FAKULTÄT FÜR
ELEKTROTECHNIK UND
INFORMATIONSTECHNIK
Faculty of Electrical Engineering and Information Technology

Synchronizing IEC 61850-9-2 with Legacy Real-time Embedded Systems

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Embedded Systems

by

Mario PETER, BSc
Registration Number 01327117

to the Faculty of Electrical Engineering and IT

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo SAUTER

Vienna, 23rd September, 2021

Mario PETER	Thilo SAUTER

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Danksagung

An dieser Stelle möchte ich mich bei all jenen Personen recht herzlich bedanken, die mich während meines Studiums und dem Schreiben dieser Diplomarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Herrn Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo SAUTER, der diese Arbeit betreut und begutachtet hat. Weiters möchte ich mich bei der Firma OMICRON electronics GmbH[1] und insbesondere bei meinen firmenseitigen Betreuern Herrn Dipl.-Ing. Tilman TRÄXLER und Herrn Domen SELINEC, MSc recht herzlich bedanken, da sie mir die Möglichkeit gegeben haben, diese Industrie-Diplomarbeit durchzuführen.

Neben den Betreuern möchte ich mich auch noch bei meinen Freunden und Studienkollegen bedanken, die mir bei Fragen oder Unklarheiten geholfen haben.

Abschließend möchte ich mich bei meinen Eltern bedanken, die mir mein Studium ermöglicht haben und stets ein offenes Ohr für meine Sorgen hatten.

[1] https://www.omicronenergy.com/de/

Acknowledgements

At this point I would like to thank all those peoples who supported and motivated me during my studies and the writing of this diploma thesis.

First, my thanks go to Mr. Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo SAUTER, who has supervised and reviewed this work. Furthermore, I would like to thank the company OMICRON electronics GmbH[2] and in particular my company advisers Mr. Dipl.-Ing. Tilman TRÄXLER and Mr. Domen SELINEC, MSc because they gave me the opportunity to do this industrial diploma thesis.

In addition to the caregivers, I would also like to thank my friends and colleges from university that have helped me with questions or ambiguities.

Finally, I would like to thank my parents, who made my studies possible and always had an open ear for my worries.

[2] https://www.omicronenergy.com/en/

Kurzfassung

Die elektrische Infrastruktur lässt sich in zwei Bereiche aufteilen. Der erste Bereich ist die Primärtechnik. Diese umfasst alle Schaltanlagen, die unmittelbar an der Erzeugung, dem Transport und der Verteilung elektrischer Energie beteiligt sind. Somit beinhaltet die Primärtechnik beispielsweise Generatoren, Transformatoren, Schaltgeräte, Sammelschienen und Leitungen. Neben der Primärtechnik gibt es ergänzend dazu auch noch die Sekundärtechnik. Dieser Bereich beinhaltet sämtliche Hilfseinrichtungen zur Messung, Überwachung, Kommunikation, Automatisierung und Fernsteuerung. Die Verbindung der zwei Bereiche wird über Messwandler und Leistungsschalter realisiert.

Wie in fast allen anderen Bereichen und Branchen durchläuft auch die elektrische Infrastruktur einen Digitalisierungsprozess. In dieser Diplomarbeit wird der Fokus auf die Digitalisierung der Messwandler gelegt. Der Begriff Messwandler umfasst Stromwandler (Current Transformers (CTs)) und Spannungswandler (Voltage Transformers (VTs)). Durch diese Änderung wird die Norm International Electrotechnical Commission (IEC) 61850 sehr wichtig. Für diese Diplomarbeit ist besonders der Bereich über Transport von Sampled Values (SVs) über Ethernet-Netzwerke interessant, welcher im IEC 61850-9-2 zu finden ist. Dieses Kapitel ist von Bedeutung, da es bei unkonventionellen Wandler keine Möglichkeit gibt die Sekundärseite analog zu messen, sondern nur noch über die SVs, welche über die standardisierte Bus-Schnittstelle zur Verfügung gestellt werden.

Die Messwandler können weiterhin mit einem analogen Signal auf der Primärseite gespeist werden, jedoch kann die Antwort des Messwandlers nur noch durch das Konsumieren und Interpretieren des SV-Streams erreicht werden. Die SVs werden mit einer von er Netzfrequenz abhängigen Abtastrate generiert und dann übers Netzwerk verteilt. Bei einer Netzfrequenz von 50 Hz ist die Abtastrate 4000 Hz und 4800 Hz bei einer Netzfrequenz von 60 Hz. In dieser Diplomarbeit wird ein solcher SV-Stream von einem existierenden Echtzeit-Embedded-System konsumiert, wobei das System mit einer nicht veränderbaren Systemfrequenz getaktet ist, die sich von der Abtastrate der SVs unterscheidet. In unserem Fall ist das System mit 10 kHz getaktet und daher wird in dieser Diplomarbeit eine Möglichkeit aufgezeigt, wie man diese SVs in das vorhandene Echtzeit-Embedded-System integrieren kann.

Damit das System die empfangenen SVs konsumieren und interpretieren kann, müssen diese ebenfalls auf eine Abtastrate von 10 kHz transformiert werden. Mit Hilfe der Kubische Spline Interpolation wird die Abtastrate des empfangenen digitalen Signals

auf ein Vielfaches der Systemfrequenz erhöht. Anschließend werden die nicht benötigten Abtastwerte wieder entfernt, damit man am Ende eine Abtastrate von 10 kHz erreicht. Dieses transformierte Signal kann dann von der Echtzeitanwendung verwendet werden, um diverse Berechnungen durchzuführen.

In dieser Diplomarbeit konnte aufgezeigt werden, dass die gezeigte Implementierung keinen Einfluss auf das existierende Echtzeit-Embedded-System hat. Weiters wird gezeigt, dass der Fehler, der durch die Transformation des Signals entsteht, in einem sehr guten Bereich liegt und für die Anwendung ausreichend ist. Aufgrund der begrenzten Rechenleistung des Systems und der bereits hohen Auslastung der Echtzeitanwendung konnte die Evaluierung nur mit einem von acht Kanälen des SV-Streams durchgeführt werden. Die vorgestellte Lösung kann aber jederzeit erweitert werden, damit alle empfangenen Ströme und Spannungen transformiert und verwendet werden können.

Abstract

The electrical infrastructure can be divided into two areas. The first area is primary technology. This includes all switchgears that are directly involved in the generation, transport and distribution of electrical energy. Thus, the primary technology includes for example generators, transformers, switching devices, busbars and cables. In addition to the primary technology there is also the secondary technology. This area contains all auxiliary equipment for measurement, monitoring, communication, automation and remote control. The connection between the two areas is implemented using instrument transformers and circuit breakers.

As in almost all other areas and industries the electrical infrastructure is also going through a digitization process. In this diploma thesis the focus is placed on the digitization of the instrument transformers. The term measuring transformer includes Current Transformers (CTs) and Voltage Transformers (VTs). With this change a standard of the International Electrotechnical Commission (IEC) becomes very important. The mentioned standard is IEC 61850. For this diploma thesis the area about the transport of Sampled Values (SVs) over Ethernet networks is particularly interesting, which can be found in IEC 61850-9-2. This chapter is important because with Non-Conventional Instrument Transformers (NCIT) there is no possibility to measure the secondary side analogously, but only via the SVs, which are made available via the standardized bus interface.

This diploma thesis shows how this SVs can be integrated into an existing real-time embedded system that works with a specific and unchangeable system frequency. In this special case Ethernet for Control Automation Technology (EtherCAT) specifies the frequency, which is for the used system a frequency of 10 kHz. Since the SVs are received at a different frequency, an interface has to be created that transforms this SVs to the given system frequency. This data can only be used if the received SVs have been synchronized to the system frequency. Otherwise, there is no relationship between the measuring signals and the received result data.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement

As in almost all other areas and industries, the electrical infrastructure is also going through a digitization process. Through this digitization, the conventional instrument transformers (including Current Transformers (CTs) and Voltage Transformers (VTs)) are replaced by so called Non-Conventional Instrument Transformers (NCIT), which have only digital output signals on the secondary side of the transformer and no analog outputs anymore. Therefore, the measurement methods also have to be adapted to the new situation.

These NCIT are more frequently used not only because of digitization, which is currently the trend, but also because they operate in non-conventional ways. One example of this is that NCIT sometimes also use optical technology and thus have some advantages over conventional transformers [TNT+97]. For example a fiber optic cable is used as a sensor and is mounted around the conductor to measure the current. This is done based on the Faraday effect. This effect describes the rotation of the plane of polarization of a light beam by a magnetic field [CXW20]. Another reason for using NCIT is that they are capable of higher voltages and currents while still being as compact as possible. In order to achieve this compactness, the transformers must be isolated with a special gas and then can not simply be broken up to make a measurement [TNT+97].

Therefore, the standard International Electrotechnical Commission (IEC) 61850 on "Communication Networks and Systems in Substation" becomes an important topic which defines communication protocols, data formats and the configuration language for Intelligent Electronic Devices (IEDs) at electrical substations. All parts of the standard are listed in Table A.1. For this thesis the subsection IEC 61850-9-2 is the most relevant one and defines the protocol how Sampled Values (SVs) shall be transmitted over an Ethernet network. In this context, SVs are discrete samples of the voltages and currents

of the transformer. So all analog voltages and currents are sampled with a defined sample rate and these samples are summarized in the SV packet with the associated phases and status information and then transmitted over the network.

With this digital interface we face now the situation that the transformers on the primary side can still be excited with an analog signal, but there are no analog ports on the secondary side to measure the response analogously. We only have the possibility to consume the generated SVs and thus receive the response of the transformer. The values of the secondary side are very important, since many of the test methods used in asset diagnostics allow a conclusion only with this data and in combination with the measuring signal on the primary side. Such a measurement is for example to measure the transmission accuracy or the phase shift error of the transformer or to identify short windings on instrument transformers by measuring the transformation ratio [PFA17].

Figure 1.1 shows a measurement setup for a conventional instrument transformer where an analog signal is output from the measuring device to the primary side of the transformer to be tested. The secondary side is connected to the measuring device with some cables, so that an analog signal can also be received from this side. Both sides are connected directly to the device under test so that there is almost no time delay between outputting some signals and receiving the corresponding response form the asset.

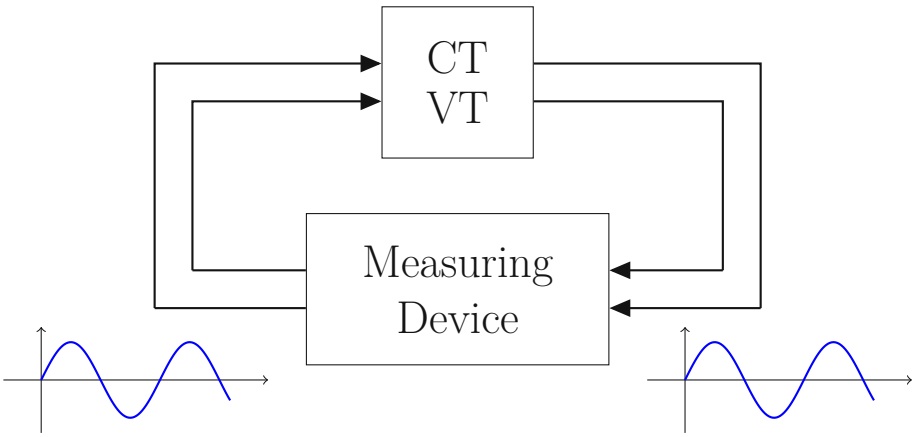


Figure 1.1: Measuring setup for a conventional instrument transformer

In contrast to that, Figure 1.2 shows a measurement setup for a NCIT. Also in this setup we have a measuring device which is directly connected to the primary side of the transformer and outputs an analog signal. The NCIT looks just a little different compared to the conventional one. Beside the CT or VT there is also a Merging Unit (MU) integrated into the transformer. The main task of this MU is to convert the analog signals to time discrete digital signals and provide them in the IEC 61850-9-2 format as SVs over an Ethernet interface. In most of the cases the SV packets will be transmitted

over a Local Area Network (LAN) and can than be consumed from the measuring device.

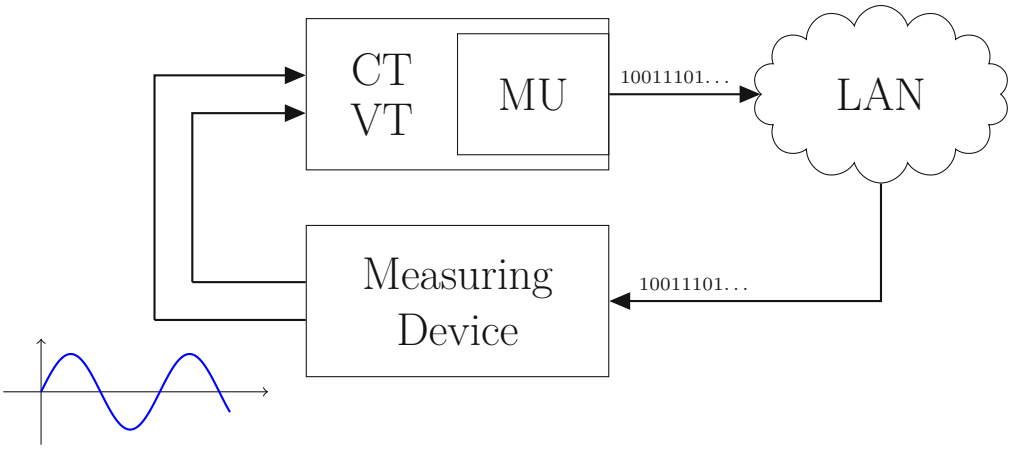


Figure 1.2: Measuring setup for a NCIT

For this thesis we assume that the measuring device is a hard real-time embedded system with a specified system clock which is higher than the sample rate of the SVs. So we have a thread running with the hard real-time system clock and we have another thread which runs based on the received SV packets. This system clock should be as high as possible to achieve the best results, but is limited by the Central Processing Unit (CPU) clock rate and the tasks which have to be performed in each iteration. An illustration of this architecture is shown in Figure 1.3. As an example we used here a system clock frequency of 10 kHz which means that the thread is executed every 100 µs. On the other hand we used the sample rate of 4000 Hz which is defined in the IEC 61850-9-2 for an operating frequency of 50 Hz otherwise it would be 4800 Hz for an operating frequency of 60 Hz. The received SVs will then be forwarded over Inter-Process Communication (IPC) to the main thread.

Furthermore, we can already see in Figure 1.2 and 1.3 some challenges. One challenge is that the sample rate of the SVs will be 4000 Hz, but the packets will arrive at the measuring device with some jitter because of the network latency which could be different for each packet, so we have to find a way to solve this issue. The main problem is, that the two threads have completely different clock rates and it is not an integer multiple so we have to transform the SVs sampled with 4000 Hz to SVs which were sampled with 10 kHz so that the main thread can use this data. Another point to keep in mind is that we have to use a solution for the IPC which does not use any mutex. A mutex is mechanism in computer science that restricts access to a shared resource so that only one thread can access the shared resource at a time. In our case we have a shared memory where one thread writes the data and the other one is reading the data. Therefore, we need a solution where reading and writing is possible at the same time because waiting for

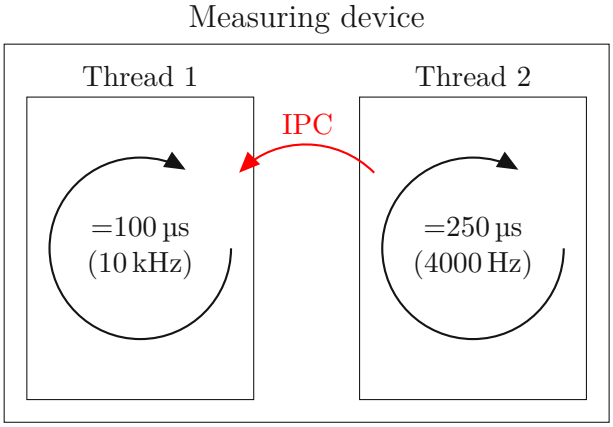


Figure 1.3: Thread overview of the measuring device

the mutex to be released from the other thread could break the hard real-time condition because the main thread has no time left for waiting. Nevertheless, thread safety must be guaranteed at all times, even without using a mutex.

Another important point is that the NCIT is connected to a high accuracy time source so that the signals can be sampled with a high precision resulting in equidistant samples. In the past a 1 Pulse Per Second (PPS) signal was used to synchronize the local running clock on the device. Over time the 1PPS signal was replaced by the Institute of Electrical and Electronics Engineers (IEEE) 1588 protocol which can be used to time synchronize multiple devices over Ethernet [SNKB19]. Therefore, it is mandatory that the measuring device can also receive the IEEE 1588 packets and can synchronize the local time to the time information provided form the Precision Time Protocol (PTP) grandmaster clock. In our case the grandmaster provides a Global Positioning System (GPS) controlled time reference and distributes this time information to the measuring device as well as to the integrated MU of the device to be tested.

A last point to consider is that the solution will be implemented and evaluated on a hard real-time embedded system with limited resources and computation power. Therefore, the solutions should not be too complex and be as resource efficient as possible so that the already running hard real-time system does not get disturbed.

1.2 Aim of the Thesis

The aim of this thesis is to provide some background information about the related topics and the different standards, which are parts of this problem. Another goal of this thesis is to deal with the time jitter of the received SVs which is caused by the network. The main intention is to find a time and resource efficient solution to exchange sample data between two threads which are running on completely different clock frequencies. A

practical implementation should be based on a commercial off the shelf IPC architecture. Therefore, we need a smart solution to use shared memory without using any kind of mutual exclusion mechanism because the hard real-time system has no time left for waiting to access a shared resource. For performance evaluation, an existing measuring device should be extended with this solution and evaluate what is possible and what restrictions the implementation has on the test device. An important point here is that the already existing hard real-time embedded system works as before and is not disturbed by the solution of this thesis. The last aim of this thesis is to evaluate the implementation on the measuring device and compare the results with expected results from different simulations and check the timing behaviour of the system.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows. Chapter 2 provides the reader with helpful background information related to the topic of this thesis. General terms in the field of power supply systems are explained and in Section 2.2 and 2.3 the necessary standards are specifically addressed and explained. In Section 2.5 the measuring device is described in detail, which was used to implement and evaluate the solution.

In the next Chapter 3 we provide an overview about existing state-of-the-art approaches to resampling and synchronization of two threads with different iteration cycle times.

The implementation of the resampling algorithm and the bridge between the SV-system and the Ethernet for Control Automation Technology (EtherCAT)-master is documented in Chapter 4.

Chapter 5 provides the evaluation of the existing embedded system with the synchronization to the SV-system.

Finally, in Chapter 6 the thesis is concluded with a summary of the presented work and give an outlook on future work.

CHAPTER 2

Background

2.1 Electric power system

In 1882, the first power station was commissioned in New York City. The power industry grew very fast and generation stations and transmission and distribution networks spread all over the country [Gö14].

The main tasks of an electrical power system are to generate, transmit and distribute electrical energy efficiently. In electric power systems, generators produce the energy as needed. They are therefore not storage systems such as water or gas systems [Blu07].

A power system can be divided into two major infrastructures. On the one hand it is the power grid and on the other the communication system. The power grid will be described in the next paragraph and the communication system will be described in the next Section 2.2 [SW05].

Figure 2.1 shows the different stations from power generation to the consumer. It is a simplified power system, but the basic principles, concepts, theories and terminologies are the same. On the left side of the image the power generation in the power plant is shown as a first step. The power plants convert energy sources into electrical energy. Such energy sources are, for example, heat, mechanical, hydraulic, chemical, solar, wind, geothermal, nuclear and other energy sources. Subsequently, this electrical energy is converted at the power station into high-voltage electrical energy, as this is better suited for efficient long-distance transport. The next part is the high-voltage transmission lines, which efficiently transport electrical energy over long distances to the consumption locations. At the end of the high-voltage lines are substations, which convert the electrical high-voltage into a medium-voltage. Subsequently, this medium-voltage energy is transmitted via distribution power lines and can be directly connected to primary customers such as bigger industries. Last but not least, the energy is converted once more close to the secondary customer to the low-voltage level. In Figure 2.1 the transformer is attached

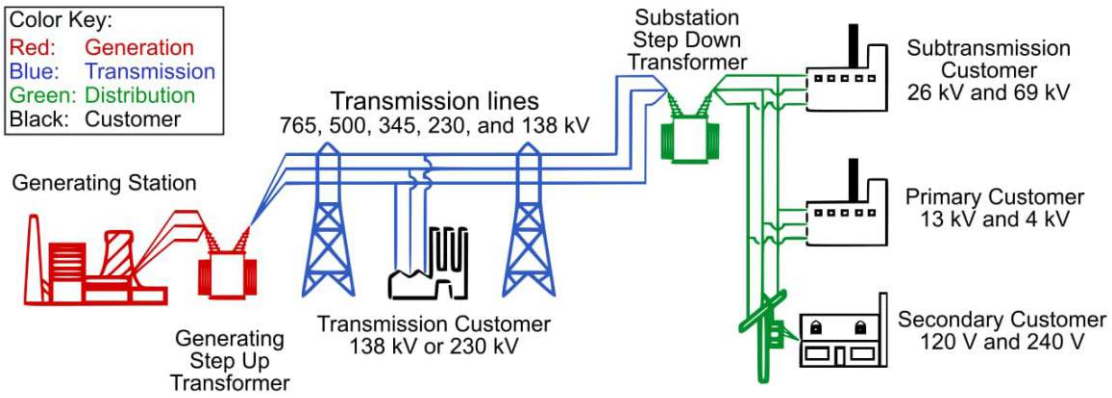


Figure 2.1: Electric power system overview [Blu07]

to the power pole and does the conversion from the medium voltage to the low voltage level [Blu07].

Figure 2.2 shows the required geographically distributed and functionally complex monitoring and control systems. As indicated in Figure 2.2 the Energy Management System (EMS) exercises overall control over the total system. The Supervisory Control and Data Acquisition (SCADA) system includes generation and transmission systems that perform automatic monitoring and control functions in the system. In addition to the distribution system, the Distribution Automation and Control (DAC) system also monitors the connected load [Gö14].

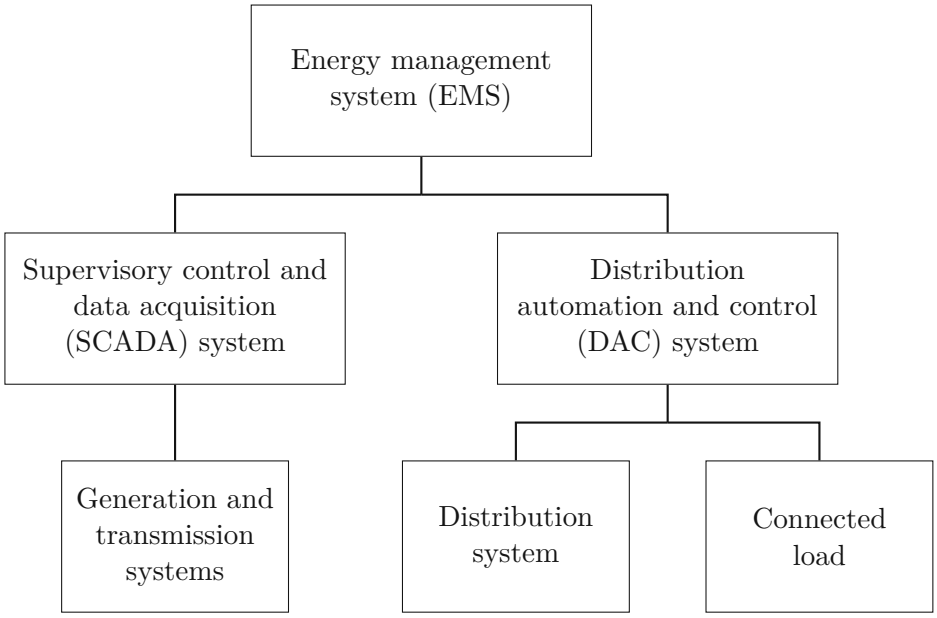


Figure 2.2: Monitoring and controlling of an electric power system [Gö14]

2.2 IEC 61850 – Communication networks and systems for power utility automation

Compared to the past the protection functions have improved considerably in terms of time behaviour. This is partly due to the much faster relays which are now based on numerical technologies and partly because of the excellent development of communication capabilities. The protection systems have profited not only by increasing the operating speed but also by simplifying equipment and hardware. These improvements increased reliability and improved the cost-benefit ratio. Considering the large application not only for relay applications but also for all aspects of power automation modern communications developments and in particular the IEC 61850 standard is a must [GH11].

In the past substations used protection and control schemes implemented with single-function, electromechanical or static devices and hardwired relay logic. The SCADA functions were centralized and limited to monitoring the most basic functions. Fault and event sequence data could only be recorded centrally and locally in the substation. Today there are microprocessor-based IEDs that can perform multiple functions. Therefore, fewer devices are needed resulting in simpler designs with reduced wiring. Furthermore, the IEDs are capable of communication and therefore they can be accessed remotely. Thus, it is possible to read or control the system without having to be at the substation. These IEDs are used everywhere because it saves a lot of space and money. In addition, these devices can be integrated very well into preexisting frameworks of relay application. Such a IED can replace an entire field of electromechanical relays. In terms of SCADA

integration, there were initially interoperability issues when IEDs from different vendors were used. For this reason initiatives were founded in the early 1990s to design a communication architecture which should facilitate the design of systems for protection, control, monitoring and diagnostics in the substation. This unified architecture is intended to simplify the development of Substation Automation Systems (SAS) from multiple vendors and thus achieve a higher level of integration. These initiatives have given rise to the precursor to the international standard IEC 61850 [GH11].

The first edition of the standard IEC 61850 had the title "Communication networks and systems in substations" and was published between 2003 and 2005. The standard has been revised several times over time and has been expanded for automation outside substations. Therefore, the title of the standard was no longer appropriate and was changed to "Communication networks and systems for power utility automation" [PA18].

The intention of the standard is to simplify the interoperability between IEDs from different manufacturers. This can be achieved by standardizing the information used in power utility automation and provide it in an object-oriented manner. The IEC 61850 only provides a standardized information model and defines how information should be exchanged between the devices, but does not define any functionality of the devices. To define the functionality of a device other standards or methods have to be applied. Such a standard could for example be IEC 61131-3 or IEC 61499 which are also used in the area of programming Programmable Logic Controllers (PLCs) [SSA+11].

The current version of the standard consists of 35 different parts and technical reports (see Table A.1), whereby the content can be divided thematically into the following three main topics [TC 19]. A graphical representation of how these three topics work together is shown in Figure 2.3.

Modelling This topic deals with the mapping of physical system data into a formal data model which is defined by the standard. Logical Nodes (LNs) are used to model devices and functions of the power system. Among other things, they can be used to model power system components such as switches, transformers or inverters as well as power system functions such as measurement, protection or voltage control which can also be seen in Figure 2.3. Therefore, they are among the most important part of modelling [PA18]. Figure 2.4 shows an example of a LN for a non-phase related measurement (MMXN). It can be seen that the properties of LNs are presented in a tabular format [SSA+11].

Configuration The second main topic of IEC 61850 is the configuration of the other two main topics. Part 6 – "Configuration description language for communication in electrical substations related to IEDs" – of the standard describes the System Configuration description Language (SCL) based on Extensible Markup Language (XML). With this configuration tool it is possible to configure a complete power utility system [SSA+11]. The SCL can be used to describe which components and functions are modelled, from a whole substation to a single IED. Furthermore, it

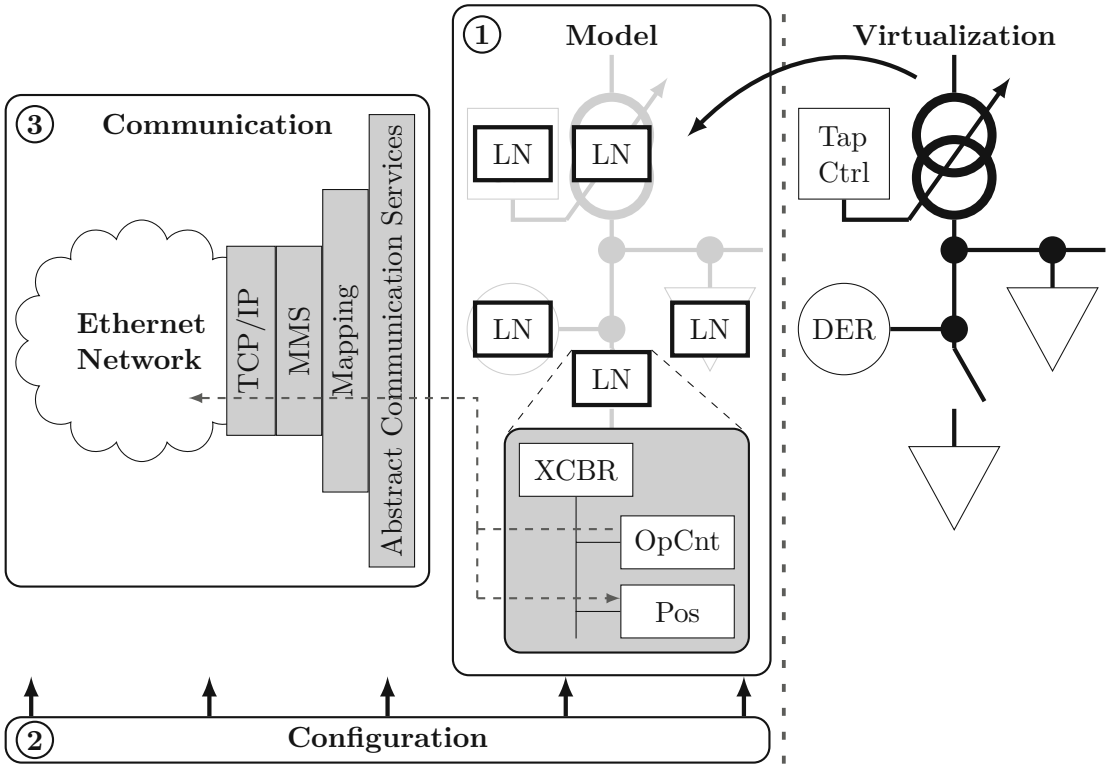


Figure 2.3: Overview of IEC 61850 three main topics: (1) Modelling, (2) Configuration, (3) Communication [PA18]

can also be configured which service and which protocol should be used to exchange information. Finally, SCL also provides tools for modeling and configuring parts of the communication network. The result is a configuration file that can be used to configure the different devices in the system [PA18].

Communication The last main topic defines the different communication services that are available to transfer data between IEDs. For example, client-server-based communication is defined where the information is polled or reported, as well as publisher-subscriber-based communication or real-time communication using Generic Object Oriented Substation Event (GOOSE). GOOSE offer a fast and reliable transmission of critical substation events between control and protection devices which improves system stability and performance [HHFK17]. The standard also defines different mappings between communication services and different protocols. An example of such a mapping is the mapping of a client-server communication to the Manufacturing Message Specification (MMS) which is defined in Part 8-1 – "Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3" [TC 19].

MMXN		
Data object name	Common data class	Explanation
Common LN information		
Beh	ENS	Behaviour
Measured and metered values		
Amp	MV	Current I
Vol	MV	Voltage V
Watt	MV	Power (P)
VolAmpr	MV	Reactive power (Q)
VolAmp	MV	Apparent power (S)
PwrFact	MV	Power factor
Imp	CMV	Impedance
Hz	MV	Frequency

Figure 2.4: Example of a logical node for a non-phase related measurement [SSA+11]

Besides the problem that the function of the devices is not defined in the standard there is another problem. The IEC 61850 approach is not very user-friendly. The LN data model is only displayed in text tables, as shown in Figure 2.4. Furthermore, it is nessecery to switch between different parts of the standard to find all defined subtypes of a LN. This is not only time-consuming but also extremely error-prone. The available tools to assist the system administrators are mostly commercial or proprietary products that are difficult to integrate into a rapid engineering method. The TC 57 (Technical Committee) therefore works on a UML presentation of LNs [PA18].

2.2.1 IEC 61850-9-2

IEC 61850-9-2 has proposed an Ethernet based communication network between switch-yard equipments and IEDs. MUs are the key elements of this communication network which is also call process bus. MUs gather information, such as voltages and currents from instrument transformers and convert the analog values to digital values, also known as SVs. After the conversion the SVs are encapsulated and transmitted as a multicast service over Ethernet [Sam17].

That the IEDs can function correctly the SVs have to be transmitted fast and cyclic and they should not be lost or delayed over the network. In most cases, the MUs are also in a IEEE 1588 network, so that the SVs can be provided with an exact time stamp. The standard specifies that a maximum communication delay of 3 ms is allowed for the highest class. This must be achieved at all times, no matter how much traffic is on the

process bus communication network. GOOSE and SV packages belong to the highest class and thus to the most time-critical messages. These messages are mapped directly to the Data Layer and not to the Transport Layer of the OSI Model. As a result, the time-critical messages are transmitted without transmission reliability. This means that no acknowledgment will be sent if the packet has been received or the packet is not retransmitted if it is lost during transmission. Therefore, it has been defined in the Standard IEC 61850-8-1 that GOOSE messages should be transmitted several times in order to improve transmission reliability. This approach can not be used for SVs as they are continuously fed into the network by multiple MUs at a rate of 80 Samples Per Cycle (SPC). By sending the same package multiple times, the network load would be increased enormously. It was therefore decided that there are no assurance or reliability measures for SV communication over the process bus [KSZ11].

On the other hand, the IEC 61850-9 process bus also offers advantages. First, it's interoperability. With this process bus, all connected devices can exchange any information with the IEDs or MUs, even if they are from different manufacturers. Furthermore, the entire communication network is simplified, as many point-to-point connections are replaced by a few Ethernet connections. This makes the entire communication network much clearer and easier to maintain. And last but not least, both the installation costs and the labor costs can be significantly reduced because, as already mentioned, the entire network can be simplified [KSZ11].

2.2.2 IEC 61850-9-2-LE

IEC 61850-9-2-LE (Light Edition) is just an implementation guideline and is based on the IEC 61850 standard. This document is used by most vendors manufacturing IEC 61850-9-2 compatible devices. The guideline was published by the UCA International Users Group. Among other things, this guideline defines that a SV stream transmits 8 instantaneous values. The three phases and neutral values for current and voltage. These values are measured at a fixed sampling rate of 80 or 256 SPC. This means that 4000 or 12 800 frames per second are transmitted at a network frequency of 50 Hz [KRP18].

In Figure B.1 shows the content of an Ethernet frame, which is used to transmit SVs. Furthermore, in Figure B.2 the Application Protocol Data Unit (APDU) data field is shown in more detail. As demonstrated the APDU data field is divided into many values. The sample counter (smpCnt) is one of those values. This value allows the receivers to match the samples of different MUs in the correct order. This value is set to zero after each period and then incremented by one with each sample [HS07].

2.3 IEEE 1588

In a SAS all devices must have the same time reference, so that the global time behaviour can be analyzed and in case of a fault, it can be precisely analyzed why, when and where the error occurred. For samples of currents and voltages a time synchronization accuracy

in the order of 1 µs is needed. For the fault detection and location on the transmission lines of the power grid a synchronization accuracy of less than 1 µs is needed because a time error of 1 µs results in a location error of 300 m [BLW03, MML+16].

Inter Range Instrumentation Group-B (IRIG-B) was originally used as a synchronization scheme in substations. This allowed an accuracy of 1 µs to be achieved. To achieve this accuracy, an extra cabling infrastructure had to be created, which in most cases was not redundant and therefore very prone to error. As a result, the implementation and maintenance costs were extremely high. Another problem was that the various devices were located at different distances from the control building, resulting in different propagation delays that could only be compensated with the help of complicated and bothersome calibration processes [DFF+11].

Today, the trend is toward using a single Ethernet-based data network for all kind of communication. With the Simple Network Time Protocol (SNTP) an accuracy of 1 ms can be achieved, which is needed to perform a post-event fault analysis. However, most experts recommend for high precision time synchronization in SASs to use the PTP, which is defined in the IEEE 1588 standard [MML+16].

PTP is much better compared to SNTP because PTP automatically compensates for propagation delays and distributes absolute time directly over Ethernet in the substation. Furthermore, the time can be transmitted over redundant Ethernet networks, which results in a higher reliability of the time distribution. The PTP provides absolute time with nanosecond accuracy. Figure 2.5 shows that the PTP is based on a master-slave hierarchy where the master imposes the time by sending regular sync messages to the slaves with exact timestamps. The slaves synchronize to the master both in phase and in frequency. By time measurements between consecutive sync messages, the slave can adjust its clock frequency to be the same as that of the master. These adjustments are achieved by a control loop which increases or slows down the clock frequency of the slave [MML+16].

In Figure 2.5 the protocol message exchange pattern is shown. The master first sends a multicast sync packet to the slaves containing the local time. The actual time, if a hardware time stamp is used, is recorded near the physical layer and transmitted to the software driver. The master then sends a follow-up packet containing the time stamp of the previous sync packet. The receiver can determine the time difference between its clock and the master clock by receiving the sync packet and the contents of the follow-up packet. However, this also includes the propagation delay between master and slave (Equation 2.1a). To determine this time, the slave sends a delay request packet to the master. The master responds to the slave by means of a delay response packet in which the local reception time of the delay request packet is contained (Equation 2.1b). Starting from a symmetrical delay in both directions, the propagation delay can thus be determined (Equation 2.1c) [Exe07].

$$Delay + Offset = t_2 - t_1 \tag{2.1a}$$

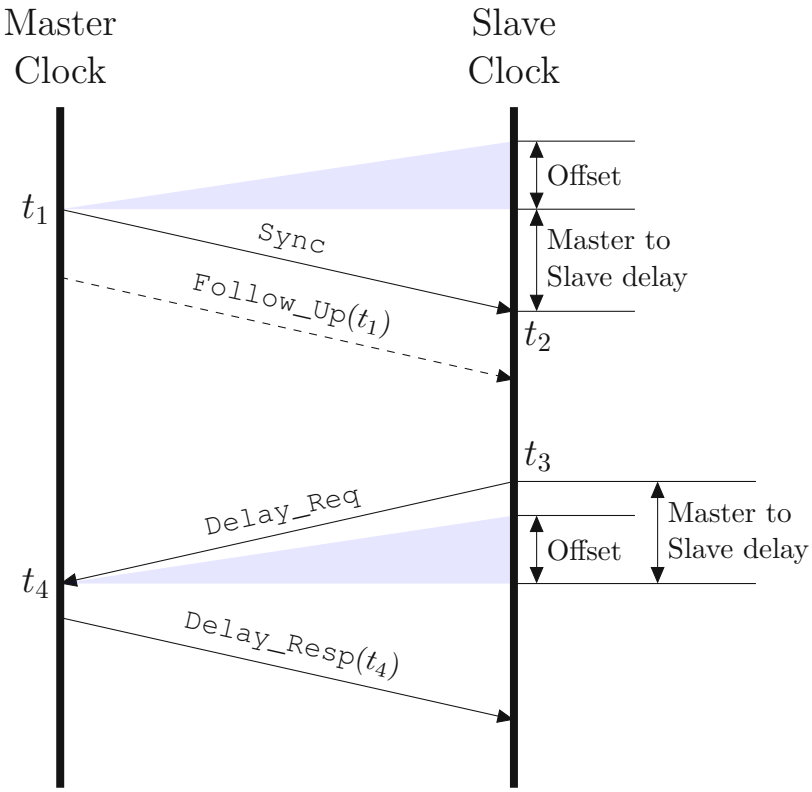


Figure 2.5: IEEE 1588 protocol message exchange pattern [MML+16].

$$Delay - Offset = t_4 - t_3 \tag{2.1b}$$

$$Delay = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{2.1c}$$

The measurement of the propagation delay takes place in intervals of 1 s to 64 s. The synchronization process by sending the sync packets is done every 2 s to 8 s, because the drift rate of the local clocks generally causes a greater impact per time interval than the slow changing propagation delay. The accuracy of clock synchronization achieved depends primarily on whether PTP is implemented using hardware time stamping or not [MES14].

2.4 EtherCAT

EtherCAT was developed by Beckhoff[1] and is part of the IEC standard since 2005. The technology is supported and fostered by users who have joined the EtherCAT Technology

[1] https://www.beckhoff.de/

Group (ETG) [Dü17].

A EtherCAT system consists of a single master and several slaves logically connected by a ring topology, as shown in Figure 2.6. The subscribers of a EtherCAT network are physically arranged in a line if there is no branch. However, as illustrated in Figure 2.6, it is also possible to build a physical tree topology by branching in the return channel. The telegram first runs from a branch along a path to the last participant of this path and is sent back by this to the branch. There the telegram is then directed to the other path. Logically speaking, it is still a ring topology [WB05].

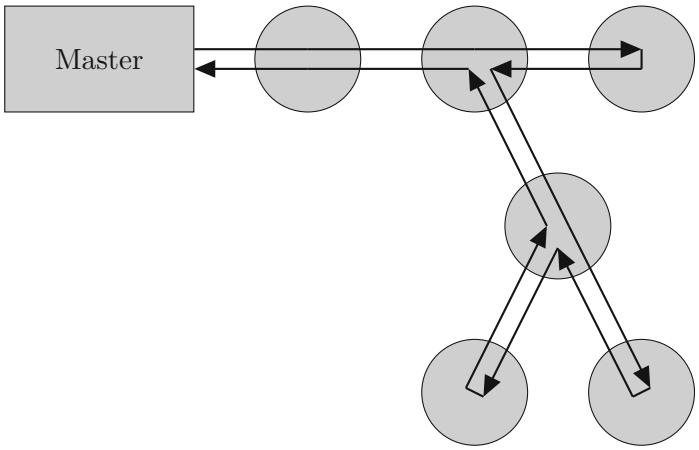
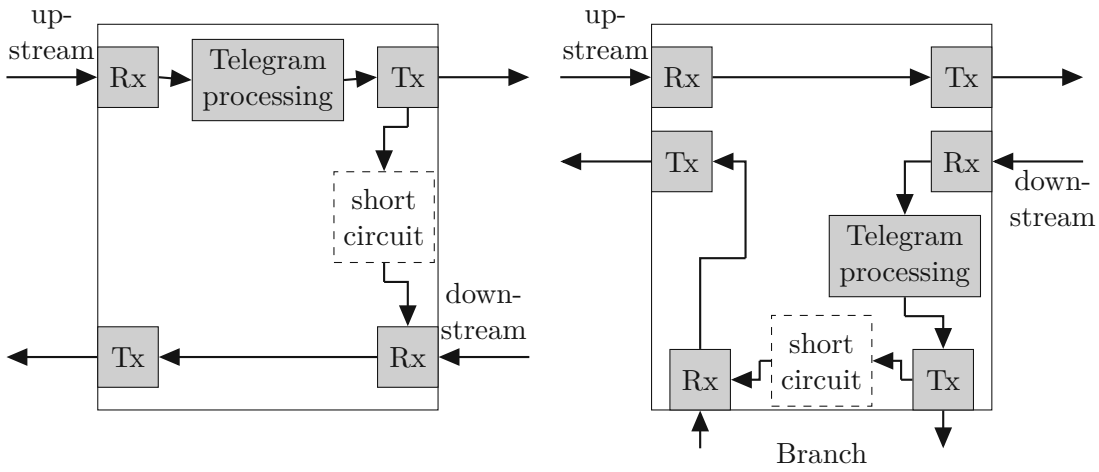


Figure 2.6: EtherCAT topology with branches [WB05]

The data exchange takes place in such a way that in the upstream the master sends a frame into the ring and each of the following slaves receives the frame, processes and forwards the possibly modified frame to its successor. The last slave of the ring is connected to its physical predecessor, so that the frame in turn in downstream passes through each slave until it finally reaches the master again. Figure 2.7 shows schematically the two types of EtherCAT slaves. It can be seen that in Figure 2.7a no branching is possible and the telegram processing is carried out in the upstream. On the other side in Figure 2.7b, branches are allowed and the telegrams are processed in the downstream. In the last slave of the line or at the end of a branch, the short-circuit block is used to connect the upstream with the downstream, thus creating the ring topology. Due to the full-duplex feature of Ethernet, two nodes need only be connected with one cable. EtherCAT guarantees maximum real-time characteristics with cycle times of up to 0.1 ms with a jitter of less than 1 µs for one hundred slaves in the network [WB05, Dü17].

EtherCAT slaves require special hardware, the so-called Fieldbus Memory Management Unit (FMMU). This unit receives and interprets the telegram. If the slave is addressed and the EtherCAT command is reading, the FMMU copies the telegram section belonging to the slave into the slave's memory or writes it from the slave's memory to the corresponding telegram section, if the EtherCAT command is writing. By implementing the FMMU as



(a) without branches (b) with branches

Figure 2.7: Two different EtherCAT slave types [WB05]

an Application-Specific Integrated Circuit (ASIC), the time for these operations is in the range of nanosecond [WB05].

The EtherCAT telegram which is shown in Figure 2.8 is transmitted as the data content of an Ethernet frame. All data exchange of all participants, whether sending or receiving data, takes place via this telegram. The telegram can contain one or more EtherCAT commands. A command consists of a 10 byte header, n bytes of data and 2 bytes for the Working Counter (WC). It is incremented by 1 by each slave after processing the commands. After the telegram has passed through the ring, the master can check whether all participants have processed the telegram. The FMMU executes the communication function according to the commands. Then the telegram is sent to the next participant [WB05].

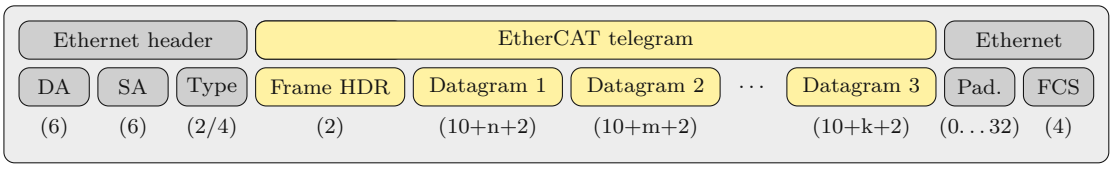


Figure 2.8: EtherCAT telegram structure [WB05]

Figure 2.8 shows the structure of a EtherCAT telegram, if only EtherCAT commands in a single subnet are used. There is also the possibility of extending this telegram with an Internet Protocol (IP) and User Datagram Protocol (UDP) header. This allows routing to other connected subnets. However, the telegram runtime increases due to the routing. Transmission Control Protocol (TCP) packets can be transmitted by fragmenting and pasting them after the EtherCAT commands into a telegram. Accordingly, they must be

reassembled at the receiver by the EtherCAT software [Dü17].

2.5 Measuring device – TESTRANO 600

This chapter explains the measuring device, which was used for this thesis and shows a possible experimental setup for the purpose of this diploma thesis. Furthermore, the software architecture of the embedded system is described and how this thesis can be integrated in the already running system without disturbing it.

For this diploma thesis the already existing measuring system TESTRANO 600[2] from OMICRON electronics GmbH[3] is the basis for the implementation and the subsequent evaluation of the results. Figure 2.9 shows the used device from two different angles. The TESTRANO 600 is a three-phase test system for power transformers. With this device it is possible to perform most common standard electrical testing on power transformers for routine and diagnostic testing on site or during Factory Acceptance Tests (FAT). The TESTRANO 600 has three big advantages. First it is worldwide the first portable power transformer test system. Second it is possible to perform various tests on power transformers without re-connecting and last but not least, the three integrated amplifiers can speed up testing, as all three phases are measured simultaneously. Therefore, the testing time is only one third compared to testing times with conventional single-phase test systems.

Figure 2.9: TESTRANO 600 [OMI19]

This measuring system is based on EtherCAT. This means that all control values such as settings or individual samples for the actuators or amplifiers as well as the measured values are exchanged with one or more specific EtherCAT slaves via this protocol. Between the slaves mediates a so-called EtherCAT master, which runs on an Embedded Linux. The lowest applicative part is often referred to as Soft-Digital Signal Processor (DSP). This is called soft because it is not a true bare-metal DSP. It's just a software component that's part of a Device Abstraction Layer (DAL). The DAL is used to control the

[2] https://www.omicronenergy.com/en/products/testrano-600/
[3] https://www.omicronenergy.com/en/

EtherCAT master running on the same Linux system. The Soft-DSP enables largely freely configurable signal routing between the individual data fields of the EtherCAT frames and performs simple calculations that can be performed on an embedded processor with limited Floating-Point Unit (FPU) or for embedded systems typically medium-speed Arithmetic Logic Unit (ALU).

The amplifier and measuring technology are represented and made accessible by the EtherCAT slaves. During a measurement these slaves communicate with the embedded system with 10 kHz hard real-time. These slaves are usually associated with an asset in order to stimulate them and measure the results mainly from these stimuli. Besides voltages and currents sometimes movements and very often time response are measured. For this diploma thesis the most relevant assets are the instrument transformers (CTs and VTs).

Figure 2.10 shows roughly a possible experimental setup to test and use the result of this diploma thesis. On the bottom left the measuring device TESTRANO 600 with the three external interfaces is shown. The interface marked with (1) are the inputs and outputs to perform a measuring on the asset. In our case we just need the outputs to stimulate the CT or VT because the measured results will be received via the SV streams. The device has also two Ethernet interfaces. The *eth1* is used to control and communicate with the device via the software Primary Test Manager (PTM)[4]. Furthermore, the user of the device can download log files or upgrade the device over this interface and the help of the website. Along that this interface can be used to receive PTP packets from the PTP Master which is shown in Figure 2.10. The second Ethernet interface *eth2* is used to receive the SV streams. A closer look at the structure of the device and the software architecture will be described in more detail in the following sections. At the top left of Figure 2.10 the asset to be tested is illustrated. In our case it could be a VT or CT. The asset has a primary side (2) which is connected to the measuring device. The secondary side (3) is still shown as an external analog interface in Figure 2.10, but at NCIT this interface is digital and can only be accessed via SV streams. Top right a MU is displayed. The MU may be part of a transformer in the field or a separate unit. For better illustration, a separate unit was shown in Figure 2.10.

[4] https://www.omicronenergy.com/en/products/primary-test-manager-ptm/

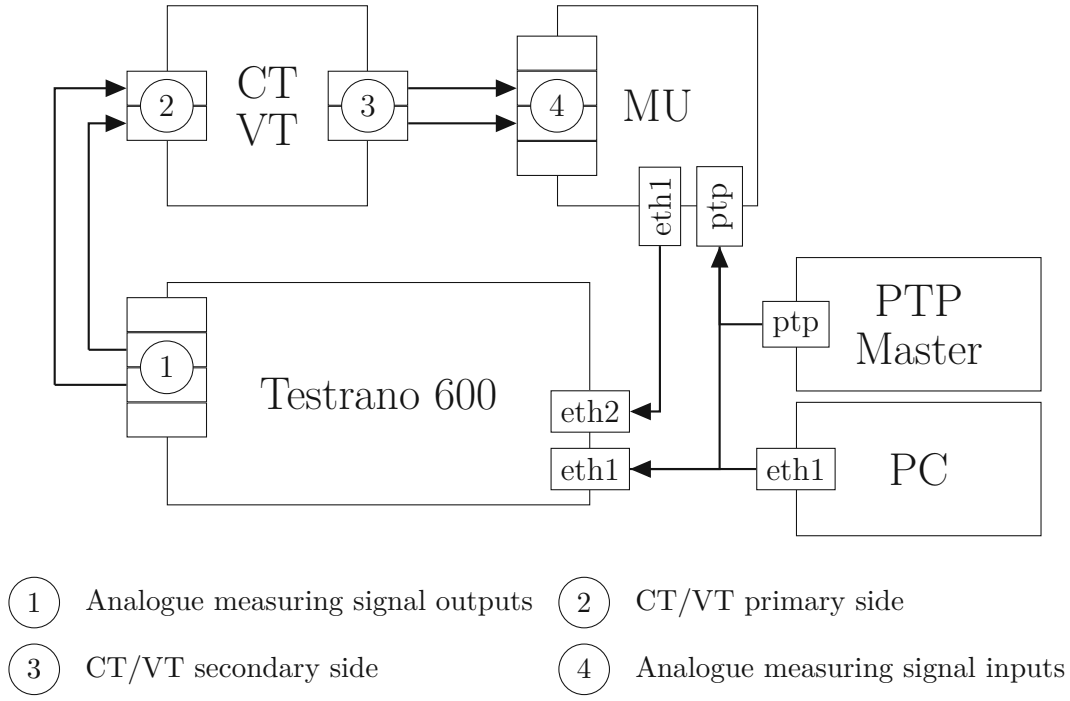


Figure 2.10: Possible experimental setup for this diploma thesis

Figure 2.11 presents the core modules of the TESTRANO 600 measuring device. As illustrated, the device can be split up into three main units. On the top left the Amplifier Manager (AmpMan) is shown. It has twelve independent voltage channels measuring in range of −300 V to 300 V and three current channels measuring in range of −40 A to 40 A. The inputs and outputs of this module are used to stimulate the device to be tested and measuring the response of the asset. The AmpMan communicates with the system via EtherCAT. On the top right the display module is shown. It has a 10.6″ integrated multi-touch display and can be used to configure the measurement and also to show the results directly on the device. For this diploma thesis the most important part is shown at the bottom of Figure 2.11 and is labeled with *System*. In addition to the two external Ethernet interfaces already described in Figure 2.10, the system board has another internal Ethernet interface used for data exchange with the display module. The rest of the system will be split up in smaller parts and discussed in more details.

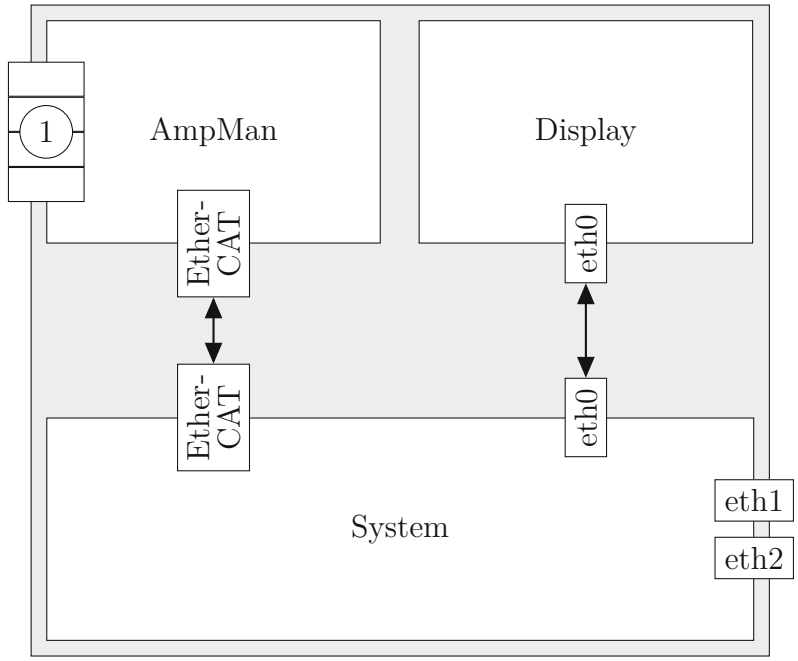


Figure 2.11: Core modules of the TESTRANO 600

Figure 2.12 shows the core modules of the system that are relevant to understand this diploma thesis. The EtherCAT master top left scans the bus regularly and detects all connected slaves. In our case it recognizes the AmpMan and communicates with it. It also exchanges data with the DAL via IPC and the Master Asynchronous Input Output (MAIO) driver. This driver sends commands to the MAIO-Field Programmable Gate Array (FPGA) which controls the oscillator tuning circuit on bottom left so that we have a very precise clock and the real-time of 100 µs can be realized. The internal data from the system to the display and vise-versa are sent and received over the internal socket and proceed in the general thread. As already described in Figure 2.10 the PTP frames are received at the *eth1* and parsed to the PTP socket which provides the necessary data to the PTP thread. This data are handled by the thread and then sent to the MAIO-FPGA, which then synchronizes its internal reference clock with the external PTP master. The Real-Time State Engine (RSE) Soft-DSP is executed on an isolated core of the CPU, so that the system can fulfill the real-time conditions and can not be disturbed by interrupts. The RSE synchronizes itself each time before reading and writing a frame with the MAIO-FPGA via Direct Memory Access (DMA). Furthermore, a red marked area is highlighted in Figure 2.12, which represents the expansion of the system through this diploma thesis. The exact structure and function of this area is described in detail in Section 4.2.

Figure 2.12: Core modules of the system

CHAPTER 3

Related Work

In the paper [CY10] of Chao Cai and Yuping Lu they present a scheme how a Sampling Rate Synchronization (SRS) can be implemented in a protection IED. The realized the SRS based on oversampling interpolator and quasi-continuous Finite Impulse Response (FIR) interpolator. An illustration of the basic structure diagram of the SRS scheme is shown in Figure 3.1.

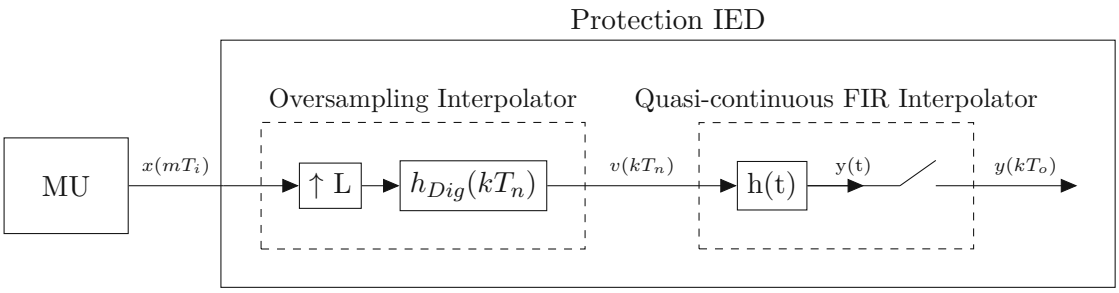


Figure 3.1: Illustration of SRS system in a protection IED [CY10]

The MU generates a signal $x(mT_i)$ with a sampling rate of f_i. Inside the protection IED the received signal $x(mT_i)$ gets interpolated by a 1-to-L interpolator. The result is a highly oversampled signal $v(kT_n)$ with a sampling rate of $L\dot{f}_i$. This signal is connected to a quasi-continuous FIR interpolator $h(t)$ which calculates arbitrary time instants with the sampling rate of the protection algorithm f_o.

For the oversampling interpolator they used the optimum equiripple method to design the filter, as this method gives the best approximation of an ideal filter. To reduce the computation effort a poly-phase network was realized. The detail about that can be found in the corresponding paper [CY10].

For the quasi-continuous FIR interpolator the number of calculations and therefore the order of the filter should be as low as possible but with decreasing the order of the filter the difficulty to design such a filter increases. Therefore, it is important to find a compromise where the amount of calculations as well as the complexity of the design is considered.

For the evaluation they compared the proposed SRS scheme with a zero-order-holder followed by the oversampling interpolator and the cubic spline interpolation. For the test setup with an oversampling factor $L = 20$ the presented SRS scheme has an instantaneous current error within 0.5%. With the zero-order-holder following the oversampling interpolator with the same oversampling factor L the instantaneous current error is larger than the previous one. Some experiments showed that an oversampling factor $L \geq 40$ is needed to get an error within 0.5%. The instantaneous current error of the cubic spline interpolation is comparable to the one of the proposed SRS scheme. The exact results and the diagrams of the measurement can be found in the paper [CY10]. With the cubic spline interpolation the needed SRS could also be achieved but in this paper the authors wanted to use the full high speed computing capacity of the DSP [CY10].

For our thesis the SRS will be done in a Soft-DSP and therefore we can not use this special computing capacity of a DSP. Because of that, this proposed SRS scheme of Chao Cai and Yuping Lu is not really suitable for our problem.

R. Tao, B. Jiang and C. Wang described in their paper [TJW11] the problem that in a SAS often Electronic Instrument Transformerss (EITs) are provided from different manufacturers and therefore it can happen that they use different sampling rates because of different standards of the manufacturers. Due to a probable mismatch between the EITs and the IED the data cannot directly be transmitted. In this paper [TJW11] they showed how this problem can be solved in a full digitalization process using interpolation and decimation methods.

Therefore, they presented the Big MU which does the sampling rate conversion and afterwards a data synchronization. An illustration of this Big MU can be found in Figure 3.2. The MUs receive data from the EITs and send them to the next module which includes an interpolation method, a Digital Filter (DF) and a decimation algorithm. The resulting asynchronous data are then forwarded to the data synchronization module to get synchronized data again which then can be transmitted to the IEDs.

Figure 3.3 shows the steps needed to convert a sampling rate. The signals $x_1(n)$ and $x_2(n)$ are two data channels from MU_1 and MU_2 with the sampling frequencies f_1 and f_2 where $f_1 \neq f_2$. The two data channels get interpolated by factor L_1 and L_2 and then low-pass filtered by h(n). After that the data get decimated by a factor of M and the result is the data sequence y(m) with the sampling frequency of $f_p = \frac{L_1 \cdot f_1}{M}$

For the data synchronization they used the quadratic interpolation method. The simulation of the Big MU showed good results and can be found in the paper [TJW11]. For this thesis we could see the general idea of resampling data and how this could be done.

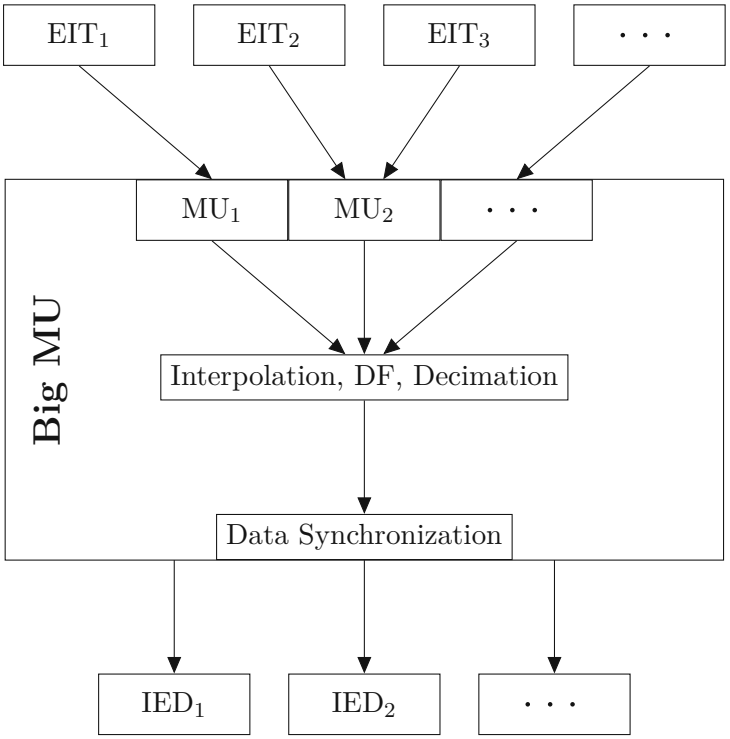


Figure 3.2: A new digital system with EITs, Big MU and IEDs [TJW11]

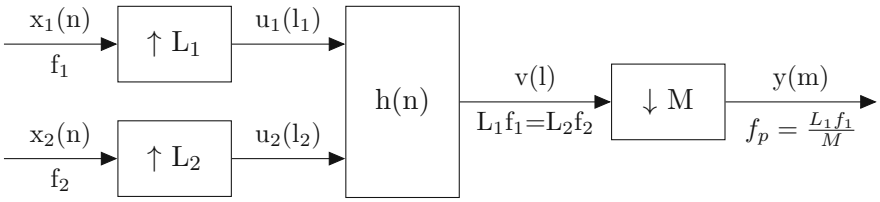


Figure 3.3: Sampling rate conversion [TJW11]

Unfortunately they did not describe what kind of interpolation method they used and there are also no results to compare with other methods. Nevertheless, we will consider the general approach of resampling signal in a full digitalization process. Another point to consider is that in this paper [TJW11] different asynchronous signals were processed in the MU which is not the case for our thesis.

In the paper [PT16] of A. Patki and G. Thiagarajan they described a modified version of the Akima algorithm which is often used in computer graphics. The baseline algorithm uses a third order piece-wise polynomial function Equation 3.1 to fit an interval of output points and has to satisfy some conditions so that there is a smooth transition from the last point to the new calculated point. The mathematical derivation can be found in the paper [PT16].

$$f(t) = p_0 + p_1(t - t_3) + p_2(t - t_3)^2 + p_3(t - t_3)^3 \tag{3.1}$$

The big advantage of the Akima algorithm is that it is only dependent on the output sampling rate. For example the spline interpolation is depending on the input and output sampling rate. In their paper they showed how to modify this algorithm to reduce algorithmic complexity and so improving the performance. Beside just using four points instead of six points to calculate the new point they did some further simplifications. They also used a different Equation 3.2 to describe the polynomial curve. For details about the simplifications and the mathematical ideas can be found in the original paper [PT16].

$$f(t) = p_0 + p_1(t - t_3) + p_2(t - t_3)(t - t_4) + p_3(t - t_3)^2(t - t_4) \tag{3.2}$$

To improve the quality of the results an Augmented Least-Squares Solver (ALSS) was used. The block diagram of this solver is shown in Figure 3.4. In the first step a coarse resampling algorithm e.g. the modified Akima algorithm is used to get the first estimated resampled signal y_m at the synchronous time points $m{\cdot}T_s$. In the next step a Least-Squares Solver (LSS) is used to improve the accuracy of y_m and get as a result values z_m at time points $m{\cdot}T_s$ with higher precision. The LSS algorithm needs some output points form the first step and some of its own previous output values with a high precision.

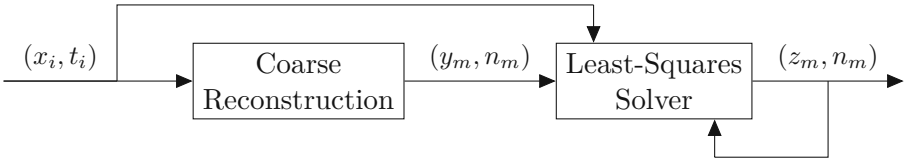


Figure 3.4: Block diagram of the ALSS resampler [PT16]

The simulation in the paper [PT16] showed that the modified Akima algorithm achieved better results than the original one but is marginally worse than the cubic interpolation. From the complexity point of view the modified Akima algorithm has the lowest one and the cubic interpolation the highest one. The proposed ALSS algorithm is more

complex than the modified Akima algorithm and has a similar performance than the cubic interpolation algorithm. So the conclusion is that the modified Akima scheme is suitable for low complexity and medium performance applications and the ALSS scheme should be used for medium complexity and high performance applications.

For our thesis we would need to go for the ALSS algorithm but we can not really benefit because this algorithm is optimized for asynchronous to synchronous resampling and we just need a resampling algorithm and therefore the cubic interpolation can also be simplified as we know that we have already synchronous samples at the input. So the results are anyways almost the same and the complexity of the cubic interpolation can be reduced with the information that we have equidistant sample points.

Yeying Chen, Enrico Mohns, Michael Seckelmann and Soeren de Rose described in their paper a precise amplitude and phase determination for calibrating SV instruments by using resampling algorithms [CMSdR20]. For the resampling process they used a modified sinc interpolation in the time domain. Furthermore, a phase correction is described to preserve the phase accuracy to the 1 PPS signal.

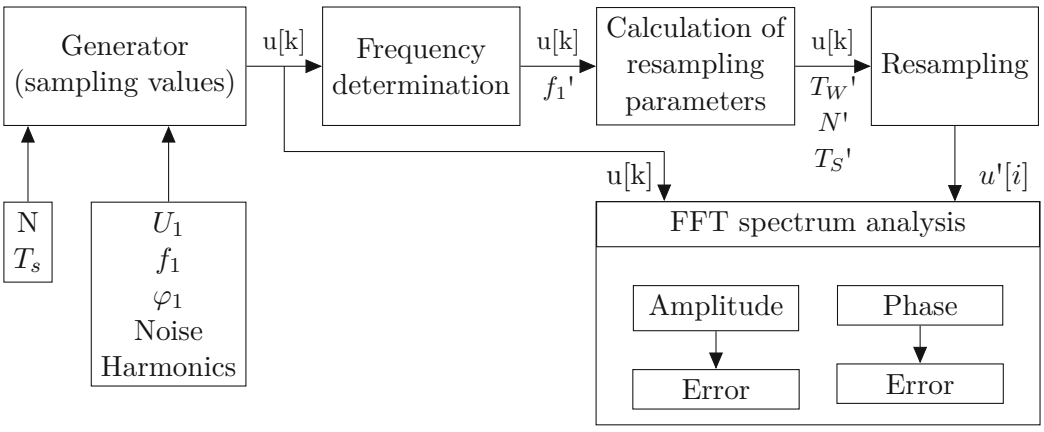


Figure 3.5: Program structure of the resampling process [CMSdR20]

Figure 3.5 shows the program structure of the used resampling process. The discrete signal samples $u[k]$ covers three components which are a fundamental sinusoidal signal, a white noise signal and a series of sinusoidal signals to simulate several harmonics. The mathematical equation to generate this signal is shown in Equation 3.3 where T_S is the sampling time of the process. U_1 and U_m are the Root Mean Square (RMS) voltages of the fundamental sinusoidal signal and the m^{th} harmonic signal. f_1 and f_m are the frequencies of the signals and φ_1 and φ_m are the phase angles. The white noise signal is generated with a white noise random generator with a rectangular distribution.

$$u[k] = \sqrt{2}\cdot U_1 \cdot sin(2\pi \cdot f_1 \cdot k \cdot T_S + \varphi_1) + u_{Noise} + \sum_{m>1} \sqrt{2}\cdot U_m \cdot sin(2\pi \cdot f_m \cdot k \cdot T_S + \varphi_m) \quad (3.3)$$

The Frequency determination block in Figure 3.5 uses the IEEE-STD-1057 four-parameter sine wave fit algorithm to determine the fundamental frequency f_1. In the next block the required parameters for the resampling process are calculated. The Resampling block contains of three different resampling algorithms to compare the results of the different methods. It is possible to choose between a quadratic, cubic or the modified sinc interpolation. The result is a discrete signal $u'[i]$ with a sampling rate of f_S' which is higher than the original sampling rate.

The Fast Fourier Transform (FFT) spectrum analysis evaluates the quality of the positive relative amplitude errors $|\frac{\Delta U}{U}|$ and the positive phase errors $|\Delta\varphi|$. The detail description about the modified sinc interpolation and the phase synchronization can be found in the paper [CMSdR20].

For the first simulation a pure sinusoidal signal was used without noise and the harmonic series. The modified sinc interpolation had the lowest relative amplitude errors $|\frac{\Delta U}{U}|$ followed by the cubic interpolation and the quadratic interpolation. The phase errors of the modified sinc and cubic interpolation was almost the same, only the quadratic interpolation provided a bigger phase error. For the computation time the quadratic interpolation was the fastest algorithm and needed about 8 ms at a sampling rate of 20 kHz. The cubic interpolation was slightly slower and needed about 9 ms. By far the slowest algorithm was the modified sinc interpolation which needed about 250 ms. The simulation also showed that fluctuating parameters have no significant effect on the accuracy of the resampling algorithms. Furthermore, the simulation pointed out that with increasing the sampling rate f_s also the relative amplitude errors have a growing trend for all resampling algorithms.

As a next step the simulation was done with single harmonic interaction. The results showed that the relative amplitude errors for the fundamental as well as for the harmonic signal have almost the same errors for all algorithms. All of them showed an increasing trend for a higher order of the harmonics. And the order of algorithms based on the relative amplitude errors is the same as for the fundamental signal. After that the simulation was done with a harmonic series and the results were slightly higher than the resampling with a single harmonic. As a final step they simulate the modified sinc interpolation and checked the results when also white noise is involved. These results and all others can be found in more detail in the paper [CMSdR20].

For this thesis this paper showed, that for low frequencies which is the case in a substation the cubic interpolation has almost as good results as the proposed modified sinc interpolation. The main reason for not taking the modified sinc interpolation for this thesis is the computation time which is much higher. In our case the resampling algorithm has to run on an embedded hard real-time system and therefore we have to take the compromise for a slightly higher relative amplitude error but much lower computation time.

CHAPTER 4

Implementation

4.1 Resample

As already mentioned in Section 1.1, our hard real-time measurement system runs at a system frequency of 10 kHz, which can not be changed. It was also noted in Section 1.1 that the SVs are received at 4000 Hz or 4800 Hz depending on the network frequency. Since the system frequency and the frequency of the SVs do not match, the SVs must be resampled so that the SVs are also available to the measuring system at a frequency of 10 kHz.

Since 10 kHz is neither a multiple of 4000 Hz nor 4800 Hz, we first need to upsample to a higher frequency than 10 kHz and then downsample again, so that we finally reach the desired 10 kHz. Now the question arises, on which frequency we first have to upsample. The simplest solution is simply to multiply the two frequencies, but this does not always give the best solution, which means that unnecessary calculations are done and more time is needed. The better approach is to calculate the least common frequency. A possible implementation is shown in Algorithm 4.1. By calling the function `LCM` with the two frequencies the function will return the resulting upsampling frequency.

Algorithm 4.1: Calculation least common frequency

```
1 function GCD(a, b):                          ▷ Greatest Common Divisor
2 |   return ((b == 0) ? a : GCD(b, a  mod b))
3 end
4 function LCM(a, b):                          ▷ Least Common Multiple
5 |   return (a · b) / GCD(a, b)
6 end
```

In Subsection 4.1.1, the theoretical part of upsampling is first explained and a suitable algorithm is selected. Subsequently, the selected algorithm is described in Subsection 4.1.3 and in Subsection 4.1.4 the downsampling part is described.

4.1.1 Upsampling

There are several options for upsampling. All algorithms have their advantages and disadvantages, so we have to choose a variant that will give the best results for the specific application. In the following points a few algorithms are presented and finally an algorithm is selected. Of course there are further algorithms but they are not all described in this diploma thesis.

Previous Neighbor Interpolation Previous Neighbor Interpolation is better known as zero-order hold. With this method, the received SV is kept constant until the next SV. It is a very simple method and does not require any calculations as the extra points between the two SVs will be the same as the last received SV. This method is used in Analog-to-Digital Converters (ADCs). An illustration of the method is shown in Figure 4.1.

Next Neighbor Interpolation This algorithm is very similar to the previously described algorithm. The only difference is, that this method does not keep the last received SV constant, but the next received SV. This means that two SVs must first be received before the interpolation can begin. Thus, the extra points between SV n and $n+1$ all get the value of SV $n+1$. Figure 4.2 shows an example of the algorithm.

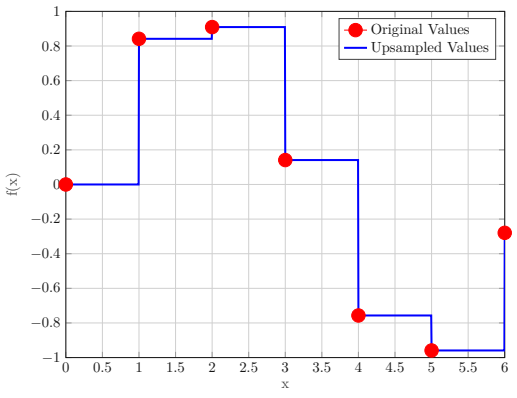


Figure 4.1: Previous Neighbor

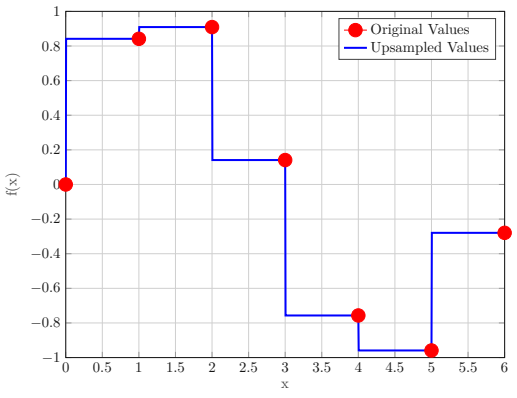


Figure 4.2: Next Neighbor

Nearest Neighbor Interpolation This algorithm is a combination of the Previous Neighbor and Next Neighbor Interpolation, as illustrated in Figure 4.3. With this method, not all additional points between the two SVs receive the same value. Any additional points before the time center of the two SVs will receive the value

according to the Previous Neighbor Interpolation and all values after the time center will get the value according to the Next Neighbor Interpolation. Even with this method, no major calculations are needed. It only needs to be determined the time, that is exactly in the middle of the two SVs.

Linear Interpolation Another simple method is the Linear Interpolation. The extra points between two SVs (x_n, y_n) and (x_{n+1}, y_{n+1}) can be easily calculated using the linear relationship. Mathematically, the equation is established that the straight-line between the two SVs must have the same slope and ordinal distance as the straight-line between an additional point and a SV. After the mathematical transformation, Equation 4.1 is obtained, which can be used to calculate the additional points. An example of a Linear Interpolation is shown in Figure 4.4.

$$y = y_n + (y_{n+1} - y_n) \cdot \frac{x - x_n}{x_{n+1} - x_n} \tag{4.1}$$

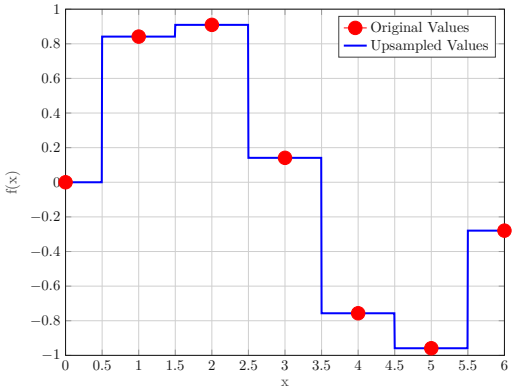


Figure 4.3: Nearest Neighbor

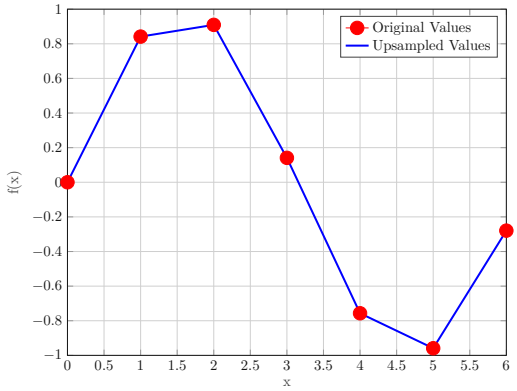


Figure 4.4: Linear Interpolation

Polynomial Interpolation This method is a generalization of linear interpolation. Linear interpolation uses a linear function and this method replaces the interpolant with a higher order polynomial. The order of the polynomial depends on the number of points used for the calculation. In general, we can say that for n points, there exists exactly one polynomial with the highest degree of $n-1$ passing through all n points. So for the example shown in Figure 4.5 a polynomial of degree six was used, because all seven given points were used for the calculation.

Piecewise Cubic Spline Interpolation As described above, Linear Interpolation uses a linear function for each of the intervals $[x_n, x_{n+1}]$. For the Piecewise Cubic Spline Interpolation low order polynomials are used for each interval. The polynomial parts are selected so, that they fit together smoothly. Figure 4.6 shows an example of this interpolation method. Already anticipated, this algorithm provided the best results for this particular application and therefore this algorithm and mathematical derivation is discussed and explained in more detail in Subsection 4.1.3.

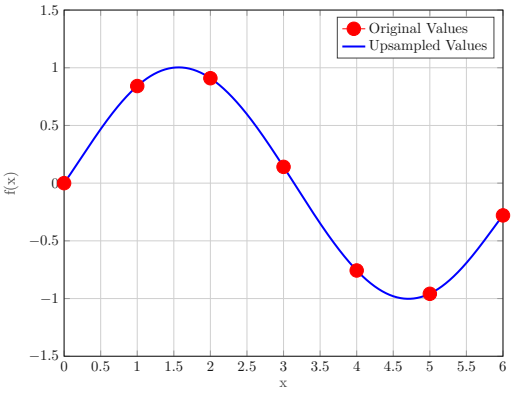


Figure 4.5: Polynomial Interpolation

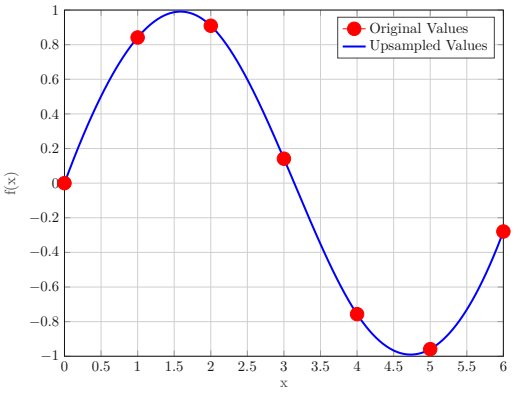


Figure 4.6: Piecewise Cubic Spline

Shape-Preserving Piecewise Cubic Interpolation Like the cubic spline interpolation this method uses also the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP). The only difference is, that the spline interpolation uses other conditions for the slopes at the given points. This algorithm tries to find a shape-preserving interpolant that is visually pleasing. This is mainly achieved by determining the slopes at the given points, so that the function values do not overshoot the data values locally [Mol04]. An example of this method is shown in Figure 4.7.

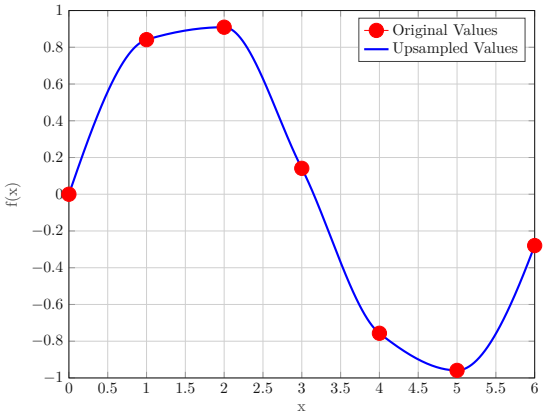


Figure 4.7: Shape-Preserving Piecewise Cubic

4.1.2 Selection of a suitable algorithm

To find a suitable algorithm for our problem we will compare the advantages and disadvantages of the different methods and choose the algorithm based on this comparison. As a first step, we will start with the most simple algorithms. The big advantage of the Previous Neighbor, Next Neighbor and Nearest Neighbor Interpolation is that the extra points can be easily determined with less or none calculation. Furthermore, the

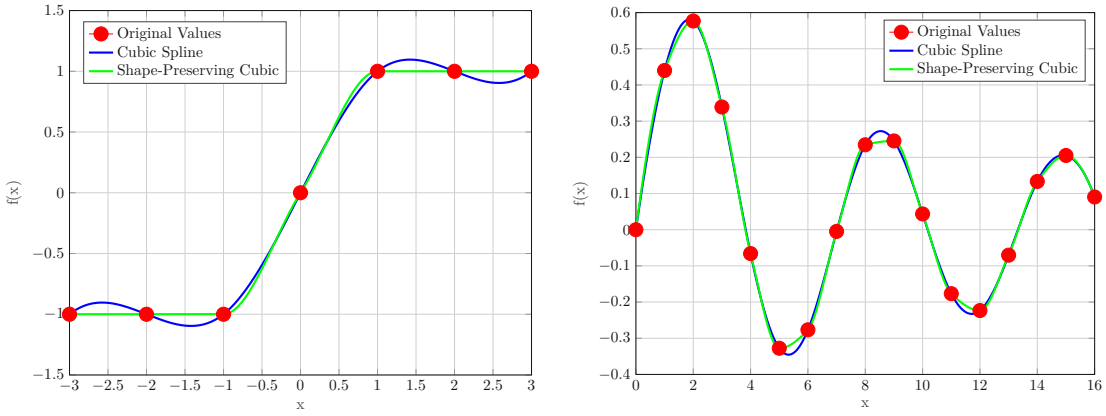
interpolation is really fast and does not use a lot of memory. If we have continuous signals, the results of these simple methods are unfortunately not perfect as clearly visible in the Figures 4.1 – 4.3. Depending on the sample rate of the received values the interpolation error can get huge. That's mainly the reason why these algorithms were not selected for this diploma thesis.

The Linear Interpolation is also a very simple algorithm. The interpolated values can be calculated very easy and quick but on the other hand the interpolant is not differentiable at the received SV and therefore the result is not very smooth. Furthermore, the results are not very precise as shown in Figure 4.4.

The Polynomial Interpolation uses a polynomial of degree at most $n-1$ passing through all n SVs. Therefore, the interpolant is infinitely differentiable. It is therefore very easy to see that polynomial interpolation avoids the disadvantages of linear interpolation. Figure 4.5 shows, that the Polynomial Interpolation delivers a very good result with a very small interpolation error. Nevertheless, this method also has a significant disadvantage. The calculation of the interpolation polynomial is very computationally intensive and takes a certain amount of time because some points are needed to get a good result. Therefore, this method was also not selected.

Finally, the two PCHIP methods are compared. The Shape-Preserving Piecewise Cubic Interpolation is a local method and uses only two points to do the calculations. The first derivative of the interpolant is continuous, but the second derivative is probably not continuous. There may be jumps to the given points. On the other hand the Spline Interpolation is a global method which means, that the algorithm uses also points which are more far away to do the calculations. The sensitivity to data far away is less than to nearby data. Both algorithms have their advantages and disadvantages depending on the signal to interpolate [Mol04]. An illustration of this dependency can be seen in Figure 4.8. On the left side in Figure 4.8a we have a step function. The illustration shows clearly that the Cubic Interpolation has an oscillating result in the area of the step. The Shape-Preserving Piecewise Cubic Interpolation is much better suited for this type of signal. On the other side in Figure 4.8b a Bessel function of the first kind is shown. This part of the image shows, that the Cubic Interpolation gives a better result in the area of the peaks and is therefore better suited, if a continuous oscillating signal has to be interpolated. From the complexity and the computational effort, the two methods are about the same.

Since we only deal with sinus signals in our field of application, the Spline Interpolation was chosen. In the next section the mathematical derivation will be explained and at the end the algorithm will be described.



(a) Step function

(b) Bessel function of the first kind

Figure 4.8: Comparison Spline Interpolation and Shape-Preserving Interpolation

4.1.3 Spline interpolation

A cubic spline is a smooth curve that passes through given points in the coordinate system and has a minimal total curvature. Each section is defined by a cubic parabola

$$a_i \cdot x^3 + b_i \cdot x^2 + c_i \cdot x + d_i \tag{4.2}$$

with suitable coefficients a_i, b_i, c_i and d_i. Smooth curve means in the mathematical sense that the curve should be twice continuously differentiable [Pla06].

In Figure 4.9 an example of a spline interpolation is shown where the red dots are the given points and the black curve is the resulting interpolated function. The thin dashed lines are the cubic parabolas (Equation 4.3) between two given points. How to obtain the resulting function will be explained below.

Suppose we have $n+1$ points $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \cdots, \begin{pmatrix} x_n \\ y_n \end{pmatrix}$ where $x_0 < x_1 < \cdots < x_n$. To obtain the coefficients, it is convenient to define the n parts of the spline with

$$S_i(x) = a_i \cdot (x - x_i)^3 + b_i \cdot (x - x_i)^2 + c_i \cdot (x - x_i) + d_i, \tag{4.3}$$

where $0 \leq i < n$ and $S_i(x)$ represents the curve between the points $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$ and $\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix}$. Since the sections seamlessly merge at the given points [CK12]

$$S_{i-1}(x_i) = S_i(x_i) = y_i \tag{4.4}$$

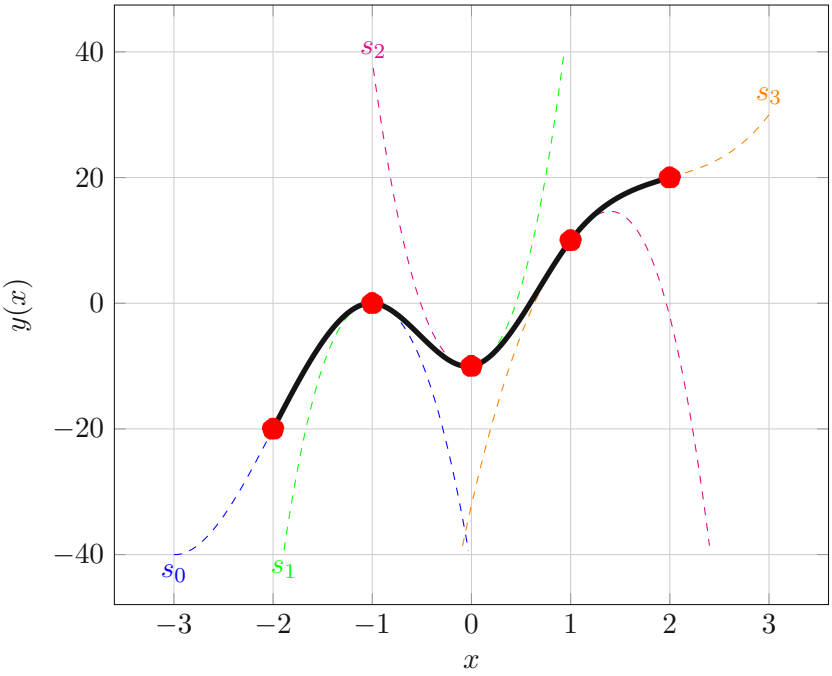


Figure 4.9: Example of a spline interpolation [Pla06]

for $1 < i \leq n$. From Equation 4.3 we get that $S_i(x_i) = d_i$ and combined with Equation 4.4 we can see, that the coefficient $d_i = y_i$. Furthermore, we can see from Equation 4.3 and 4.4 that

$$a_{i-1} \cdot (x_i - x_{i-1})^3 + b_{i-1} \cdot (x_i - x_{i-1})^2 + c_{i-1} \cdot (x_i - x_{i-1}) + d_{i-1} = d_i \tag{4.5}$$

Furthermore, in all given points, the adjoining sub-curves have the same tangents, so we have [CK12]

$$S'_{i-1}(x_i) = S'_i(x_i) \tag{4.6}$$

where the derivative is

$$S'_i(x) = 3 \cdot a_i \cdot (x - x_i)^2 + 2 \cdot b_i \cdot (x - x_i) + c_i. \tag{4.7}$$

Thus, we get from Equation 4.6 that

$$3 \cdot a_{i-1} \cdot (x_i - x_{i-1})^2 + 2 \cdot b_{i-1} \cdot (x_i - x_{i-1}) + c_{i-1} = c_i. \tag{4.8}$$

Finally, the abutting partial curves also have identical curvatures in all given points, so [CK12]

$$S''_{i-1}(x_i) = S''_i(x_i), \tag{4.9}$$

where the second derivative is

$$S''_i(x) = 6 \cdot a_i \cdot (x - x_i) + 2 \cdot b_i. \tag{4.10}$$

If we insert this equation into Equation 4.9, we get

$$6 \cdot a_{i-1} \cdot (x_i - x_{i-1}) + 2 \cdot b_{i-1} = 2 \cdot b_i \tag{4.11}$$

By transforming this equation we get

$$a_{i-1} = \frac{b_i - b_{i-1}}{3 \cdot (x_i - x_{i-1})}. \tag{4.12}$$

In the next step we insert Equation 4.12 in Equation 4.8 and get

$$(b_i + b_{i-1}) \cdot (x_i - x_{i-1}) + c_{i-1} = c_i. \tag{4.13}$$

After inserting Equation 4.12 in Equation 4.5 and transforming the resulting equation we get

$$c_{i-1} = \frac{d_i - d_{i-1}}{x_i - x_{i-1}} - \frac{(b_i - b_{i-1}) \cdot (x_i - x_{i-1})}{3} - b_{i-1} \cdot (x_i - x_{i-1}). \tag{4.14}$$

If we do an index shift at the last equation, we obtain

$$c_i = \frac{d_{i+1} - d_i}{x_{i+1} - x_i} - \frac{(b_{i+1} - b_i) \cdot (x_{i+1} - x_i)}{3} - b_i \cdot (x_{i+1} - x_i). \tag{4.15}$$

Now we put the Equations 4.14 and 4.15 into Equation 4.13. After some mathematical transformations, we then get the following equation.

$$(x_i - x_{i-1}) \cdot b_{i-1} + 2 \cdot (x_{i+1} - x_{i-1}) \cdot b_i + (x_{i+1} - x_i) \cdot b_{i+1} = 3 \cdot (\frac{d_{i+1} - d_i}{x_{i+1} - x_i} - \frac{d_i - d_{i-1}}{x_i - x_{i-1}}) \tag{4.16}$$

Because of $d_i = y_i$, the right side of Equation 4.16 is known for $0 < i < n$. Thus, the b_i for $0 < i < n$ can be obtained from all equations of Equation 4.16 using a linear system of Equations 4.17.

$$
A = \begin{pmatrix}
2\cdot(x_2-x_0) & x_2-x_1 & 0 & \dots & 0 & 0 & 0 \\
x_2-x_1 & 2\cdot(x_3-x_1) & x_3-x_2 & \dots & 0 & 0 & 0 \\
0 & x_3-x_2 & 2\cdot(x_4-x_2) & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & x_{n-2}-x_{n-3} & 2\cdot(x_{n-1}-x_{n-3}) & x_{n-1}-x_{n-2} \\
0 & 0 & 0 & 0 & x_{n-1}-x_{n-2} & 2\cdot(x_n-x_{n-2})
\end{pmatrix}
$$

$$
A \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix} = \begin{pmatrix}
3\cdot\left(\frac{y_2-y_1}{x_2-x_1} - \frac{y_1-y_0}{x_1-x_0}\right) \\
3\cdot\left(\frac{y_3-y_2}{x_3-x_2} - \frac{y_2-y_1}{x_2-x_1}\right) \\
3\cdot\left(\frac{y_4-y_3}{x_4-x_3} - \frac{y_3-y_2}{x_3-x_2}\right) \\
\dots \\
3\cdot\left(\frac{y_{n-1}-y_{n-2}}{x_{n-1}-x_{n-2}} - \frac{y_{n-2}-y_{n-3}}{x_{n-2}-x_{n-3}}\right) \\
3\cdot\left(\frac{y_n-y_{n-1}}{x_n-x_{n-1}} - \frac{y_{n-1}-y_{n-2}}{x_{n-1}-x_{n-2}}\right)
\end{pmatrix} \tag{4.17}
$$

After solving the linear system of equations, we get the coefficients b_1 to b_{n-1}. b_0 and b_n are the half curvatures in the first and the last point, which can be specified freely and are assumed here to be 0 [CK12]. The calculated coefficients b_i are now inserted into Equation 4.12 and 4.14, and thus we obtain the values for the coefficients a_i and c_i. An example implementation to solve this mathematical problem is shown in algorithm 4.2.

This algorithm is based on the one mentioned in [Sto12]. For this application the algorithm was modified due to the fact, that in this case the given points are equidistant and this simplified the implementation.

Algorithm 4.2: Computing natural cubic splines [Sto12]

Input: a set of $n+1$ points $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$, $sampleRate$

Output: n 5-tuples for each polynomial piece S

$$S_i(x) = a_i \cdot (x - x_i)^3 + b_i \cdot (x - x_i)^2 + c_i \cdot (x - x_i) + d_i$$

Create five arrays a, c, d, α and μ of size n
Create three arrays b, l and z of size $n+1$
$l_0 = 1,\ \mu_0 = 0,\ z_0 = 0$
$l_n = 1,\ b_n = 0,\ z_n = 0$
for $i \leftarrow 0$ **to** $n-1$ **do**
    $\alpha_i = 3 \cdot sampleRate \cdot (y_{i+1} - 2 \cdot y_i + y_{i-1})$
    $l_i = \frac{4 - \mu_{i-1}}{sampleRate}$
    $\mu_i = \frac{1}{l_i \cdot sampleRate}$
    $z_i = \frac{\alpha_i - \frac{z_{i-1}}{sampleRate}}{l_i}$
end
for $i \leftarrow n-1$ **to** 0 **do**
    $d_i = y_i$
    $b_i = z_i - \mu_i \cdot b_{i+1}$
    $c_i = (y_{i+1} - y_i) \cdot sampleRate - \frac{b_{i+1} + 2 \cdot b_i}{3 \cdot sampleRate}$
    $a_i = \frac{(b_{i+1} - b_i) \cdot sampleRate}{3}$
end

4.1.4 Downsampling/Decimation

By upsampling we have extended the received SVs to the least common frequency. Now we have to reduce the frequency again so that we get to the desired 10 kHz. We can distinguish between two main methods. One is the downsampling and the other one is the decimation. The downsampling method simply uses one value and the next M (Equation 4.18) values are not used or removed and the $M+1$ value is then used again.

$$M = \frac{inputrate}{outputrate} - 1 \tag{4.18}$$

There are several algorithms available for decimation. As part of this diploma thesis, three different methods were simulated with the help of MATLAB®[1] and then compared

[1] https://www.mathworks.com/products/matlab.html

with simple downsampling which will be presented in Section 4.1.4.1.

IIR Chebyshev Low Pass Filter Type 1 Infinite Impulse Response (IIR) filters are specific type of filters in signal processing. These are Linear Shift Invariant (LSI) filters. Figure 4.10 shows some frequency responses of Chebyshev low-pass filters with different orders. In this example the normalized cutoff frequency is 0.4. Figure 4.10 illustrates that with higher order, the transition from the passband to the stopband becomes steeper, but the order of the IIR filter should not be increased too far, otherwise the results will be unreliable.

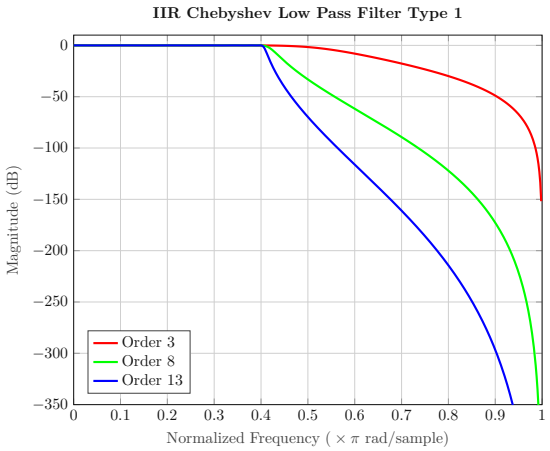


Figure 4.10: Chebyshev low pass filter with different orders

FIR Filter FIR filters have an impulse response which settles to zero in finite time. Compared to the IIR filters which may have internal feedback and probably continue to a response indefinitely. Figure 4.11 shows some frequency responses of FIR low-pass filters with different orders. Like for the IIR filters before we have a normalized cutoff frequency of 0.4. Also for this type, the filter becomes steeper with a higher filter order, but has some ripple at the end of the frequency response. The order cannot be increased arbitrarily for the FIR filter, but it is possible to connect two FIR filters one behind the other and thus achieve a higher order.

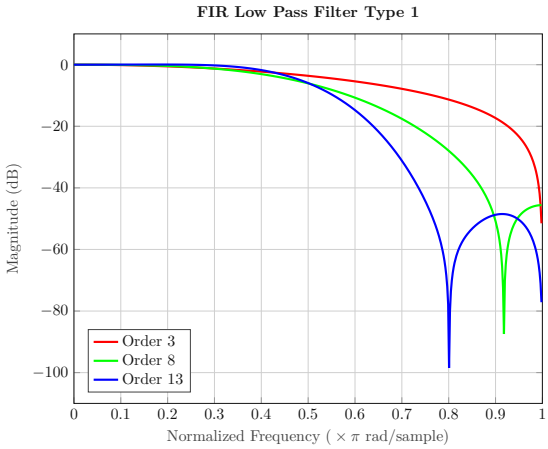


Figure 4.11: FIR low pass filter with different orders

Average Filter The average filter is a special implementation of a low pass algorithm. The output of this filter is calculated from a finite number of input samples. This filter is used to smooth data that carry high frequency distortions and afterwards a downsampling is performed. Figure 4.12 shows the different functions which are used to perform the digital convolution with the input samples. The order n describes the number of input samples which are used to calculate the current sample.

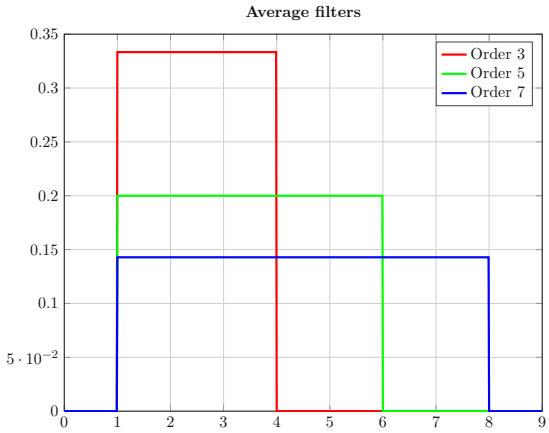


Figure 4.12: Average filter with different orders

4.1.4.1 Comparison of downsampling and decimation

Figure 4.13 shows the input signal which was sampled with 4000 Hz and was used to compare the different methods. Before we can compare the different algorithms, we first have to upsample the signal to 20 kHz with the spline interpolation which was already

described in Section 4.1.3. Now we have the same starting point for all algorithms and we can compare the individual methods.

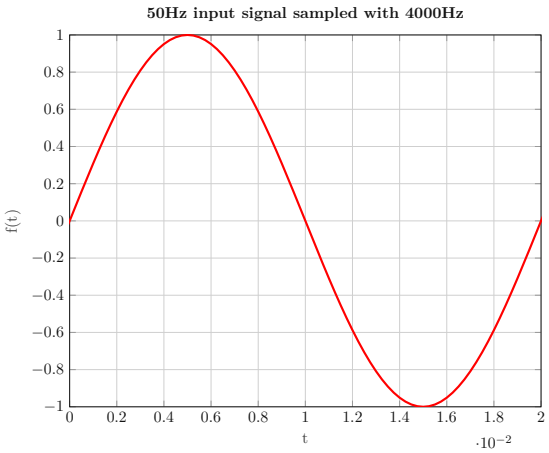
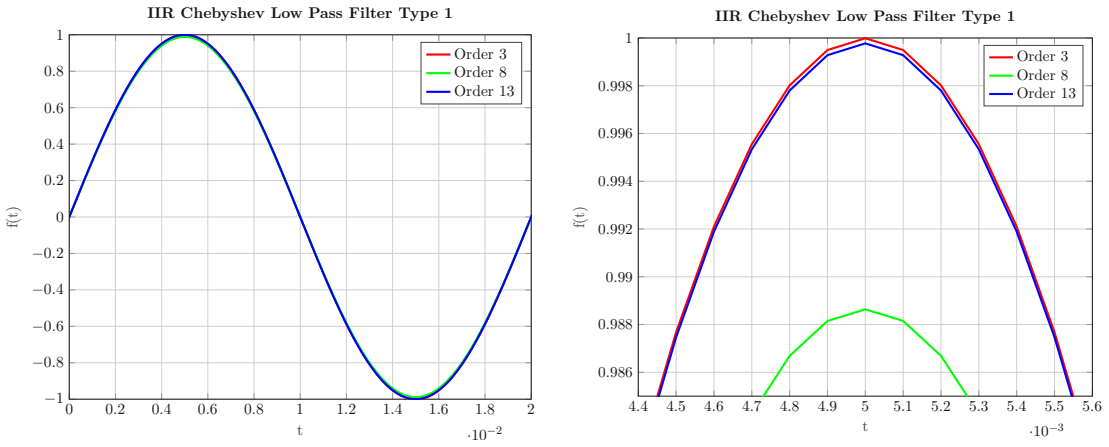


Figure 4.13: 50Hz input signal sampled with 4000Hz

First we will show the results of the IIR Chebyshev Low Pass Filter Type 1 with different orders. The results can be found in Figure 4.14. As clearly visible, the order can have a big impact on the result. For this specific example the IIR filter with order 8 provided the worst result in the area of the peak (Figure 4.14b) and in general the IIR filter with order 3 had the best result and will therefore be compared with the best results of the other methods.



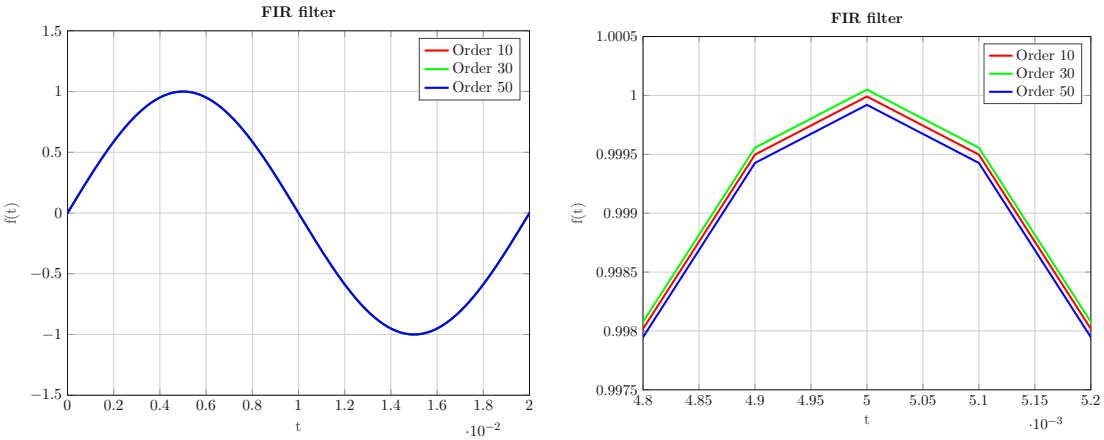
(a) One period of the signal

(b) Detailed view of the peak

Figure 4.14: IIR Chebyshev Low Pass Filter Type 1 with different orders

As a next step we will use the FIR filter with different Hamming window sizes and we will present the results in Figure 4.15. Like before we get slightly different results for different Hamming window sizes. Therefore, a detailed view is shown in Figure 4.15b

again to better see the differences. In this illustration the sinus signal doesn't look that smooth because of the high zoom factor which can be derived from the step size of the x-axis. For further comparison we will take the FIR filter with a Hamming window size of 10 because for this simulation we got the best result.



(a) One period of the signal

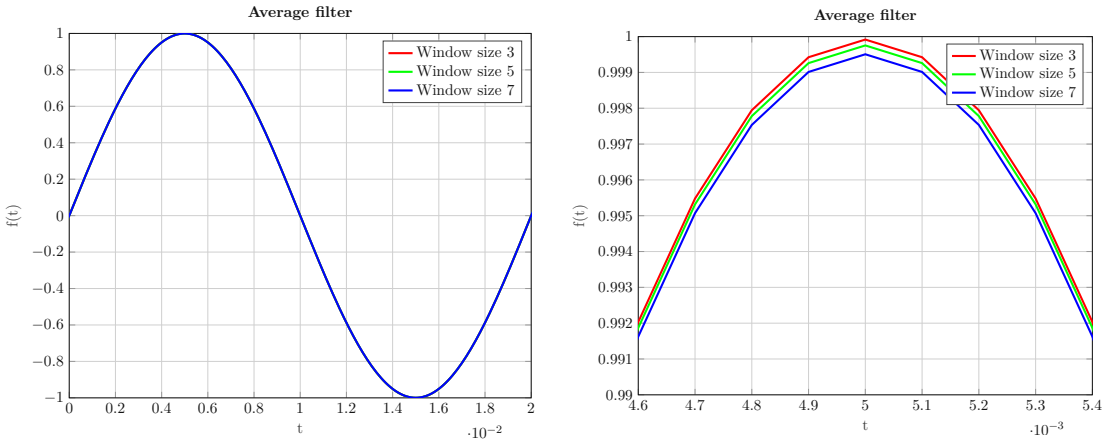
(b) Detailed view of the peak

Figure 4.15: FIR Filter with different Hamming window orders

Next we will look at the results of the average filter with different kernel sizes and present them in Figure 4.16. As with the methods already shown, there is again a figure with an entire period (Figure 4.16a) and a detailed view (Figure 4.16b) to better recognize the differences between the various filter configurations. Figure 4.16b shows, that the differences between the average filters with different kernel sizes are not that big and they provide almost the same results. For the further comparison we will go with the average filter with a kernel window size of 3 because for this use case we got the best results at the simulation.

In order to be able to complete the comparison we now have to compare the individual methods with each other so that we can see which method is the best for our simulation. As already mentioned before we will compare the IIR low pass filter with order 3, the FIR low pass filter with order 10 and the average filter with a kernel window size of 3 with the simple downsampling. So all of these methods used the sinus signal which was upsampled with help of the spline interpolation to 20 kHz and applied there decimation calculation on this signal to get the resulting 10 kHz sinus signal. To see the error of the up- and downsampling we calculate the error with the squares of the deviation. The mathematical formula can be found at Equation 4.19. The y_{ideal} at this equation is a perfect sinus signal which was sampled with 10 kHz and is used to evaluate the results. This error includes also the error of the upsampling but as all of them used the same signal the resulting error shows the quality of the downsampling.

$$e_{methode}(x) = (y_{ideal}(x) - y_{methode}(x))^2 \tag{4.19}$$



(a) One period of the signal

(b) Detailed view of the peak

Figure 4.16: Average filter with different kernel size windows

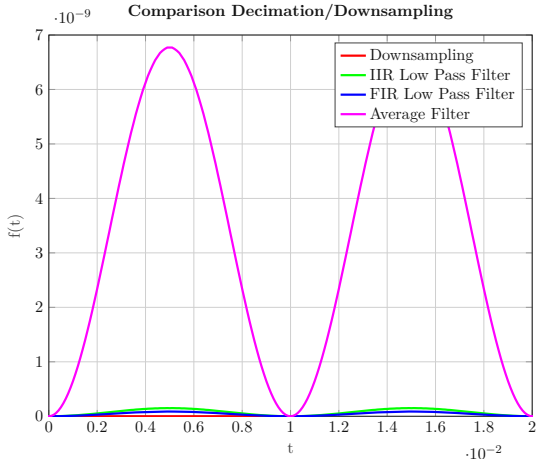
The results of this comparison can be found in Figure 4.17. Figure 4.17a shows very clearly that the average filter delivers the worst result in this comparison. To better see the errors of the other methods the average filter is removed and the reduced comparison can be found at Figure 4.17b. For this simulation we can see that IIR low pass filter has the worst results after the average filter and on the second place we can see the FIR low pass filter. As it is almost impossible to see the error of the downsampling, a detailed view of the error of the downsampling of a whole period is shown in Figure 4.17c.

Furthermore, another detailed view of only one half of a period is shown in Figure 4.17d. In this illustration it is noticeable, that every 5th point of the error is exactly 0. This is the case because during upsampling we calculate four more points between two sampled points to get to the 20 kHz. When we perform the downsampling we ignore every second point and so we have the exact value at each 5th downsampled point. This behaviour is illustrated in Table 4.1. After this comparison we decided to just do the simple downsampling because it delivers the best results and is the method with the least computation effort.

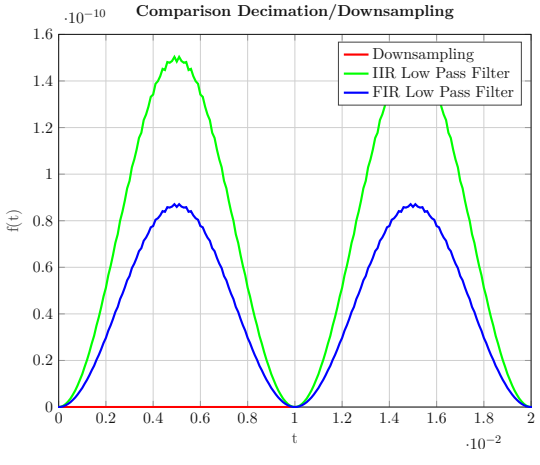
S	C	C	C	C	S	C	C	C	C	S
D		D		D		D		D		D

S. . . Sampled value
C. . . Calculated value
D. . . Downsampled value

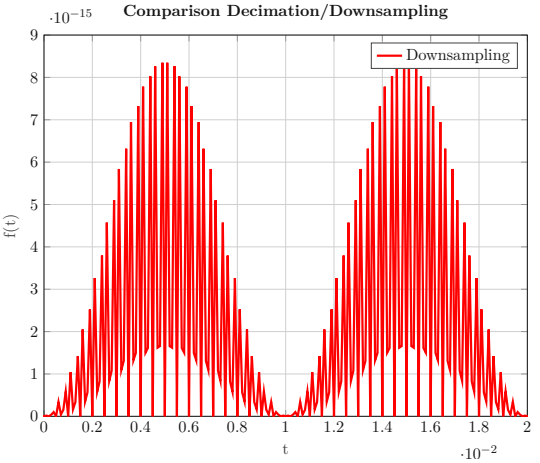
Table 4.1: Illustration of up- and downsampling



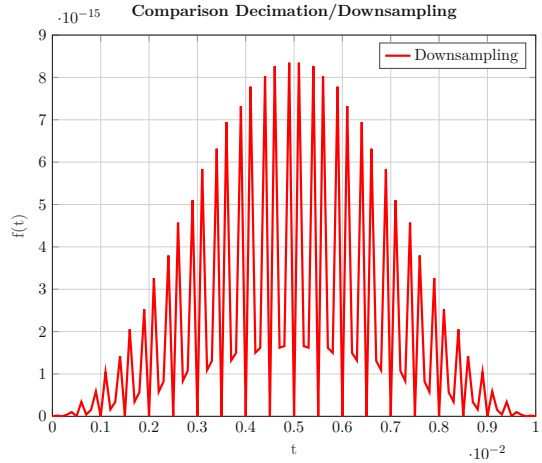
(a) Comparison of all 4 methods



(b) Comparison without average filter



(c) Detail view of downsampling (full period)



(d) Detail view of downsampling (half period)

Figure 4.17: Comparison of the different decimation methods and simply downsampling

4.2 Bridge

In this section the implementation of the so-called bridge will be explained. In Figure 4.18 illustrates how the SVs are read, processed and provided to the RSE. As demonstrated in Figure 4.18, the system consists of two Soft-DSPs. Both of them consist of several Soft-DSP executables which are executed in a periodic time.

To follow the data flow we will start to explain from the right side of Figure 4.18. There a *Frame-Buffer* is illustrated which is filled by the SV socket as we already saw in Figure 2.12. This socket parses the received SV frames and provides the relevant data to *Frame-Buffer*. This buffer contains SVs with the specified sample rate of 4000 Hz in a substation with 50 Hz operating frequency or 4800 Hz in a 60 Hz environment.

The first Soft-DSP executable (9) of the Digital Substation (DS) Soft-DSP reads only one record of the *Frame-Buffer*. The next executable (8) is the loop executable and can contain several executables. For example, it could be a recorder (7) which writes the read data to a persistent storage. This can be very useful for debugging purposes. Another executable in the loop is designed to do the upsampling calculation which was described in Subsection 4.1.3. Another one (6) is used to write the upsampled values to the *Sample-Buffer*. These steps are executed over and over again and in case we have a SV stream with a sample rate of 4000 Hz we have a cycle time of ≈250 µs. The cycle time is not exactly 250 µs because this thread has to share a core of the CPU with other threads. This could cause, that in some cases the Soft-DSP needs a bit more time than the planed 250 µs. However, this is not a major problem as this thread is not part of the hard real-time condition of the system and the *Frame-Buffer* can compensate such issues.

The *Sample-Buffer* holds the upsampled values and has a sample rate of 20 kHz if the *Frame-Buffer* is filled with 4000 samples per second or 120 kHz if the *Frame-Buffer* is filled with 4800 samples per second. This sample rate can be calculated with the least common frequency which was already described with the Algorithm 4.1. The 10 kHz sample rate of the RSE is fixed and cannot be changed so the resulting sample rate of the *Sample-Buffer* is just depending on the SV frame rate on the input. A very important point in regard to this buffer is that data is written and read in real-time and therefore we cannot use any kind of mutexes or locks. Otherwise, we could be blocked and cannot reach the 100 µs of our real-time system. Therefore, a smart ring buffer is used which does not need any mutexes at all but can still recognize if data is overwritten or no data is available. Both of these cases should not happen in a normal run because we upsampled the data to a common frequency but if something goes wrong an error message can be shown.

On the other side of Figure 4.18 we have the RSE Soft-DSP. As a first step (1) of every cycle, the EtherCAT frame is written. As next step the loop executable (2), as we already saw in the DS Soft-DSP, is executed. This also consists of several executables. On this side we have an executable which reads a specific number of upsampled values. How many of these values are read in one cycle is as well depending on the SV frame rate.

Since the sampling rate is higher than the 10 kHz of the RSE we have to remove some read data as described already in Subsection 4.1.4. Now we have a sample rate of 10 000 samples per second and can be used for calculation purposes in other executables of the RSE Soft-DSP. On this side of the bridge we have again a recorder in place so that we can write the up- and then downsampled values to a file and compare them with the simulation. As a last step of every cycle, the EtherCAT frame is read. Such a cycle has to be exactly 100 µs. As shown in Figure 4.18 we have quite some limited time to do the calculations and therefore the up- and downsampling is simplified as much as possible so that the newly added bridge does not disturb the already running system.

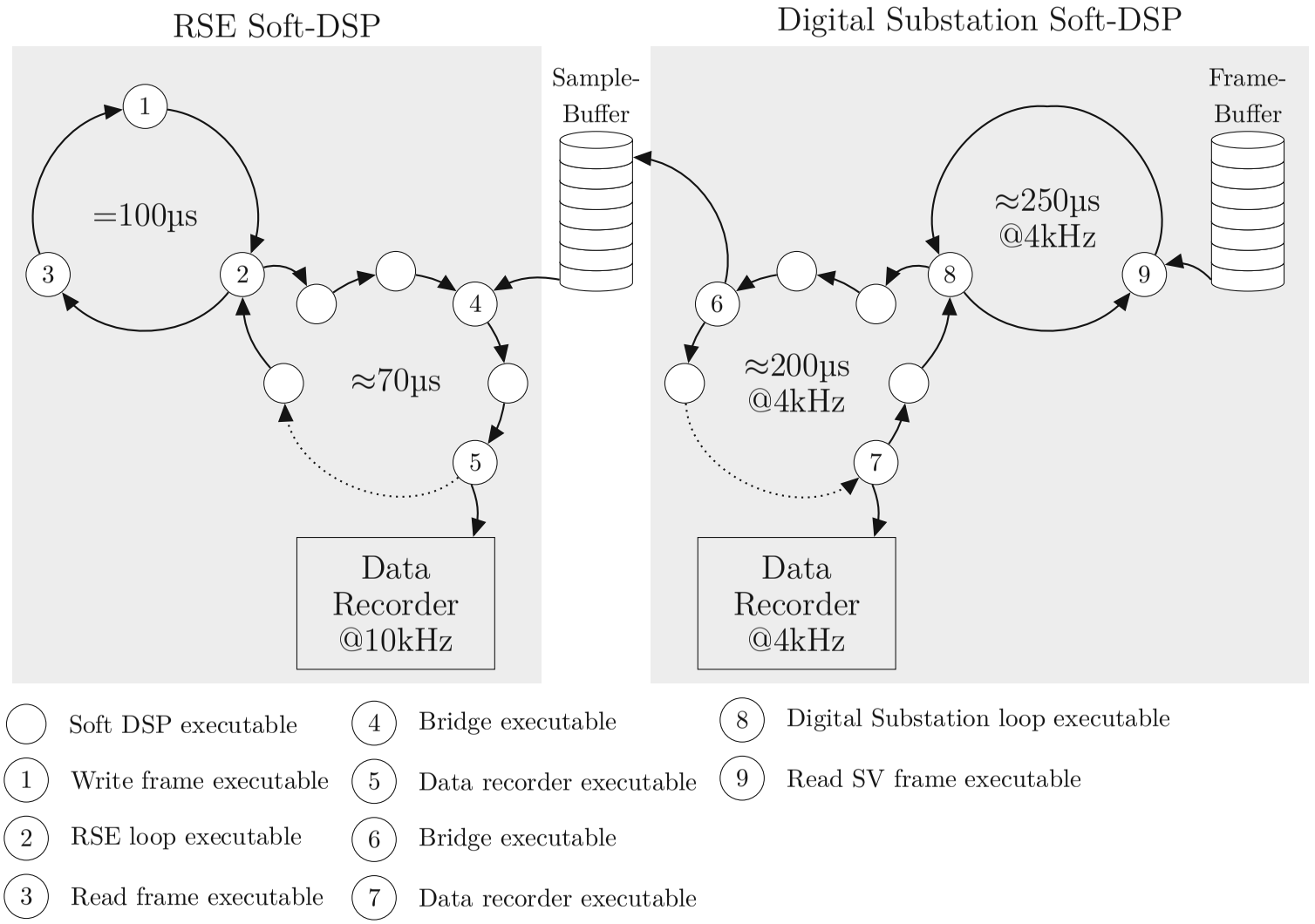


Figure 4.18: Interaction between Digital Substation Soft-DSP and Real-Time State Engine Soft-DSP

CHAPTER 5

Evaluation

In this chapter the theoretically discussed approaches are tested and evaluated on the real device. The basic idea is to create an evaluation setup, where we can create three analog signals with different amplitudes which are connected to a MU. Three different amplitudes are used to better distinguish the different channels when we analyse and plot the data. This MU will then perform the analog to digital conversion and will provide the SVs over the Ethernet interface. These packets will then be forwarded over LAN to the measuring device where the resampling process will be performed.

During this evaluation we will log the whole network traffic, so that we can compare the values which were sent form the MU and what we interpreted in our software. Based on this data, we will analyse the behavior of the packets over time. In particular, we will evaluate the time difference between the packets as real-time performance is a major criterion for this work.

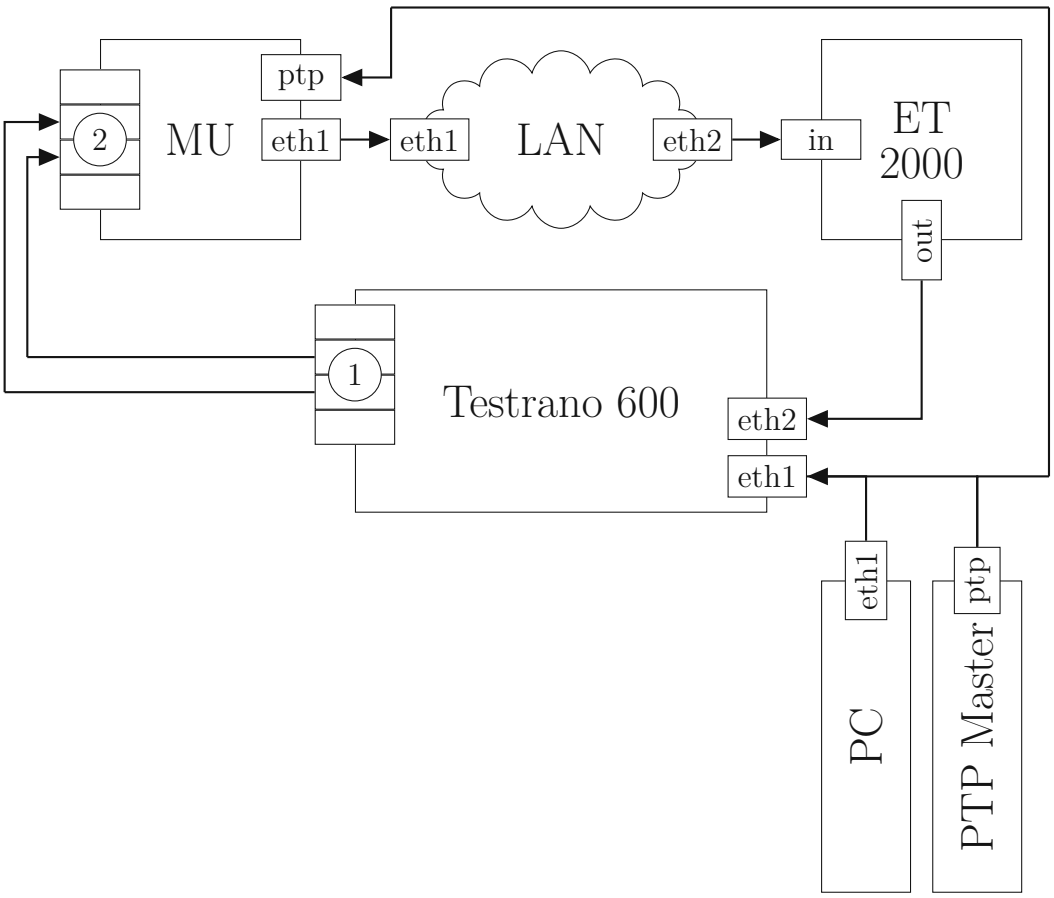
Furthermore, we have also the possibility to log the values of the output signals and the input signals before resampling as well as after resampling. This allows us to evaluate the correctness of our *Bridge*. In addition to this correctness, we also must check that this additional *Bridge* has no negative influence on the already existing hard real-time system. Some timing information of the RSE Soft-DSP thread and the DS Soft-DSP thread is logged after each execution and can be also analysed. If the real-time condition could not hold during execution we would anyways get an exception and the execution would stop.

5.1 Evaluation setup

Figure 5.1 shows the evaluation setup which was used to check the implementation on the running system. As illustrated in Figure 5.1, we simplified the setup a bit compared to the one shown in Figure 2.10. This decision was taken because for testing the system

it is not relevant if the measured values are received from a real CT, VT or from our measurement system itself.

The basic workflow is to output some sinus signals with the help of the AmpMan of the system. These output signals are connected to the MU which converts the analog signals into SV-streams. These packets are then transferred over the LAN to the ET 2000 which is an industrial Ethernet analyzing tool designed by Beckhoff[1]. This device can be used to add a high precise timestamp to all packets which were received. The resolution of this timestamp is 1 ns and the cycle delay is less than 1 µs, so the influence of the network traffic is very small. After adding this timestamp to the packets, the network traffic is forwarded to the device where the resampling of data starts.



(1) Analogue measuring signal outputs (2) Analogue measuring signal inputs

Figure 5.1: Evaluation setup

[1] https://www.beckhoff.de/

5.2 Network traffic analysis

In the first step of the evaluation we analysed the network traffic to see the timing of the received packets. Therefore, we logged the data with the help of Wireshark[2]. An example output of this logging can be found in Table 5.1. The first column shows the number of received packet and the second column shows the precise timestamp from the ET 2000. The source Media Access Control (MAC) address is the one from the used MU. The destination MAC address is defined by the IEC 61850 protocol and has the format "01:0C:CD:04:xx:xx". The last two bytes of the address can be defined by the user and in our case we set them to "00:00" as shown in Table 5.1. The column protocol shows the Ethernet type and is "0x88BA" which is the type for the SV protocol. The last column shows the Ethernet packet size including the timestamp which was added by the ET 2000 device.

No.	Time [ms]	Source MAC	Destination MAC	Protocol	Length
1	2505292.43	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
2	2505292.68	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
3	2505292.89	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
4	2505293.15	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
5	2505293.48	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
6	2505293.68	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
7	2505293.93	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
8	2505294.14	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
9	2505294.4	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
10	2505294.66	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
11	2505294.93	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
12	2505295.18	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
13	2505295.39	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
14	2505295.65	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132
15	2505295.98	00:25:65:00:3D:3E	01:0C:CD:04:00:00	0x88BA	132

Table 5.1: Sample logging data of Wireshark

An example content of such a packet can be found in Table 5.2. The first part of the packet is the destination MAC address which we already saw in Table 5.1. The next block of the packet is the source MAC address. Also the next two bytes we saw already in the Table 5.1 and is the protocol type. The biggest part of the packet is the SV part. In Figure B.1 we show a detail description how this part is structured and the meaning of all the bytes. The last part is the one which is added by the ET 2000. A detailed description of these bytes can be found in Table C.1.

[2] https://www.wireshark.org/

01	0C	CD	04	00	00	00	25	65	00	3D	3E	88	BA	40	00
00	66	00	00	00	00	60	5C	80	01	01	A2	57	30	55	80
04	34	30	30	30	82	02	07	61	83	04	00	00	00	01	85
01	02	87	40	FF	FF	FF	EF	00	00	00	00	FF	FF	FF	C3
00	00	00	00	FF	FF	FF	F7	00	00	00	00	FF	FF	FF	CC
00	00	00	00	00	00	00	00	00	00	00	00	FF	FF	FF	FD
00	00	00	00	00	00	00	03	00	00	00	00	00	00	00	03
00	00	00	00	01	01	05	10	00	00	80	00	F0	A7	10	4F
47	02	00	00												

Table 5.2: Example content of an Ethernet packet

5.3 Packet timing analysis

After checking the content of the Ethernet packets we decided to use this additional data from the ET 2000 and analyse the timing of the packets. Especially the time difference between two SV packets. With the help of Wireshark[3] we therefore logged about 1 000 000 SV packets which is a bit more than 4 min. The distribution of the time differences can be found in Figure 5.2. As illustrated the time difference of $\approx 60\,\%$ of the received packets were in the range of 235 µs to 265 µs. Furthermore, Figure 5.2 shows that $\approx 20\,\%$ were received even faster and $\approx 10\,\%$ had a time difference of more than 300 µs. Because of these packets which take significantly longer than the expected 250 µs, we decided to buffer a certain number of packets at the *Frame-Buffer* of Figure 4.18 before we start with resampling so that we have at any point of the execution enough elements in the *Frame-Buffer*. If we would run out of elements we would not be able to do the resampling which has the consequence that the *Sample-Buffer* of Figure 4.18 will get empty and the RSE will fail because it expects that in each loop it can read a value from the *Sample-Buffer*. This initial delay is not a big problem, because the delay can be derived from the initial number of elements in the *Frame-Buffer* and is constant for all executions. For future calculation it is important to know this delay because every delay from outputting a signal until receiving the measured value will result in a measurement error.

[3] https://www.wireshark.org/

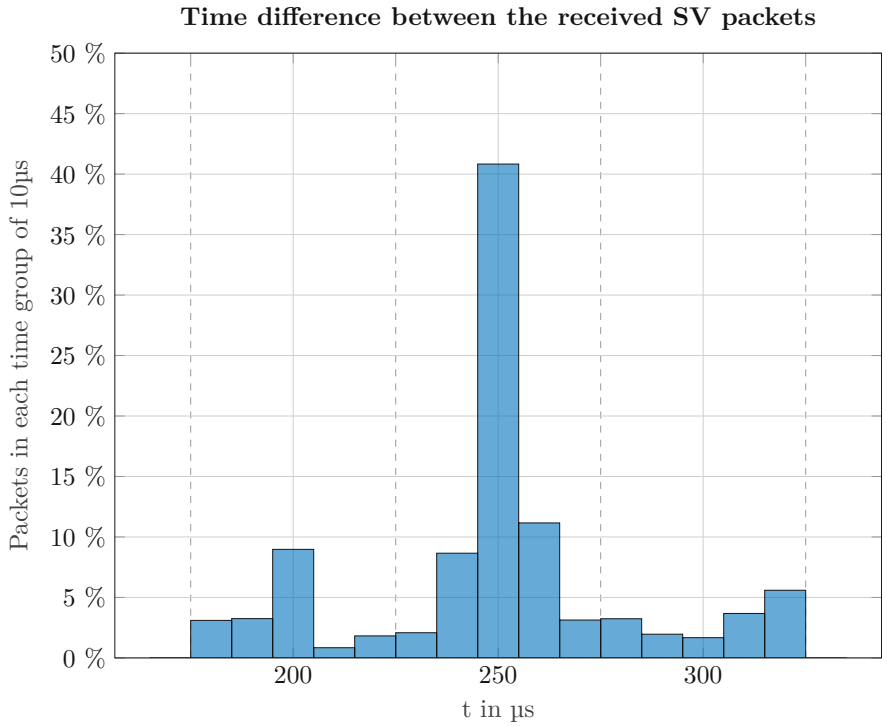
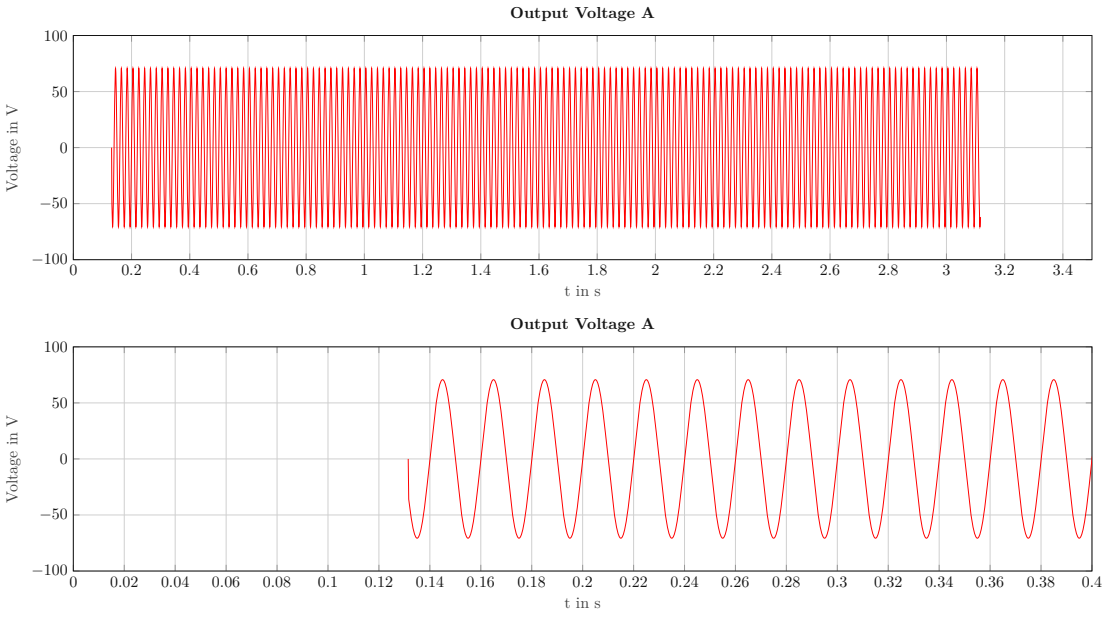


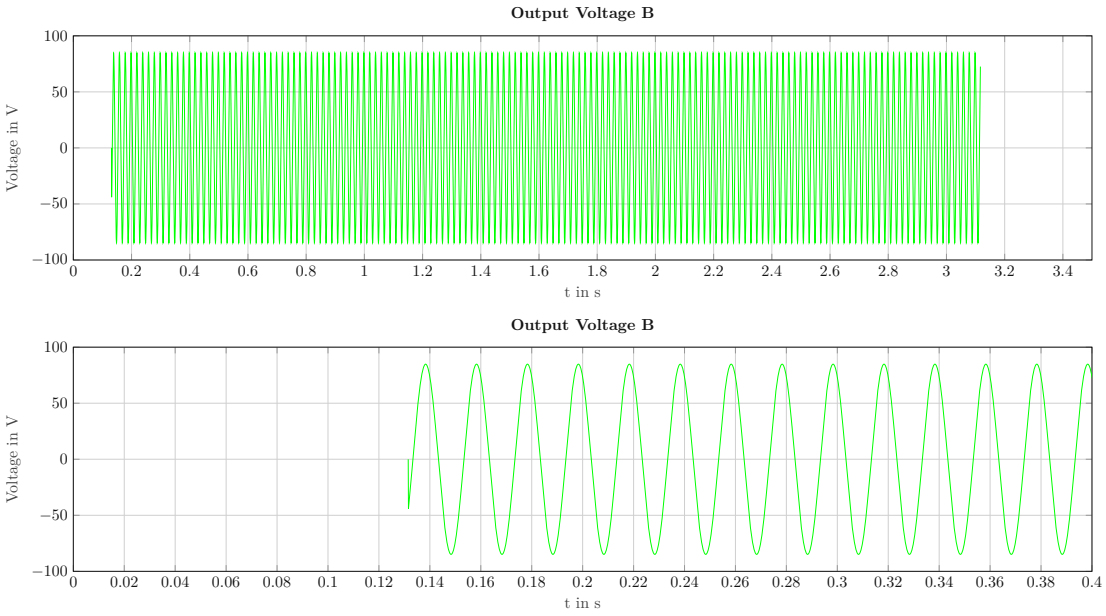
Figure 5.2: Distribution of time differences between received SV packets

5.4 Analysis of the output and input signals

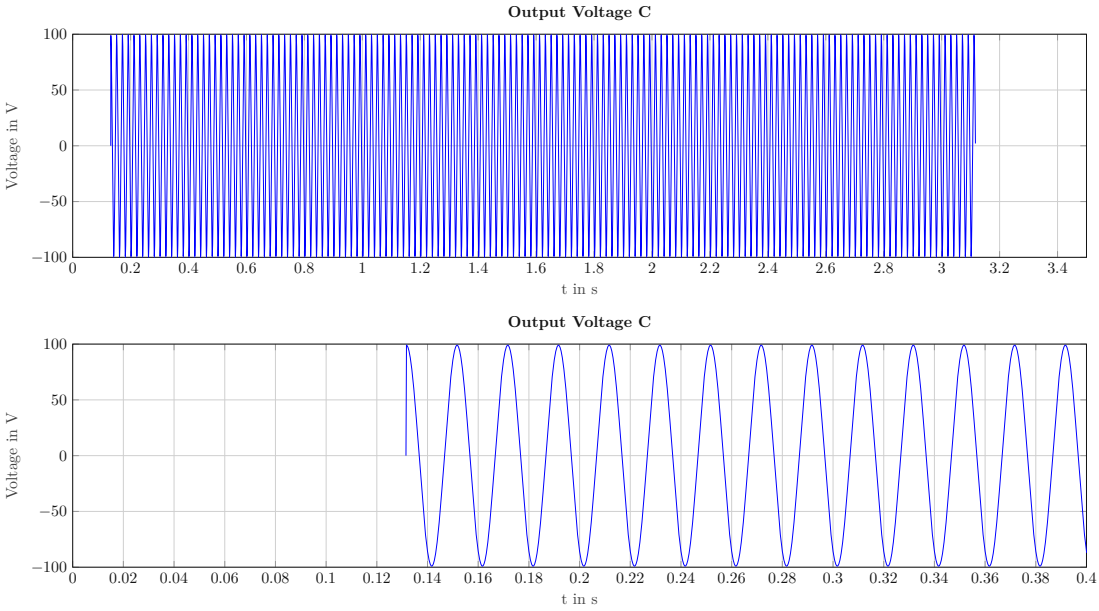
As a next step the output signal of the AmpMan of the measuring device is logged and displayed in Figure 5.3. This signal was calculated by the Soft-DSP and is then transferred to the AmpMan. The logging starts a bit earlier then the calculation of the output signal, therefore there is a small delay of about 130 ms in Figure 5.3. So that the individual three voltages can be better distinguished, different amplitudes were used for the sine signals. For *Output Voltage A* we used a peak amplitude of about 70 V which is shown in Figure 5.3a. The top signal shows a bigger time range and the bottom signals shows only a part of the top signal to better see the signal. For *Output Voltage B* and *Output Voltage C* we used a peak amplitude of 80 V and 100 V as shown in Figure 5.3b and Figure 5.3c. After approximately 3 s the thread of the RSE was stopped and the data were written to the log file. All these output data were recorded at a frequency of 10 kHz with the data recorder executable (5) which was already shown in Figure 4.18.



(a) Real-Time State Engine DSP Output Voltage A



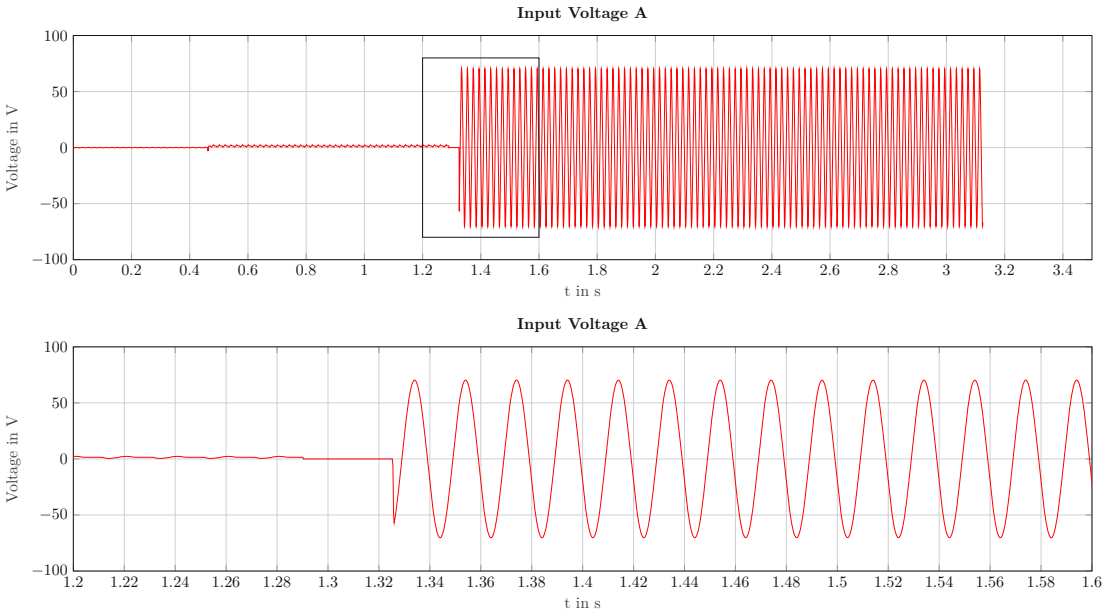
(b) Real-Time State Engine DSP Output Voltage B



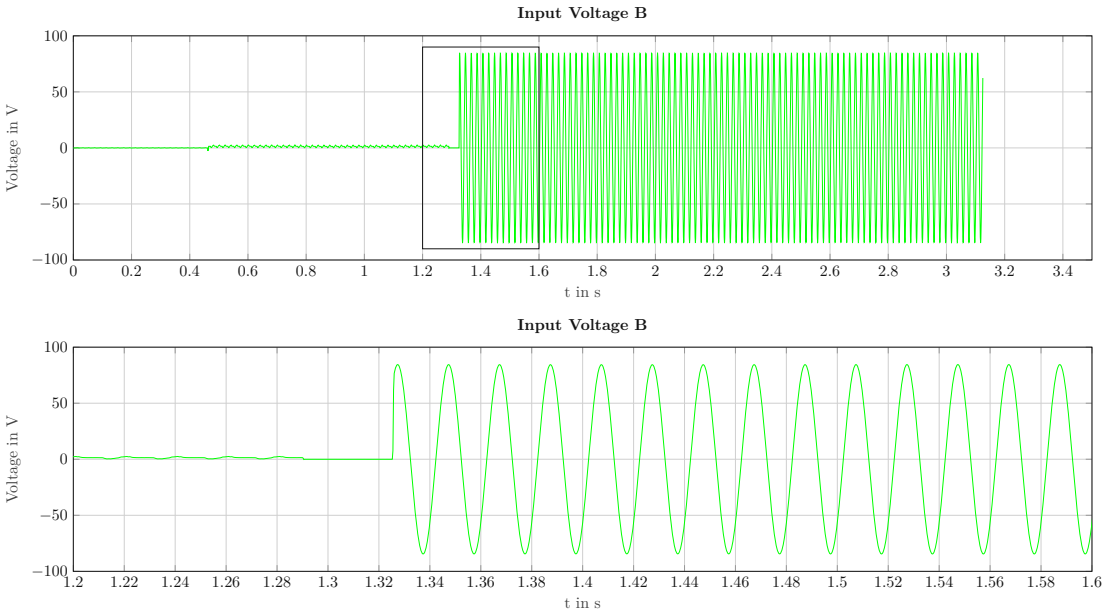
(c) Real-Time State Engine DSP Output Voltage C

Figure 5.3: Digital Substation Output Voltages

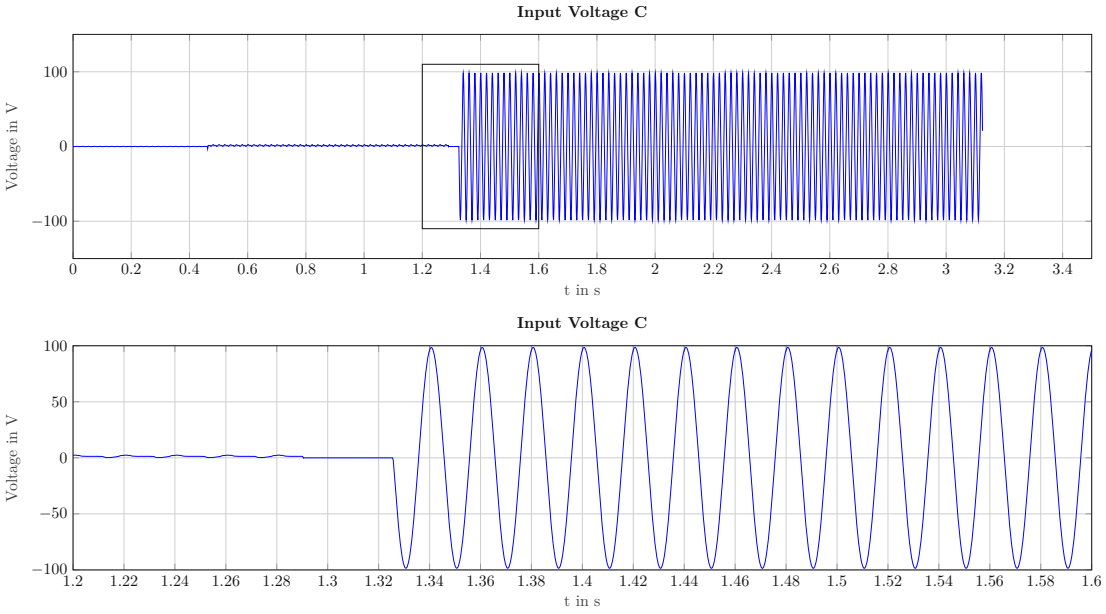
In parallel, we also recorded the data which were received over the SV stream on the DS Soft-DSP site. On this side we used the data recorder executable (7) from Figure 4.18 to record the data with 4000 Hz. The results are shown in Figure 5.4. Since the two Soft-DSPs run on two different threads, they also have a different point in time when they start to record the data. The DS Soft-DSP has less to prepare than the RSE Soft-DSP and can therefore start the logging earlier. That's why we have performed a time shift for the evaluation on the DS side, so that the results of the two Soft-DSPs can be better compared and thus the timing can be better understood. In our case the time shift was ≈ 1 s, but this is not a problem, because it has no influence on the evaluation. Furthermore, Figure 5.4 shows that the MU converted the three different voltages correctly and also the transport of the SV packets over the LAN worked as expected. To measure the quality of the received signal, we calculated the Mean Squared Error (MSE) of this signal with a perfect sine signal starting at 1.33 s where the real input signal was received. As a result we got an $MSE = 0.053612$ which is excellent.



(a) Digital Substation DSP Input Voltage A



(b) Digital Substation DSP Input Voltage B



(c) Digital Substation DSP Input Voltage C

Figure 5.4: Digital Substation Input Voltages

Finally, we looked again in the RSE Soft-DSP at the *Output Voltage A* after it has been output, converted by the MU, transmitted over the network and resampled by the *Bridge*. The result can be found in Figure 5.5. As we used here the same data recorder executable (5) as for the output signals the times of Figure 5.3 and Figure 5.5 are the same and therefore, the time how long the signal needed to be at the same point again where it was set, can be seen in Figure 5.5. After approximately 1.8 s the first sample point was received in the RSE Soft-DSP. This time is a sum of different factors. The RSE Soft-DSP needs about 450 ms until the first calculated output value can be transmitted to the AmpMan. This hardware needs then also about 800 ms until everything is stable and the outputs are outputting some voltages. Another 50 ms is used by the MU and the distribution via the LAN. The remaining 500 ms are added by the bridge at the *Frame-Buffer* from Figure 4.18. To compensate the jitter of the SV frames, we decided to buffer half a second before the bridge starts to work. Furthermore, the bottom diagram of Figure 5.5 illustrates that the resampling of the signal was correct because the received signal looks the same as the one which has been output. Also for this transformed signal we calculated the MSE and got a slightly worse value $MSE = 0.06785$ then before the bridge, but this was expected due to the up- and downsampling of the signal. Nevertheless, the value is still excellent after the resampling.

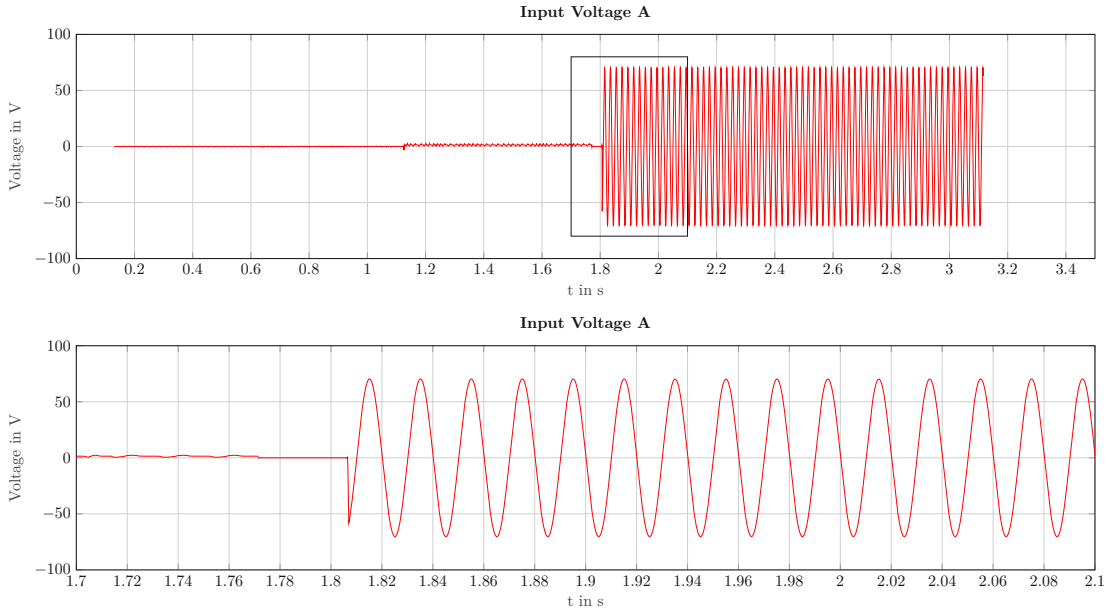
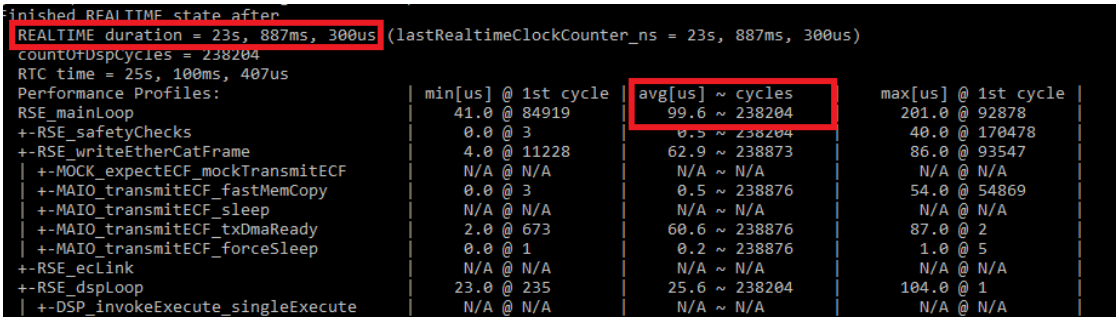


Figure 5.5: Real-Time State Engine DSP Input Voltage A

5.5 Analysis of real-time behaviour

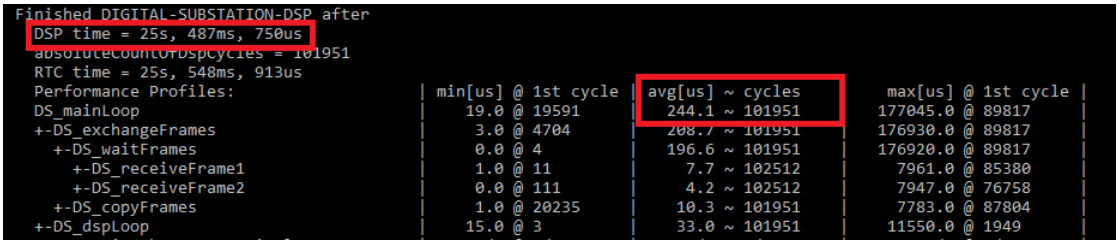
After checking the signals we also have to check our system if we do not disturb the 100 µs time cycle of our real-time system. Each of our Soft-DSPs have an infrastructure to provide some values about the timing. Parts of this timing report can be found in Figure 5.6a for the RSE Soft-DSP and in Figure 5.6b for the DS Soft-DSP. The important parts are highlighted with a red rectangle. If we compare the two times on top left of Figure 5.6a and Figure 5.6b we can again see that the *DS Soft-DSP* started a bit earlier than the *RSE Soft-DSP*. But the most important time is the other mark one in Figure 5.6a because this value shows the average cycle time of the RSE Soft-DSP. As shown the average cycle time is 99.6 µs which is in a normal range and indicates that the newly implemented feature does not disturb the already running system. A similar value can be found in Figure 5.6b for the DS Soft-DSP which is 244.1 µs and is also in a good range if we compare it to the one mentioned in Figure 4.18.

The result of the evaluation was very satisfying and showed that the idea can be implemented on the given measuring device without disturbing the hard real-time conditions. We also discovered some limits during evaluating, which we will discussed in more detail in the next Chapter.



```
Finished REALTIME state after
 REALTIME duration = 23s, 887ms, 300us (lastRealtimeClockCounter_ns = 23s, 887ms, 300us)
 countOfDspCycles = 238204
 RTC time = 25s, 100ms, 407us
 Performance Profiles:                  | min[us] @ 1st cycle | avg[us] ~ cycles | max[us] @ 1st cycle |
 RSE_mainLoop                           |    41.0 @ 84919     |   99.6 ~ 238204  |    201.0 @ 92878    |
 +-RSE_safetyChecks                     |     0.0 @ 3         |   0.5 ~ 238204   |     40.0 @ 170478   |
 +-RSE_writeEtherCatFrame               |     4.0 @ 11228     |   62.9 ~ 238873  |     86.0 @ 93547    |
 | +-MOCK_expectECF_mockTransmitECF     |     N/A @ N/A       |    N/A ~ N/A     |      N/A @ N/A      |
 | +-MAIO_transmitECF_fastMemCopy       |     0.0 @ 3         |    0.5 ~ 238876  |     54.0 @ 54869    |
 | +-MAIO_transmitECF_sleep             |     N/A @ N/A       |    N/A ~ N/A     |      N/A @ N/A      |
 | +-MAIO_transmitECF_txDmaReady        |     2.0 @ 673       |   60.6 ~ 238876  |     87.0 @ 2        |
 | +-MAIO_transmitECF_forceSleep        |     0.0 @ 1         |    0.2 ~ 238876  |     1.0 @ 5         |
 +-RSE_ecLink                           |     N/A @ N/A       |    N/A ~ N/A     |      N/A @ N/A      |
 +-RSE_dspLoop                          |    23.0 @ 235       |   25.6 ~ 238204  |    104.0 @ 1        |
 | +-DSP_invokeExecute_singleExecute    |     N/A @ N/A       |    N/A ~ N/A     |      N/A @ N/A      |
```

(a) Real-Time State Engine DSP timing analyse



```
Finished DIGITAL-SUBSTATION-DSP after
 DSP time = 25s, 487ms, 750us
 absoluteCountOfDspCycles = 101951
 RTC time = 25s, 548ms, 913us
 Performance Profiles:                  | min[us] @ 1st cycle | avg[us] ~ cycles | max[us] @ 1st cycle |
 DS_mainLoop                            |    19.0 @ 19591     |  244.1 ~ 101951  |  177045.0 @ 89817   |
 +-DS_exchangeFrames                    |     3.0 @ 4704      |  208.7 ~ 101951  |  176930.0 @ 89817   |
   +-DS_waitFrames                      |     0.0 @ 4         |  196.6 ~ 101951  |  176920.0 @ 89817   |
     +-DS_receiveFrame1                 |     1.0 @ 11        |    7.7 ~ 102512  |    7961.0 @ 85380   |
     +-DS_receiveFrame2                 |     0.0 @ 111       |    4.2 ~ 102512  |    7947.0 @ 76758   |
   +-DS_copyFrames                      |     1.0 @ 20235     |   10.3 ~ 101951  |    7783.0 @ 87804   |
 +-DS_dspLoop                           |    15.0 @ 3         |   33.0 ~ 101951  |   11550.0 @ 1949    |
```

(b) Digital Substation Soft-DSP timing analyse

Figure 5.6: Timing analyse of both Soft-DSPs

CHAPTER 6

Conclusion and Future Work

In the previous Chapter 5 we already saw that the idea of this thesis is working without disturbing the already running real-time system. Nevertheless, we faced some issues during the evaluation and found some limits for the current implementation which will be discussed in the first part of this chapter. After that the future work and how this implementation can be extended will be described.

During the first runs we noticed that the *Frame buffer* from Figure 4.18 was often empty and therefore the system stopped, which should not happen under any circumstances. As already described in Chapter 5 and shown in Figure 5.2 we found out, that approximately 20 % of all received packets were late. Furthermore, we could see in Wireshark[1] that the time difference of the time stamp added from the program is not constant to the one added from the ET 2000. After some investigation we found out, that depending on the Ethernet interface settings the operating system handles the interrupt behaviour of the received packets differently. It can happen that the operating system buffers some frames and just sends a single interrupt for multiple packets. With this knowledge we changed the settings that there is an interrupt for every received packet as soon as possible so that the DS Soft-DSP can process the packets faster and that there is no additional delay. The big disadvantage of one interrupt per packet is that the CPU workload gets increased significantly. This hardly has any effect on a powerful CPU but on our system there is a P2020[2] with only two cores with a clock frequency up to 1.2 GHz installed. Furthermore, one of these two cores is as good as possible isolated and is only used by the RSE Soft-DSP so that the hard real-time conditions can be achieved. The second core is shared for all other activities and is therefore very busy and a modification of the interrupt handling has a big influence on the performance of this second core. Nevertheless, we decided to go for this modification because the system could handle this overhead. To compensate the variety of the time differences between

[1] https://www.wireshark.org/

[2] https://www.nxp.com/docs/en/fact-sheet/QP20XXFS.pdf

the packets we decided to wait at the beginning of the execution until the *Frame-Buffer* of Figure 4.18 has a specific number stored. With this two changes we were able to get the results shown in Chapter 5.

Further we found out, that we can not resample all the signals which we received over the SV packets. As shown in Figure B.3 a single SV packet would include four voltages and four currents. One for each phase and one for the neutral. As already described in Chapter 5 we only resampled the *Output Voltage A* and forwarded this information to the RSE Soft-DSP. This was done because when we tried to do all eight signals the system was overwhelmed and the execution was stopped. In principle the implementation would work for all eight signals if there would be a more powerful CPU in place.

Overall the aim of the thesis could be met and all the problems described in Section 1.1 could be solved. We could show in this thesis how to exchange data between two independent threads which are running on different clock rates without disturbing an already running hard real-time system.

This brings us to a possible expansion in the future. The measuring device will be revised and will then get a more powerful CPU with more cores. Similar to the RSE Soft-DSP the DS Soft-DSP will then also get an isolated core and then the performance issues will be gone and all the eight channels can be resampled. Furthermore, we will get a cycle time closer to 250 µs for the DS Soft-DSP and therefore also a better result of the resampled signal.

Currently, we just provide the resampled signals to the RSE Soft-DSP but the data is not used yet. An extension to this work is to use these data and do the same calculations as already done with the conventional converters and analog signals. Another possible extension could be to evaluate the received data immediately and to change the outputs of the measuring device based on the received data. For this expansion we would also need the more powerful CPU and we would have to keep all possible delays in the chain as small as possible. This would, however, have been beyond the scope of the thesis.

APPENDIX A

IEC 61850 Parts

Part	Title	Ver.	Date
1 TR	Introduction and overview	2.0	03/2013
2 TS	Glossary	2.0	04/2019
3	General requirements	2.0	12/2013
4	System and project management	2.0	04/2011
5	Communication requirements for functions and device models	2.0	01/2013
6	Configuration description language for communication in power utility automation systems related to IEDs	2.1	06/2018
7-1	Basic communication structure – Principles and models	2.0	07/2011
7-2	Basic information and communication structure – Abstract communication service interface (ACSI)	2.0	08/2010
7-3	Basic communication structure – Common data classes	2.0	12/2010
7-4	Basic communication structure – Compatible logical node classes and data object classes	2.0	03/2010
7-6 TR	Guideline for definition of Basic Application Profiles (BAPs) using IEC 61850	1.0	01/2019
7-7 TS	Machine-processable format of IEC 61850-related data models for tools	1.0	03/2018
7-410	Basic communication structure – Hydroelectric power plants –Communication for monitoring and control	2.1	11/2015
7-420	Basic communication structure – Distributed energy resources logical nodes	1.0	03/2009
7-500 TR	Basic information and communication structure – Use of logical nodes for modeling application functions and related concepts and guidelines for substations	1.0	07/2017
7-510 TR	Basic communication structure – Hydroelectric power plants –Modelling concepts and guidelines	1.0	03/2012
8-1	Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1, ISO 9506-2) and to ISO/IEC 8802-3	2.0	06/2011

Part	Title	Ver.	Date
8-2	Specific communication service mapping (SCSM) – Mapping to Extensible Messaging Presence Protocol (XMPP)	1.0	12/2018
9-2	Specific communication service mapping (SCSM) – Sampled values over ISO/IEC 8802-3	2.0	09/2011
9-3	Precision time protocol profile for power utility automation	1.0	05/2016
10	Conformance testing	2.0	12/2012
80-1 TS	Guideline to exchanging information from a CDC-based data model using IEC 60870-5-101 or IEC 60870-5-104	2.0	07/2016
80-3 TR	Mapping to web protocols – Requirements and technical choices	1.0	11/2015
80-4 TS	Translation from the COSEM object model (IEC 62056) to the IEC 61850 data model	1.0	03/2016
90-1 TR	Use of IEC 61850 for the communication between substations	1.0	03/2010
90-2 TR	Using IEC 61850 for communication between substations and control centers	1.0	02/2016
90-3 TR	Using IEC 61850 for condition monitoring diagnosis and analysis	1.0	05/2016
90-4 TR	Network engineering guidelines	1.0	08/2013
90-5 TR	Use of IEC 61850 to transmit synchrophasor information according to IEEE C37.118	1.0	05/2012
90-6 TR	Use of IEC 61850 for Distribution Automation Systems	1.0	09/2018
90-7 TR	Object models for power converters in distributed energy resources (DER) systems	1.0	02/2013
90-8 TR	Object model for E-mobility	1.0	04/2016
90-10 TR	Models for scheduling	1.0	10/2017
90-12 TR	Wide area network engineering guidelines	1.0	07/2015
90-17 TR	Using IEC 61850 to transmit power quality data	1.0	05/2017

Table A.1: Overview of all parts of the IEC 61850 standard [TC 19]

APPENDIX B

Content of an IEC 61850-9-2-LE Ethernet frame

Figure B.1 shows the content of a IEC 61850-9-2-LE Ethernet frame. The *APDU* field mentioned in this figure is then shown in more detail in Figure B.2. The *Sequence of Data* field of Figure B.2 is then described in Figure B.3.

Figure B.1 remarks [UCA11]

- Data fields with a length of 1 byte have the Most Significant Bit (MSB) on the left side and the Least Significant Bit (LSB) on the right side.
- Data fields longer than 1 byte have the MSB at the top left and the LSB at the bottom right, as shown in the “Preamble” data field.
- Sending bytes, bit 1 is the first bit.

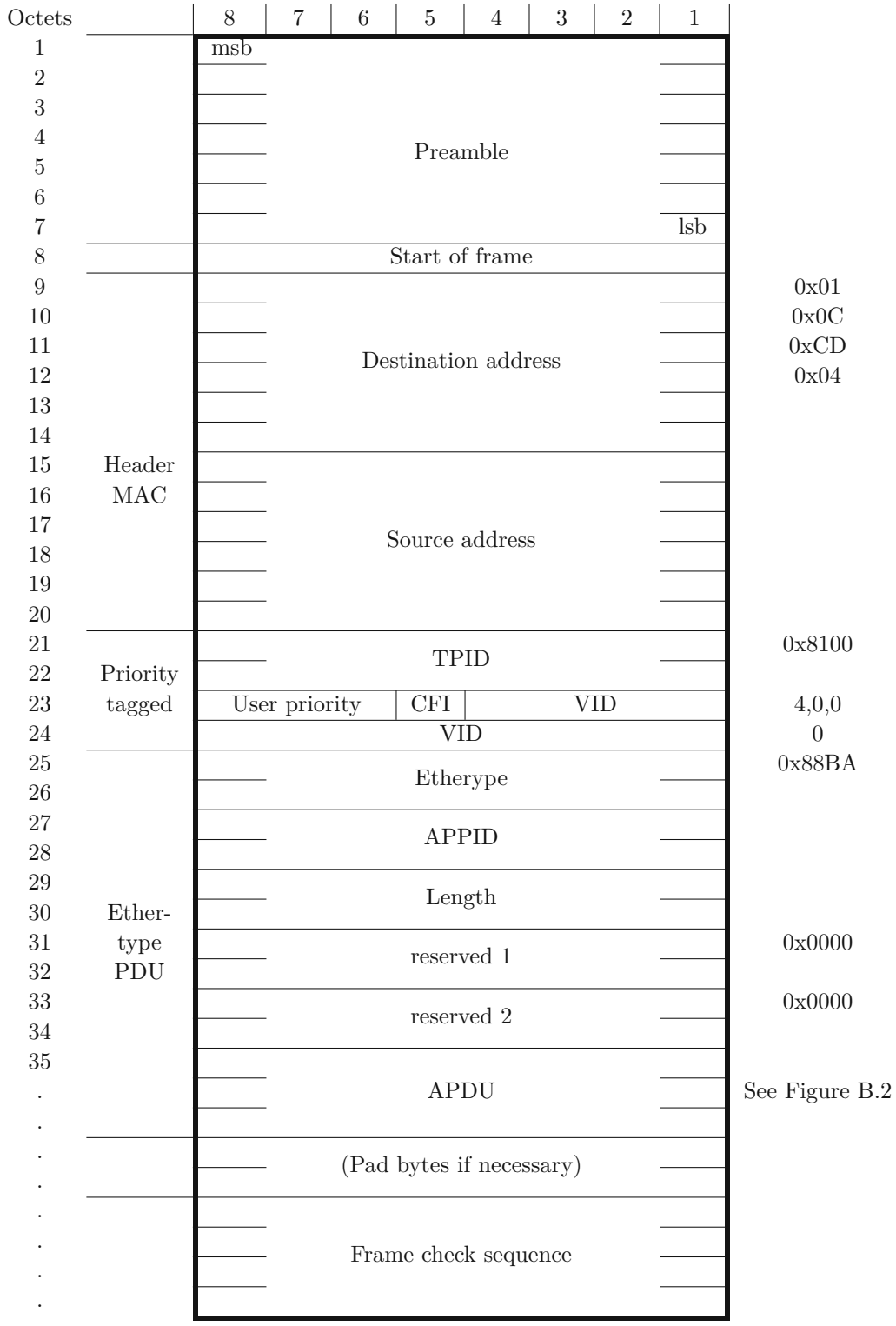


Figure B.1: Content of an IEC 61850-9-2-LE Ethernet frame [UCA11]

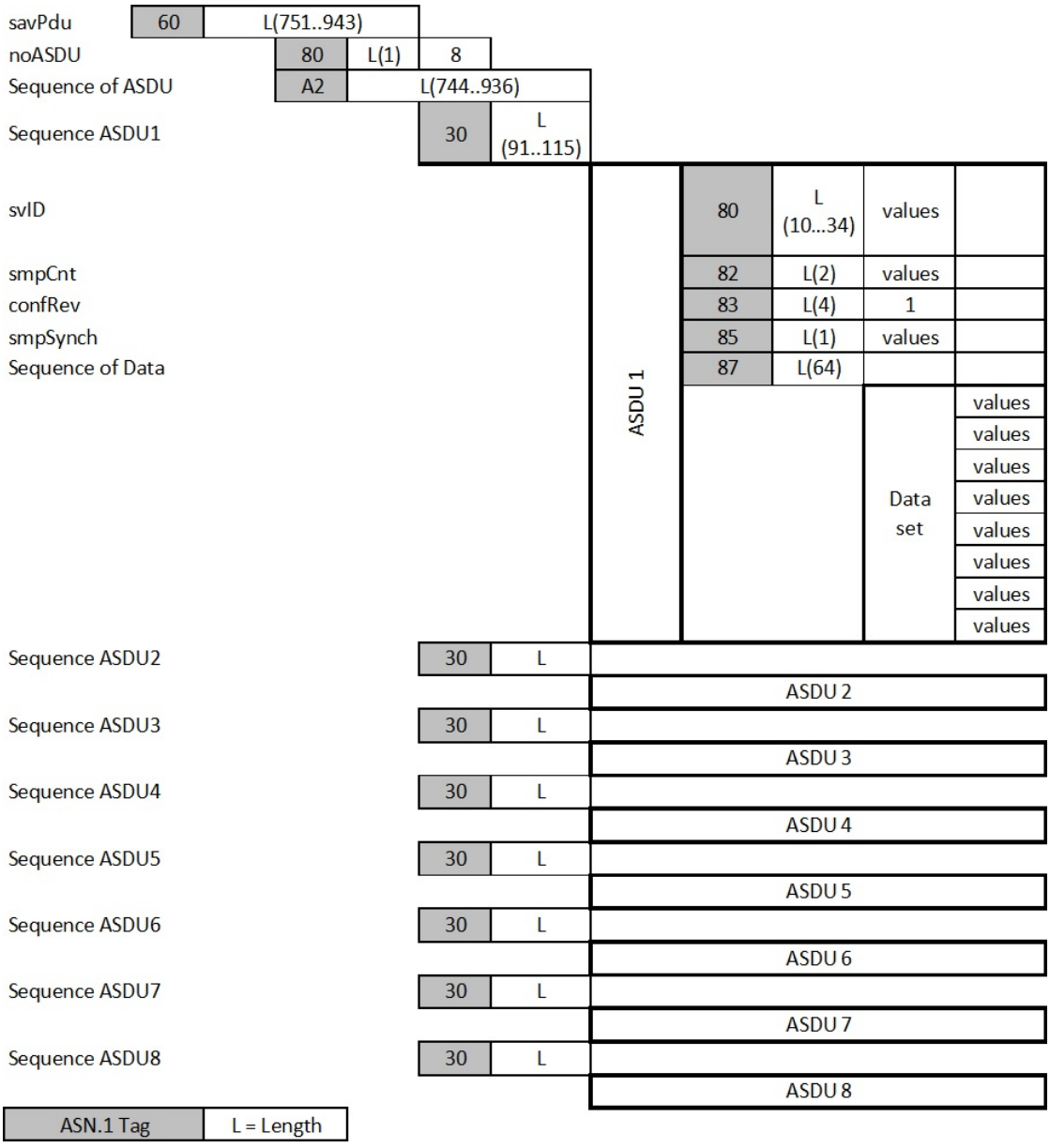


Figure B.2: Content of an APDU data field [UCA11]

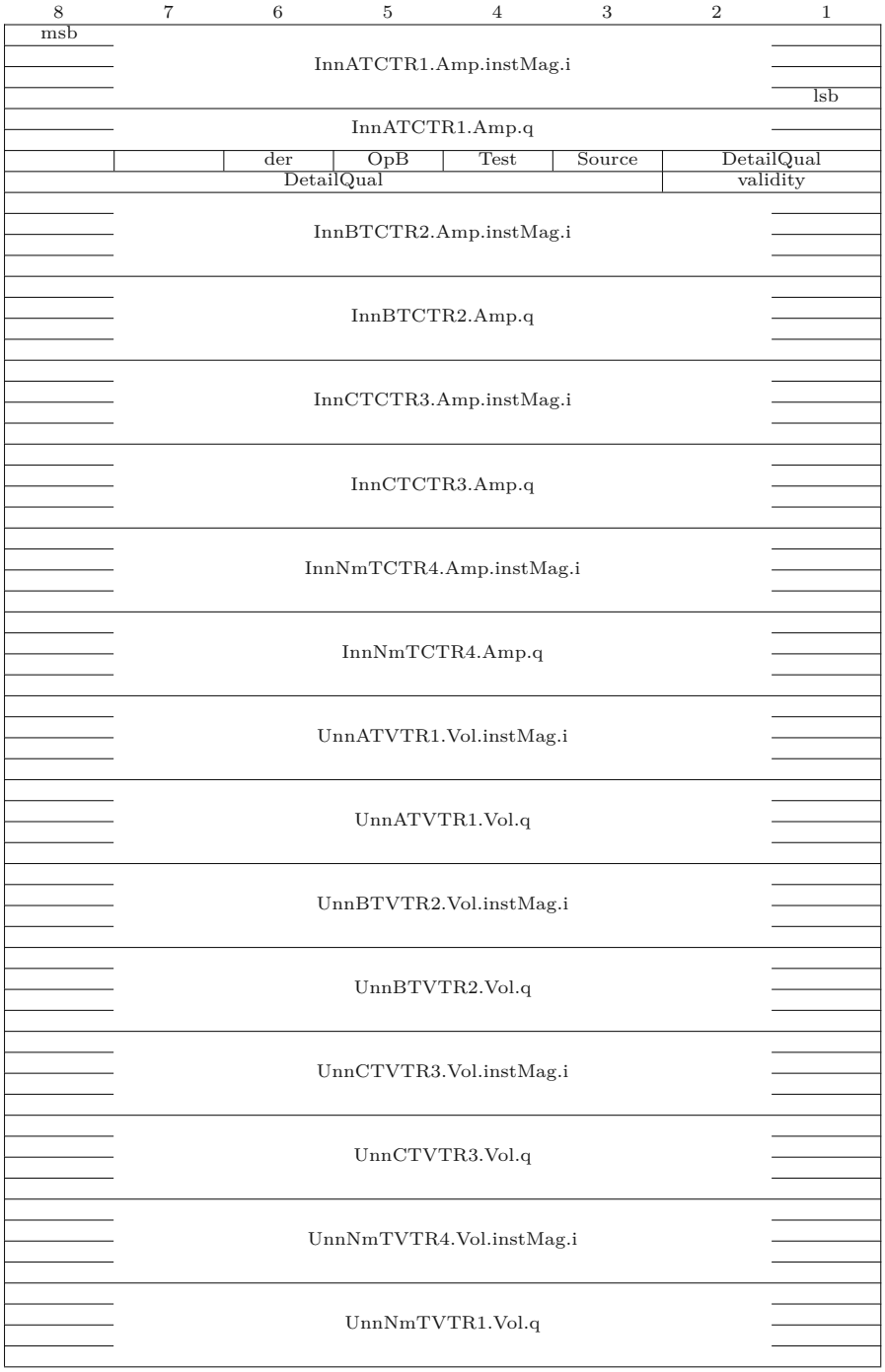


Figure B.3: Encoding of dataset [UCA11]

APPENDIX C

Content of the additional ET 2000 packet data

<table>
<tr><td>6 Bytes</td><td colspan="4">Identifier 01 01 05 10 00 00
(symbolic MAC address)</td></tr>
<tr><td rowspan="10">1 Byte</td><td colspan="4">Port designation 0...7 of arriving frame</td></tr>
<tr><td></td><td>Port</td><td>Hex</td><td>Bin</td></tr>
<tr><td></td><td>X1.0</td><td>0x80</td><td>10000000</td></tr>
<tr><td></td><td>X1.1</td><td>0x40</td><td>01000000</td></tr>
<tr><td></td><td>X2.0</td><td>0x20</td><td>00100000</td></tr>
<tr><td></td><td>X2.1</td><td>0x10</td><td>00010000</td></tr>
<tr><td></td><td>X3.0</td><td>0x08</td><td>00001000</td></tr>
<tr><td></td><td>X3.1</td><td>0x04</td><td>00000100</td></tr>
<tr><td></td><td>X4.0</td><td>0x02</td><td>00000010</td></tr>
<tr><td></td><td>X4.1</td><td>0x01</td><td>00000001</td></tr>
<tr><td rowspan="4">1 Byte</td><td colspan="4">Bits 0...2 reserved</td></tr>
<tr><td colspan="4">Bit 3 alignment error</td></tr>
<tr><td colspan="4">Bit 4 CRC error in recorded frame</td></tr>
<tr><td colspan="4">Bits 5...7 unused</td></tr>
<tr><td>8 Bytes</td><td colspan="4">Timestamp in ns
(reading direction backwards)</td></tr>
</table>

Table C.1: Constitution of the 16 additional bytes of the ET 2000

List of Figures

List of Tables

List of Algorithms

Acronyms

ADC Analog-to-Digital Converter. 30

ALSS Augmented Least-Squares Solver. 26, 27, 71

ALU Arithmetic Logic Unit. 19

AmpMan Amplifier Manager. 20, 21, 50, 53, 57

APDU Application Protocol Data Unit. 13

ASIC Application-Specific Integrated Circuit. 17

CPU Central Processing Unit. 3, 21, 45, 61, 62

CT Current Transformer. vii, ix, 1, 2, 19, 50

DAC Distribution Automation and Control. 8

DAL Device Abstraction Layer. 18, 21

DF Digital Filter. 24

DMA Direct Memory Access. 21

DS Digital Substation. 45, 49, 55, 58, 61, 62

DSP Digital Signal Processor. 18, 19, 21, 24, 45–47, 49, 53, 55, 57–59, 61, 62, 72

EIT Electronic Instrument Transformers. 24, 25, 71

EMS Energy Management System. 8

ETG EtherCAT Technology Group. 15

EtherCAT Ethernet for Control Automation Technology. ix, 5, 15–21, 45, 46, 71

FAT Factory Acceptance Tests. 18

FFT Fast Fourier Transform. 28

FIR Finite Impulse Response. 23, 24, 39, 41–43, 72

FMMU Fieldbus Memory Management Unit. 16, 17

FPGA Field Programmable Gate Array. 21

FPU Floating-Point Unit. 19

GOOSE Generic Object Oriented Substation Event. 11, 13

GPS Global Positioning System. 4

IEC International Electrotechnical Commission. vii, ix, 1–3, 9–13, 15, 51, 65, 71

IED Intelligent Electronic Device. 1, 9–13, 23–25, 71

IEEE Institute of Electrical and Electronics Engineers. 4, 12, 14, 15, 28, 71

IIR Infinite Impulse Response. 39, 41–43, 72

IP Internet Protocol. 17

IPC Inter-Process Communication. 3, 5, 21

IRIG-B Inter Range Instrumentation Group-B. 14

LAN Local Area Network. 3, 49, 50, 55, 57

LE Light Edition. 13

LN Logical Node. 10, 12

LSB Least Significant Bit. 65

LSI Linear Shift Invariant. 39

LSS Least-Squares Solver. 26

MAC Media Access Control. 51

MAIO Master Asynchronous Input Output. 21

MMS Manufacturing Message Specification. 11

MMXN non-phase related measurement. 10

MSB Most Significant Bit. 65

MSE Mean Squared Error. 55, 57

MU Merging Unit. 2, 4, 12, 13, 19, 23–25, 49–51, 55, 57, 71

NCIT Non-Conventional Instrument Transformers. ix, 1–4, 19, 71

PCHIP Piecewise Cubic Hermite Interpolating Polynomial. 32, 33

PLC Programmable Logic Controller. 10

PPS Pulse Per Second. 4, 27

PTM Primary Test Manager. 19

PTP Precision Time Protocol. 4, 14, 15, 19, 21

RMS Root Mean Square. 27

RSE Real-Time State Engine. 21, 45, 46, 49, 52, 53, 55, 57, 58, 61, 62

SAS Substation Automation Systems. 10, 13, 14, 24

SCADA Supervisory Control and Data Acquisition. 8, 9

SCL System Configuration description Language. 10, 11

SNTP Simple Network Time Protocol. 14

SPC Samples Per Cycle. 13

SRS Sampling Rate Synchronization. 23, 24, 71

SV Sampled Value. vii–ix, 1–5, 12, 13, 19, 27, 29–31, 33, 38, 45, 49–53, 55, 57, 62, 72

TC Technical Committee. 12

TCP Transmission Control Protocol. 17

TR Technical Report. 63, 64

TS Technical Specification. 63, 64

UDP User Datagram Protocol. 17

VT Voltage Transformer. vii, ix, 1, 2, 19, 50

WC Working Counter. 17

XML Extensible Markup Language. 10

Bibliography

[Blu07] S. W. Blume. *System Overview, Terminology, and Basic Concepts*, chapter 1, pages 1–18. John Wiley & Sons, Ltd, 2007.

[BLW03] K.P. Brand, V. Lohmann, and W. Wimmer. *Substation Automation Handbook*. Utility Automation Consulting Lohmann, 2003.

[CK12] E. Ward Cheney and David R. Kincaid. *Numerical Mathematics and Computing*. Brooks Cole, 2012.

[CMSdR20] Yeying Chen, Enrico Mohns, Michael Seckelmann, and Soeren de Rose. Precise Amplitude and Phase Determination Using Resampling Algorithms for Calibrating Sampled Value Instruments. *Sensors*, 20(24), 2020.

[CXW20] Jun Chen, Qingshan Xu, and Kai Wang. Research and Application of Generator Protection Based on Fiber Optical Current Transformer. *IEEE Access*, 8:172405–172411, 2020.

[CY10] Chao Cai and Yuping Lu. A digital sampling rate synchronization scheme for fully digital relay protection. In *IEEE PES T D 2010*, pages 1–6, 2010.

[DFF+11] C. M. De Dominicis, P. Ferrari, A. Flammini, S. Rinaldi, and M. Quarantelli. On the Use of IEEE 1588 in Existing IEC 61850-Based SASs: Current Behavior and Future Challenges. *IEEE Transactions on Instrumentation and Measurement*, 60(9):3070–3081, Sep. 2011.

[Dü17] Lars Dürkop. *Automatische Konfiguration von Echtzeit-Ethernet*. Technologien für die intelligente Automation, Technologies for Intelligent Automation. Springer Berlin Heidelberg Imprint: Springer Vieweg, Berlin, Heidelberg, 2017.

[Exe07] Reinhard Exel. Systemdesign und Entwicklung einer Netzwerkkarte zur Uhrensynchronisation. Master's thesis, Vienna University of Technology, 2007.

[GH11] J.M. Gers and E.J. Holmes. *Protection of electricity distribution networks, 3rd edition*. Institution of Engineering and Technology, 2011.

[Gö14] Turan Gönen. *Electric power distribution engineering*. CRC Press, Boca Raton, Florida, 3rd ed. edition, 2014.

[HHFK17] O. Hegazi, E. Hammad, A. Farraj, and D. Kundur. IEC-61850 GOOSE traffic modeling and generation. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1100–1104, Nov 2017.

[HS07] Jochen Haude and Ulrich Sundermann. Digitale Anbindung von Hochspannungs–Betriebsmitteln nach IEC 61850. In *Webbasierte Automatisierung in der elektrischen Energietechnik*, pages 1–10, 2007.

[KRP18] S. Kariyawasam, A. D. Rajapakse, and N. Perera. Investigation of Using IEC 61850-Sampled Values for Implementing a Transient-Based Protection Scheme for Series-Compensated Transmission Lines. *IEEE Transactions on Power Delivery*, 33(1):93–101, Feb 2018.

[KSZ11] Mitalkumar G. Kanabar, Tarlochan Singh Sidhu, and Mohammad R. Dadash Zadeh. Laboratory Investigation of IEC 61850-9-2-Based Busbar and Distance Relaying With Corrective Measure for Sampled Value Loss/Delay. *IEEE Transactions on Power Delivery*, 26:2587–2595, 2011.

[MES14] Aneeq Mahmood, Reinhard Exel, and Thilo Sauter. Impact of hard-and software timestamping on clock synchronization performance over IEEE 802.11. In *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, pages 1–8, 2014.

[MML+16] Naiara Moreira, Elías Molina, Jesús Lázaro, Eduardo Jacob, and Armando Astarloa. Cyber-security in substation automation systems. *Renewable and Sustainable Energy Reviews*, 54:1552 – 1562, 2016.

[Mol04] Cleve B. Moler. *Numerical computing with MATLAB*. SIAM, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2004.

[OMI19] OMICRON electronics GmbH. TESTRANO 600 – Three-phase power transformer test system. `https://www.omicronenergy.com/en/products/testrano-600/`, 2019. Accessed: 14.06.2019.

[PA18] Filip Pröstl Andrén. *Model-driven engineering for smart Grid automation*. PhD thesis, Vienna University of Technology, 2018.

[PFA17] Florian Predl, Michael Freiburg, and Martin Anglhuber. Diagnosemessungen an Messwandlern. Technical report, OMICRON electronics GmbH, December 2017.

[Pla06] Robert Plato. *Numerische Mathematik kompakt*. Friedr. Vieweg & Sohn Verlag - GWV Fachverlage GmbH, Wiesbaden, Wiesbaden, 3. aktualisierte und verbesserte auflage edition, 2006.

[PT16] A. Patki and G. Thiagarajan. Low complexity, low latency resampling of asynchronously sampled signals. In *2016 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5, 2016.

[Sam17] Carlos Samitier. *Utility Communication Networks and Services*. CIGRE Green Books. Springer International Publishing Imprint: Springer, Cham, 2017.

[SNKB19] Mohit Sharma, Lam Nguyen, Sughosh Kuber, and Dinesh Baradi. Testing IEC-61850 Sampled Values-Based Transformer Differential Protection Scheme. In *PAC World Americas Conference 2019*, 10 2019.

[SSA+11] T. Strasser, M. Stifter, F. Andrén, D. B. de Castro, and W. Hribernik. Applying open standards and open source software for smart grid applications: Simulation of distributed intelligent control of power systems. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–8, July 2011.

[Sto12] J. Stoer. *Introduction to numerical analysis: Texts in applied mathematics*. Springer Verlag, 2012.

[SW05] M. Shahidehpour and Yaoyu. Wang. *Communication and control in electric power systems*. IEEE Press series on power engineering. IEEE,; IEEE Xplore,, Piscataway, New Jersey; [Piscataqay, New Jersey], 2005.

[TC 19] TC 57 - Power systems management and associated information exchange. Communication networks and systems for power utility automation – ALL PARTS. International Standard IEC 61850:2019, International Electrotechnical Commission, 2019.

[TJW11] R. Tao, B. Jiang, and C. Wang. Sampling rate conversion and data synchronization in big merging unit. In *2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT)*, pages 531–534, 2011.

[TNT+97] M. Takahashi, H. Noda, K. Terai, S. Ikuta, Y. Mizutani, T. Yokota, T. Kaminishi, and T. Tamagawa. Optical current transformer for gas insulated switchgear using silica optical fiber. *IEEE Transactions on Power Delivery*, 12(4):1422–1427, Oct 1997.

[UCA11] UCA International Users Group. *Implementation Guideline For Digital Interface To Instrument Transformers Using IEC 61850-9-2*, 2011.

[WB05] Heinz Wörn and Uwe Brinkschulte. *Echtzeitsysteme*. eXamen.press. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2005.

Erklärung zur Verfassung der Arbeit

Hiermit erkläre ich, dass ich die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis, insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, 23. September 2021

Mario PETER