 135.

Manassero Junior, G., Pellini, E., Senger, E. & Nakagomi, R. 2012. IEC61850 based systems -- Functional testing and interoperability issues. *Industrial Informatics, IEEE Transactions on*, PP(99):1-1.

McDonald, J.D. 2003. Substation automation. IED integration and availability of information. *Power and Energy Magazine, IEEE*, 1(2):22-31.

Meliopoulos, A.P.S., Cokkinides, G.J., Mohagheghi, S., Dam, Q.B., Alaileh, R.H. & Stefopoulos, G.K. 2009, "A laboratory setup of a power system scaled model for testing and validation of EMS applications", *PowerTech, 2009 IEEE Bucharest*, pp. 1.

Mercurio, A., Di Giorgio, A. & Cioci, P. 2009. Open-Source Implementation of Monitoring and Controlling Services for EMS/SCADA Systems by Means of Web Services— IEC 61850 and IEC 61970 Standards. *Power Delivery, IEEE Transactions on*, 24(3):1148-1153.

Min Huang, Jianqing Xi, Yongli Zhu & Bo Sun 2008, "Research on the Application of Multi-Agent to the Telecontrol Communication for Power System", *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*, pp. 743.

Nair, S.A., Hourtoule, J., Gascon, J.C., Gulati, H.K., De La Calle, R., Lescure, C. & Singh, M. 2012. IEC-61850-Based Control System for Power Distribution at ITER. *Plasma Science, IEEE Transactions on,* 40(3):596-600.

Nian Liu, Bin Duan, Jian Wang & Shenglong Huang 2006, "Study on PMI based access control of substation automation system", *Power Engineering Society General Meeting, 2006. IEEE*, pp. 7 pp.

Nieves, J.C., de Mues, M.O., Espinoza, A. & Rodriguez-Alvarez, D. 2011, "Harmonization of semantic data models of electric data standards", *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pp. 733.

Pan Yun 2011, "Research on IED Configurator Based on IEC 61850", Control, Automation and Systems Engineering (CASE), 2011 International Conference on, pp. 1.

Paulo, R. & Carvalho, A. 2004, "Design oriented architecture for integration tools case study of substation automation systems", *Industrial Technology, 2004. IEEE ICIT '04. 2004 IEEE International Conference on*, pp. 874.

Peichao Zhang, Portillo, L. & Kezunovic, M. 2006, "Compatibility and interoperability evaluation for all-digital protection system through automatic application test", *Power Engineering Society General Meeting, 2006. IEEE*, pp. 7 pp.

Price, E. 2010, "The next step in the evolution of protection and control impelementation", *Protective Relay Engineers, 2010 63rd Annual Conference for*, pp. 1.

Qureshi, M., Raza, A., Kumar, D., Sang-Sig Kim, Un-Sig Song, Min-Woo Park, Hyuk-Soo Jang & Hyo-Sik Yang 2008, "A Communication Architecture for Inter-substation Communication", *Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on*, pp. 577.

Renhui Dou, Jie Ding & Yefei Zhou 2006, "The Data-View Model of IEC 61850 Server", *Power System Technology, 2006. PowerCon 2006. International Conference on*, pp. 1.

Santodomingo, R., Rodríguez-Mondéjar, J.A. & Sanz-Bobi, M.A. 2010, "Ontology Matching Approach to the Harmonization of CIM and IEC 61850 Standards", *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pp. 55.

Santodomingo, R., Rodriguez-Mondejar, J.A., Sanz-Bobi, M.A., Rohjans, S. & Uslar, M. 2011, "Towards the automatic alignment of CIM and SCL ontologies", *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pp. 422.

Sanz, R. 2002, "Embedding interoperable objects in automation systems", *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, pp. 2261.

Shi-Jaw Chen, Yi-Hsiang Wang, Chia-Hung Lin, Tung-Sheng Zhan, Rung-Fang Chang & Ya-Chin Chang 2012, "Using Multi-vendor IEDs for IEC 61850 Interoperability and HMI-SCADA Applications", *Computer, Consumer and Control (IS3C), 2012 International Symposium on*, pp. 745.

Song Yu, Wang Qian & Wang De Wen 2011, "Mapping method of SCL and CIM model based on the semantic network of knowledge representation", *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pp. 289.

Strasser, T., Stifter, M., Andren, F., Burnier de Castro, D. & Hribernik, W. 2011, "Applying open standards and open source software for smart grid applications: Simulation of distributed intelligent control of power systems", *Power and Energy Society General Meeting, 2011 IEEE*, pp. 1.

Sucic, S., Martinic, A. & Kekelj, A. 2012, "Utilizing standarDS-based semantic services for modeling novel Smart Grid supervision and remote control frameworks", *Industrial Technology (ICIT), 2012 IEEE International Conference on*, pp. 409.

Sugwon Hong, Dae-Yong Shin & Myongho Lee 2009, "Evaluating Security Algorithms in the Substation Communication Architecture", Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009. SCALCOM-EMBEDDEDCOM'09. International Conference on, pp. 314.

Sugwon Hong, Myongho Lee & Dae-Yong Shin 2010, "Experiments for Embedded Protection Device for Secure SCADA Communication", *Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific*, pp. 1.

Thomas, M.S., Kothari, D.P. & Prakash, A. 2011. Design, Development, and Commissioning of a Substation Automation Laboratory to Enhance Learning. *Education, IEEE Transactions on,* 54(2):286-293.

Wang Haizhu, Cai Zexiang, Su Zhongyang & Zhu Zhihan 2011, "Analysis of realtime performances of process-level networks based on IEC61850 SCSM Model", *Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011 4th International Conference on*, pp. 1820.

Weifeng Xin, Guogang Zhang, Yuchen Wang, Yingsan Geng & Bo Liu 2011, "Design of intelligent component for high voltage GIS based on IEC61850", *Electric Power Equipment - Switching Technology (ICEPE-ST), 2011 1st International Conference on*, pp. 262.

Xyngi, I. & Popov, M. 2010, "Smart protection in Dutch Medium Voltage distributed generation systems", *Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES*, pp. 1.

Yang, L., Crossley, P.A., Wen, A., Chatfield, R. & Wright, J. 2011, "Performance assessment of a IEC 61850-9-2 based protection scheme for a transmission substation", *Innovative Smart Grid Technologies (ISGT Europe), 2011 2nd IEEE PES International Conference and Exhibition on*, pp. 1.

Yi, Y.H., Zhang, J.J., Liu, B., Xu, L.Z., Cao, Y.J. & Guo, C.X. 2008, "A new-style centralized IED based on IEC 61850", *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pp. 1.

Yiqing Liu, Houlei Gao, Mingjiang Xiang, Xin Wei, Peng Wei & Chunsheng Zhou 2011, "Performance testing of complete digital relays based on ATP-EMTP and IEC61850-9-2", *Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011 4th International Conference on*, pp. 83.

Zhabelova, G., Vyatkin, V. & Nair, N.C. 2011, "Standard-based engineering and distributed execution framework for intelligent fault management for FREEDM system", *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, pp. 2724.

Zhang, L.X. & Nair, N.-.C. 2008, "Testing protective relays in IEC 61850 framework", *Power Engineering Conference, 2008. AUPEC '08. Australasian Universities*, pp. 1.

Zhaojie Sun & Bin Duan 2012, "The design of function block in IED with digital substation operation", *Innovative Smart Grid Technologies - Asia (ISGT Asia), 2012 IEEE*, pp. 1.

Zhu Yongli, Wang Dewen, Wang Yan & Zhao Wenqing 2009, "Study on interoperable exchange of IEC 61850 data model", *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*, pp. 2724.