Developing a decentralized peripheral PROFIBUS core for a Xilinx FPGA

A dissertation presented to

The School of Electrical, Electronic and Computer Engineering

North-West University

In partial fulfilment of the requirements for the degree

Magister Ingeneriae

In Computer and Electronic Engineering

by

Roelof Jacobus Burger

Supervisors:

Prof G. van Schoor

R.R. le Roux

November 2010

Potchefstroom Campus

DECLARATION

I hereby declare that all the material incorporated in this dissertation is my own original unaided work, except where specific reference is made by name or in the form of a numbered reference. The work herein has not been submitted for a degree at another university.

Signed:

_____

Roelof J. Burger

SUMMARY

The McTronX research group of the North-West University has over some years established a knowledge base in active magnetic bearing (AMB) systems. In 2009, an AMB system that met industrial standards in being robust, reliable and economical was developed by the research group. The digital control of the AMB system was implemented with the use of a dedicated single-board computer and communication hardware that interface with the motor drive electronics, power amplifiers and sensor drive units of the AMB system. A Xilinx® field programmable gate array (FPGA), connected to the single-board computer, was used to control the AMB system. The AMB system was designed to be used in a helium blower application and to form a basis for AMB and digital control research.

A programmable logic controller (PLC) is connected to the controller to operate the AMB system. To establish communication between the PLC and the FPGA, the Fieldbus standard PROFIBUS DP was chosen as being a robust industrial standard communication protocol. To reduce the cost of the entire system, the need arose to implement the PROFIBUS DP protocol on the current FPGA of the system.

This project involves the research, design, implementation, verification and validation of the PROFIBUS DP protocol on a Xilinx® Virtex©-5 FPGA. The PROFIBUS DP standard was researched, analyzed and developed in VHDL for the specific Xilinx® Virtex©-5 FPGA. The implemented protocol is used to establish a standardized PROFIBUS DP network between the PLC and the FPGA controller.

Through simulation the basic protocol was tested and later implemented in the real-time environment. Intensive verification and validation was done to ensure that the developed protocol conforms to the robust PROFIBUS DP standard and simultaneously meet the requirements and specifications of the AMB control system.

This dissertation documents the entire PROFIBUS implementation process, from standard analysis through to verification and validation of the developed protocol. In conclusion, the developed protocol is compared against a commercial off-the-shelf PROFIBUS PMC module. It was found that the VHDL-based PROFIBUS DP protocol not only competes well with the commercial PROFIBUS device, but also outperforms the device in various aspects.

Keywords: *PROFIBUS, FPGA, Fieldbus, Xilinx, OSI reference model.*

ACKNOWLEDGEMENTS

I would like to acknowledge the following people, in no particular order, for their contributions and support during the duration of this project:

- My Lord and God for giving me the strength and intellect needed for this study
- Professor George van Schoor, my study leader, for his guidance, advice and support
- Rikus le Roux, my colleague and assistant study leader, for his support and determination
- The entire McTronX research group for the wonderful working environment they established and the support that each member contributed
- My girlfriend, Rachel Venter, for her encouragement and understanding
- My family for their continuous support and contributions

“Blessed is the man who remains steadfast under trial, for when he has stood the test he will receive the crown of life, which God has promised to those who love him. “ James 1: 12

Table of contents

List of figures

List of tables

List of abbreviations

ADES	Active magnetic bearing drive electronic system
AMB	Active magnetic bearing
ARQ	Automatic repeat requests
ACII	American standard code for information interchange
ASIC	Application specific integrated circuit
CAN	Controller area network
COTS	Commercial off-the-shelf
CRC	Cyclic redundancy check
Compact PCI	Compact peripheral component interconnect
CSMA/CD	Carrier sense multiple access with collision detection
DA	Destination address
DDR	Double data rate
DP	Distributed peripherals
DSAP	Destination service access point
DSP	Digital signal processor
DU	Data unit
ED	End delimiter
FAL	Fieldbus application layer
FC	Function code
FCS	Frame check sequence
FDL	Fieldbus data link layer
FEC	Forward error correction
FF	Foundation fieldbus
FMS	Fieldbus message specification
FPGA	Field programmable gate array
GUI	Graphical user interface
HART	Highway addressable remote transducer
HMI	Human machine interface
IEEE	Institute of electrical and electronics engineers
IEC	International electrotechnical commission

IP	Intellectual property
IP Core	Intellectual property core (reusable block of logic)
ISI	Intersymbol interference
I/O	Input / Output
LAN	Local area network
LE	Length
MAC	Medium access control
MIPS	Million instructions per second
PA	Process automation
PBMR	Pebble bed modulator reactor
PC	Personal computer
PCI	Peripheral component interconnect
PMC	PCI mezzanine card
PLC	Programmable logic controller
PROFIBUS	Process Fieldbus
RDS	Rotor delevitation system
r/min	Revolutions per minute
RTU	Remote terminal unit
SA	Source address
SAP	Service access point
SBC	Single-board computer
SCADA	Supervisory control and data acquisition
SD	Start delimiter
SDRAM	Synchronous dynamic random access memory
SND	Send and request data without acknowledgement
SRAM	Static random access memory
SRD	Send and request data with acknowledgement
SSAP	Source service access point
TDMA	Time division multiple access
VHDL	Very high-speed integrated circuits hardware description language
VME	Versamobile-Eurocard

List of symbols

V	Voltage
mA	Milliampere
kbps	Kilobits per second
Mbps	Megabits per second
m	Meter
Ω	Ohm
pF	Picofarad
dB	Decibel
MHz	Megahertz
a	Transition capability
µs	Micro seconds
n	Number of bits
e	Error bits
k	Information bits
R	Nominal bit rate
η_{dc}	Efficiency of the data coding scheme
η_{ma}	Medium access control efficiency
T_{ms}	Message service time
d	Number of user data octets
N_m	Number of masters
T_C	Cycle time
T_{TC}	Token time
T_R	Response time
P	Bit error rate
R_e	Residual error rate

List of standards

IEC 61158	Industrial communication networks - Fieldbus specifications
IEC 61784	Industrial communication networks - Profiles
ISO/IEC 7498	Open Systems Interconnection reference model
IEEE 802.5	IEEE token ring specification

Chapter 1

Introduction

Chapter 1 presents an introduction and overview of active magnetic bearings and a drive electronic system that forms the platform for this study. Background regarding the hardware platform helps to illustrate the problem statement of this study which is also presented in this chapter together with the research methodology and possible solutions. The basic principles in this chapter will be discussed in more detail in the following chapter where a comprehensive literature study will be done. An overview of the dissertation is also presented at the end of this chapter.

1.1 Background

In the past years the McTronX research group of the North-West University has done intensive research on active magnetic bearing (AMB) systems. In 2009 McTronX developed an AMB system that complies with industrial standards in being robust, reliable and economical. The first AMB system developed at the North-West University was controlled with an expensive dSPACE® system that was incorporated with the use of MATLAB® and Simulink®. The most recent control system that was developed is implemented with the use of a dedicated single-board computer and communication hardware sub-system that interfaces with the motor drive electronics, power amplifiers and sensor drive units of the AMB system. This AMB system was developed for high-speed turbo-machinery for next-generation nuclear reactors and to form an AMB research platform.

In order to develop a reliable and robust AMB system, all applicable industrial standards are implemented on the system. The Fieldbus standard – PROFIBUS DP – was chosen as the preferred industrial communication standard for the high-speed data exchange that was needed in the system. PROFIBUS DP (distributed peripherals) is a popular open rugged bus system that meets industrial standards. This standard is currently implemented in the system with an expensive PROFIBUS peripheral component interconnect mezzanine card (PCI mezzanine card, or PMC). In order to reduce the cost of the system, methods are investigated and developed to implement the Fieldbus standard in a more cost effective manner that still conforms to the robust communication standard needed.

This chapter provides an introduction to the study by briefly explaining the AMB environment with special focus on the data communication of the system. A method is discussed for reducing the cost of the system, which implements the PROFIBUS standard on a field programmable gate array (FPGA) that is used in the system . This will reduce the cost of the system while preserving the industrial communication standards. The chapter continues by explaining how every objective will be researched and addressed. An overview of the dissertation is presented at the end of this chapter.

1.1.1 Active magnetic bearings

AMBs are non-contacting bearings used for high-speed turbo-machine applications. Magnetic forces suspend the rotor and provide a frictionless bearing system. Electromagnetic actuators, position sensors, power amplifiers and feedback controllers form the basic elements of any AMB system. Typical AMB systems restrict the rotor in 5 degrees of freedom by using two radial AMBs and one axial AMB [1].

AMBs allow for high-speed applications where active vibration control can be achieved. Mechanical wear will no longer be applicable, which means less maintenance and lower cost [2]. The benefits of using AMBs can be utilized in various applications. The developers of next generation nuclear reactors use magnetic radial bearings in the power turbines of their high-speed turbo-machinery [3]. The special advantages of AMBs allow for application in a number of areas which include [2]:

- Vacuum techniques
- Turbo machinery
- Machine tools
- Electric drives
- Textile machinery
- Energy storage
- Space applications
- Vibration isolation

Figure 1.1 explains the principle of operation of an AMB. The sensors detect and measure any displacement of the rotor, while a controller derives a control signal from the displacement measurement and controls the signal that is being sent to the power amplifiers. After the control signal has been transformed into a control current, a magnetic force is generated within the actuating magnet to keep the rotor in a centralized hovering position [2].

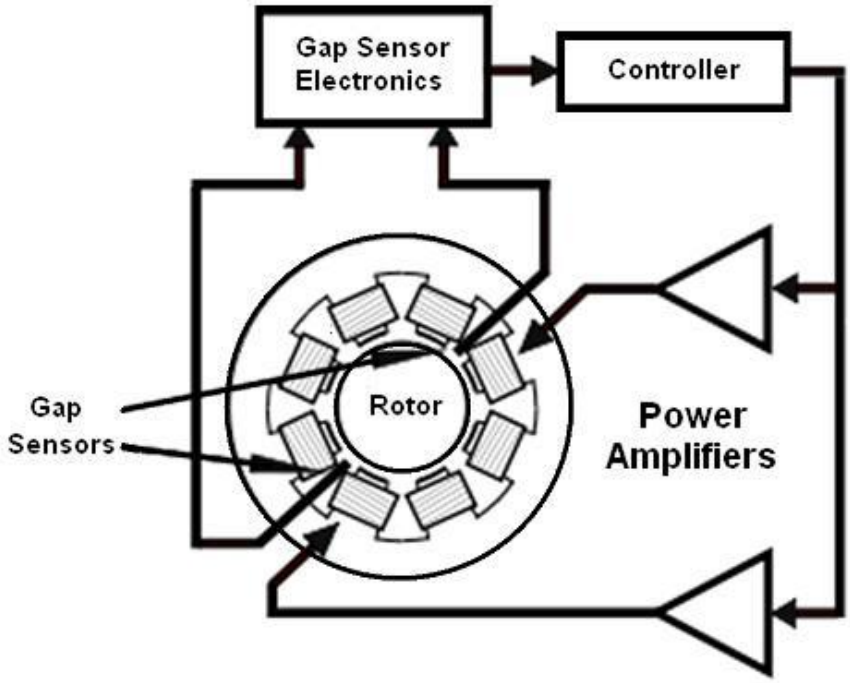


Figure 1.1 – Principle of operation of an active magnetic bearing [1]

An AMB system is a typical mechatronic system that uses electronic signals to produce, for example, forces or motion. Mechatronics is an interdisciplinary area of engineering sciences based on the traditional fields of mechanical and electrical engineering and on computer science. Sensors and microcomputers are interconnected to sense changes in the AMB environment and to react to the change with the correct method of information processing [2]. The controller forms a significant part of the AMB system in view of the fact that the entire system relies on the signal processing and control of the controller.

1.1.2 Digital AMB control

Although analogue control was considered to control AMB systems only a few years ago, it is now mainly used for very small bearings or low-cost systems. The benefits of digital controllers have added flexibility to the development of AMB controllers in the following ways [2]:

- Different control strategies can be tested at development phase through simulation
- Digital controllers can handle more complex control functions
- A digital controller can perform a large number of procedures, functions and additional tasks besides controlling and stabilizing the plant

- Digital control simplifies the calibration of sensors and other parameters
- These systems allow for conditioning of loads, displacements, vibrations, power amplifier currents and overall system performance
- Safety can be implemented through programming techniques. The system will deal with conditions, based on the safety procedures implemented through programming techniques
- Maintenance can be done by only changing the software

Digital controllers can be implemented on a development system with a host computer and peripherals. Processors, memory devices, filters, interfacing components, analogue-to-digital converters and digital-to-analogue converters form the basic hardware needed for a digital AMB controller [2].

The McTronX research group of the North-West University has designed and developed a digitally controlled AMB system that meets the standards and requirements of the industry. The AMB drive electronic system (ADES) utilizes digital control and is discussed in the following section.

1.1.3 AMB drive electronic system (ADES)

The control flow of the entire AMB system and ADES is shown in Figure 1.2. The compact PCI single-board computer (SBC) controls the main inputs and outputs through the system. “Compact peripheral component interconnect (cPCI) is an adaptation of the peripheral component interconnect (PCI) specification for industrial computer applications requiring a smaller, more robust mechanical form factor than the one defined for the desktop” [4]. A Virtex©-5 PMC module and a PROFIBUS DP PMC module are attached to the SBC. The FPGA (U6) controls the signals to the power amplifiers and receives signals from the sensor units. The controller (U7) processes the information and adjusts the reference values, sent to the power amplifiers, to keep the rotor levitated. This information is communicated via the single-board computer to the graphical user interface (GUI) (U1) and the PLC (U4) attached to the PROFIBUS DP module.

The GUI and programmable logic controller (PLC) act as the two user interfaces for the control of the AMBs. The GUI is used for maintenance by viewing sent and received data, changing the operational states of the system or to monitor the AMBs. The PLC is used as an industrial type user interface and operator. Buttons and lights are attached to the PLC that indicate the current state of the system and allow the user to control the AMB system.

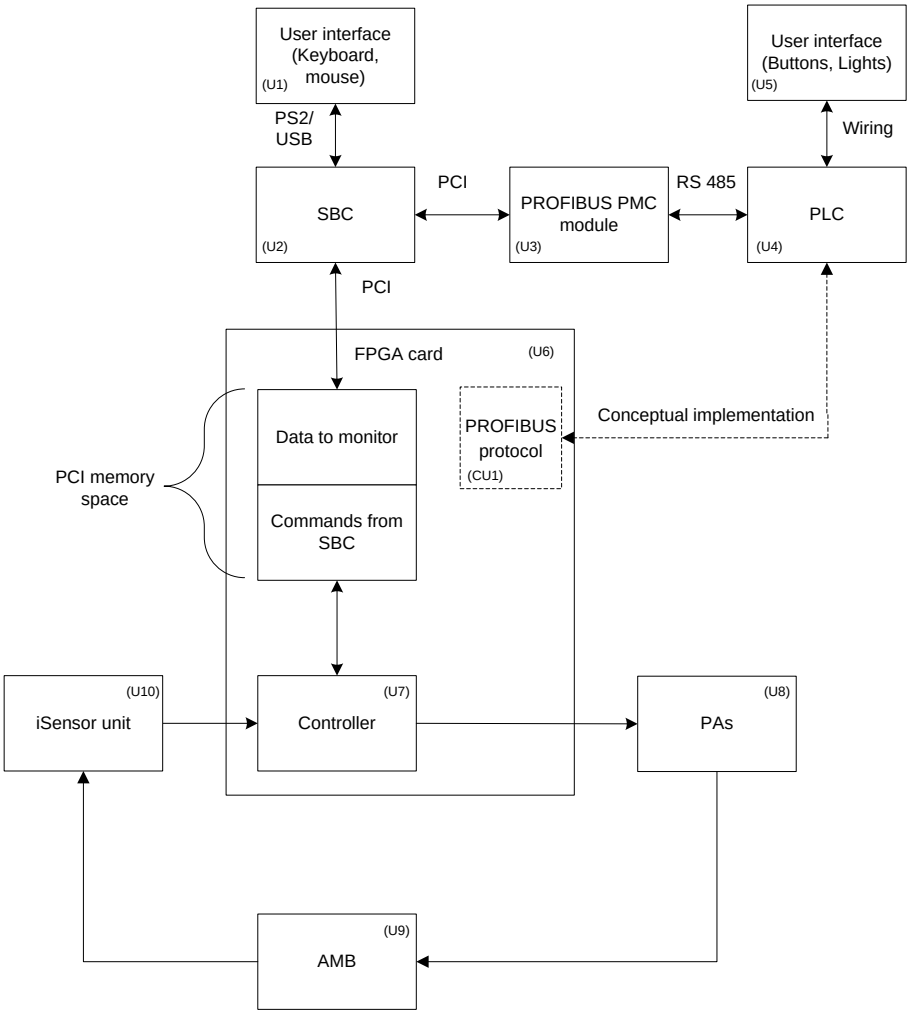


Figure 1.2 – Control flow propagation through hardware

PROFIBUS was chosen as the preferred industrial standard based on its reputation and capabilities as an industrial communication leader and its low comparative implementation cost. PROFIBUS is discussed in more detail in chapter 3. To reduce the cost of the system, PROFIBUS will be implemented on the Xilinx® FPGA. This has been done for Altera® FPGAs with the PROFIBUS standard, but no intellectual property (IP) core was available for purchase for Xilinx® FPGAs at the beginning of this study. The PROFIBUS PMC card was used at first in the system to establish the PROFIBUS network between the single-board computer and the PLC. After developing and implementing the PROFIBUS IP core on the FPGA, the PROFIBUS PMC module will be removed from the system while maintaining the PROFIBUS communication network. The PROFIBUS PMC card will facilitate the development for the PROFIBUS core on the Xilinx® FPGA. This will be done by analyzing the functionality of the PROFIBUS PMC module master device.

1.1.4 Xilinx® FPGA based PROFIBUS DP protocol implementation

An FPGA is an array of bit-processing units whose function and interconnection can be programmed after manufacturing [5]. The computation in the FPGA as well as the ASIC is done spatially [6] . An FPGA differs from the ASIC by giving the user the advantage of programming the device after manufacturing and altering the design at any time during the lifetime of the FPGA. This reconfigurable architecture achieves efficient implementation of specialized logic and provides silicon reusability [6].

By using FPGAs to implement the PROFIBUS protocol, the benefits of FPGAs will contribute to flexibility in a PROFIBUS application. At the beginning of this study no commercial PROFIBUS protocol core was available for Xilinx® FPGAs. The development of a PROFIBUS protocol for FPGAs will empower PROFIBUS users and developers to manufacture user specific and cost effective PROFIBUS designs. The PROFIBUS design can easily be reprogrammed in order to meet the requirements of a new user or to update the PROFIBUS standard. Development time of a PROFIBUS system will be lowered and companies can sustain their own maintenance.

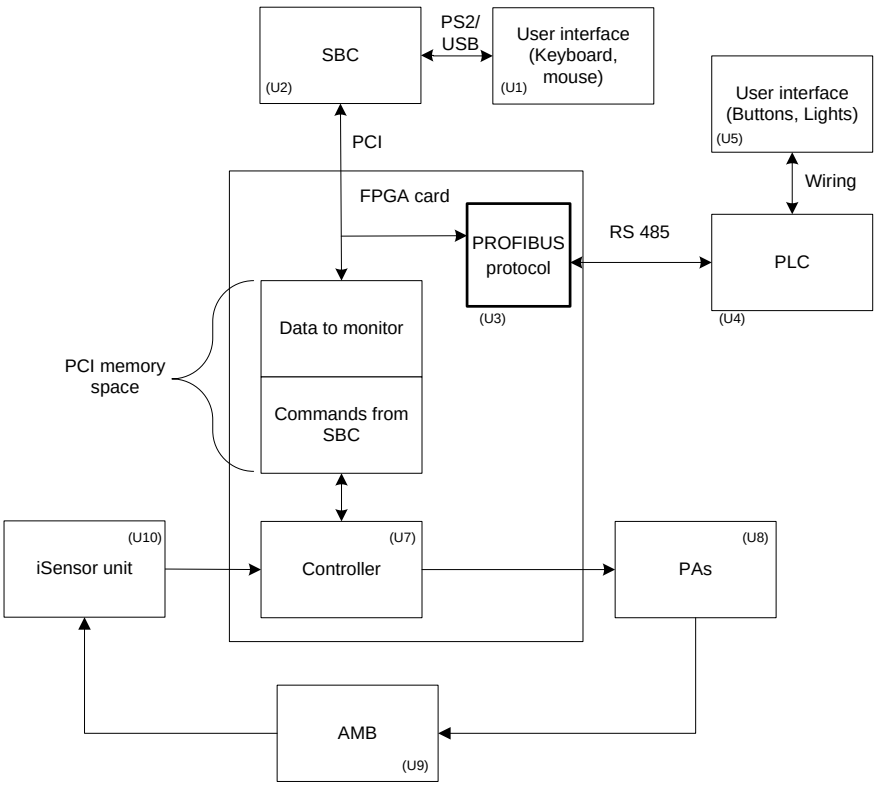


Figure 1.3 – Conceptual PROFIBUS protocol design interface

Figure 1.3 illustrates the conceptual ADES design after implementation of the PROFIBUS protocol (U3) on the FPGA. In this illustration the PROFIBUS PMC module is completely removed from the system. The developed PROFIBUS protocol will operate as a PROFIBUS master device to control the communication with the PLC slave. To achieve this, the PLC will be directly connected to the FPGA by incorporating the specified RS 485 standard as the physical layer. This implementation will reduce the cost of the ADES while maintaining the PROFIBUS standard for communication.

1.2 Problem Statement

In order to reduce the cost of the current AMB drive electronic system and to meet industrial Fieldbus standards, a PROFIBUS DP protocol has to be developed and implemented on a Xilinx® FPGA to establish a PROFIBUS DP communication network.

The AMB controller that was developed at the North-West University for the ADES utilises a Xilinx® FPGA to perform various control functions in the system. The PROFIBUS DP protocol has to be implemented on this FPGA to deal with automation communication processes in the AMB system. In order to develop a PROFIBUS DP core for the Xilinx® FPGA, the PROFIBUS standard needs to be analyzed and thoroughly understood. Methods on how to implement this standard on an FPGA, as well as evaluation methods that meet the IEC 61158 standard has to be researched. This is necessary to ensure a high quality design and to ensure problem free communication in the system.

1.3 Issues to be addressed and methodology

In order to develop a PROFIBUS DP core for a Xilinx® FPGA, a number of issues need to be addressed. These issues form the main focus of the study and establish the methods needed to address each problem. Based on the defined problem statement, this section describes how every aspect and issue of the study are addressed. Figure 1.4 describes the basic work breakdown structure and flow of the study.

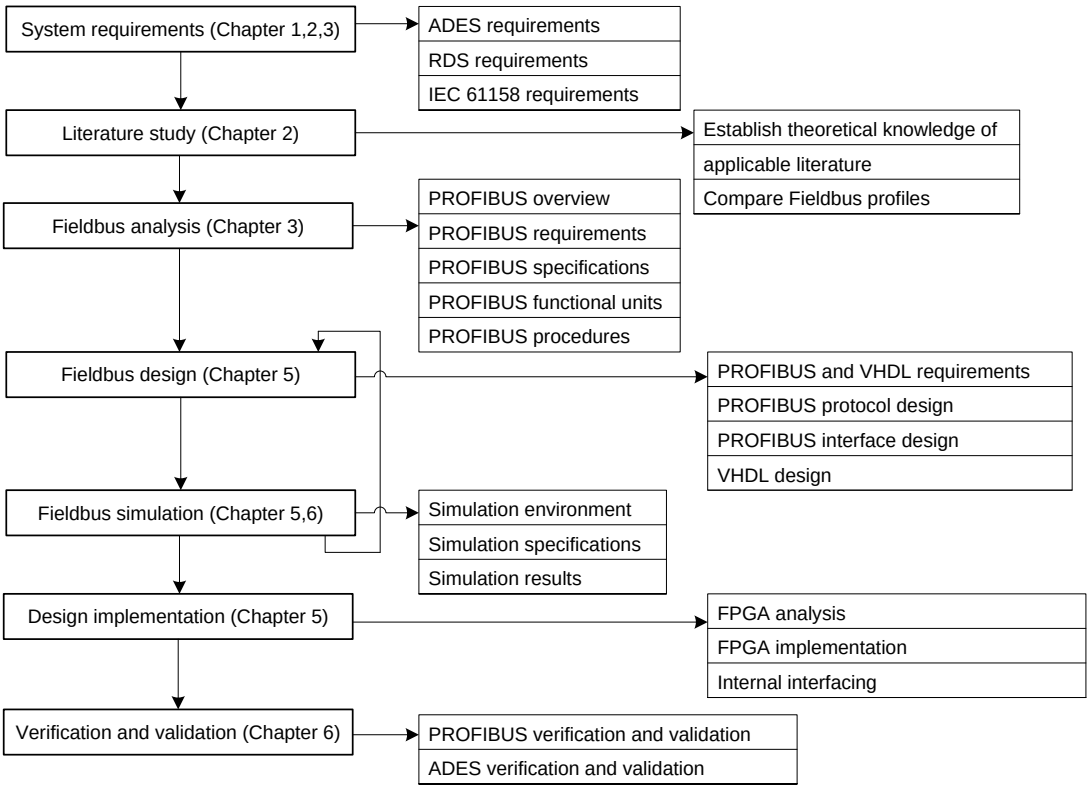


Figure 1.4 – Work breakdown structure

1.3.1 System requirements

The system requirements need to be established for the design of the PROFIBUS DP protocol on the ADES. The requirements for the design and implementation are based preliminary on the PROFIBUS DP standard specified in the IEC 61158 and IEC 61784. The specific ADES requirements for the Fieldbus communication network also need to be established. These requirements will be established in chapter 1, 2 and 3, as shown in figure 1.4.

1.3.2 Literature study

Fundamental knowledge with regards to all the aspects relating to the project needs to be established. A literature study will be conducted to thoroughly research topics of concern for this study. Once the literature study is completed, a critical literature review will be conducted to establish a relationship between the research documented in the literature study and the main problem statement. Aspects that will be researched include:

- Communication in automation
- OSI reference model
- Communication standards

- FPGAs and VHDL design
- Physical and data link layer implementation methods
- Medium access control, framing and error control
- Fieldbus systems
- IEC 61158 and IEC 61784 standards
- Evaluation methods

1.3.3 PROFIBUS DP analysis

In order to design the PROFIUBUS DP protocol for the Xilinx® FPGA, an in-depth understanding of the protocol needs to be established. The IEC 61158 and the PROFIBUS DP profile will be analysed to establish the necessary knowledge of the Fieldbus. The parts of the profile taht are mandatory together with features which will be needed for the design will be documented to form a guideline for the Fieldbus design.

1.3.4 Hardware platform

A thorough understanding of the ADES hardware platform needs to be established. As discussed earlier, the ADES is a controller based on industry standards that was developed by the McTronX research group over the past few years to control and operate the rotor delevitating system (RDS). The entire ADES will be discussed together with the initial PROFIBUS PMC module implementation. This will assist in determining the specific ADES requirements for the Fieldbus network, as well as a practical understanding of the specific PROFIBUS DP communication network. The following needs to be done to establish the initial PROFIBUS DP network when working with the PROFIBUS DP PMC module:

- The compact PCI single-board computer will be set up by means of C++ programming.
- A graphical user interface will be programmed to view sent and received data between the single-board computer and the FPGA.
- The PROFIBUS DP PMC master will be interfaced with the PLC slave used in the system, to establish a PROFIBUS DP communication network.

The purchased PROFIBUS card, in conjunction with the research that was done on the PROFIBUS standard, created the basis on how to implement the PROFIBUS protocol on the FPGA card. The following are the main focus when working with the PROFIBUS PMC card.

1.3.5 Design and implementation

After establishing the required understanding of the literature and the hardware platform, the PROFIBUS DP protocol needs to be designed in VHDL and implemented on the Xilinx® FPGA. The VHDL programming skills will be established by an introductory course and VHDL literature. The Fieldbus analysis documentation will be used as the design specification for the PROFIBUS DP protocol. This VHDL-based PROFIBUS DP implementation will be designed and simulated with the simulation software Modelsim®.

Once the design meets the PROFIBUS DP requirements and specifications, the design will be extended to meet the requirements of the ADES. This design will be implemented on the FPGA of the ADES to establish the PROFIBUS DP communication network between the FPGA and the PLC. The necessary physical layer design also needs to be designed and constructed to establish the physical link between the PLC and the FPGA. This will be designed based on the PROFIBUS DP physical layer specifications.

1.3.6 Verification and validation

Verification will be done in parallel with the design. Extensive verification will be conducted on the protocol design to ensure that all the specifications of the PROFIBUS DP protocol have been met. This will be done with Modelsim® simulations, which will simulate the real-time ADES environment.

The same tests performed during the verification will be used to validate the designed PROFIBUS DP protocol after implementation on the FPGA of the ADES. These tests will be performed while the ADES and RDS are fully operational and will ensure that the design meets the PROFIBUS DP and ADES requirements. A real-time digital oscilloscope will be used to conduct these validation tests. The physical layer implementation will also be evaluated to ensure that the communication link meets the specified requirements.

In order to establish the feasibility of this study, the developed PROFIBUS DP protocol will be compared with the commercial off-the-shelf PROFIBUS DP PMC module. This comparison is needed to determine if the developed protocol can compete with its commercial counterpart. The comparison will be done based on literature that compares different Fieldbus systems that include cost, throughput, data coding efficiency, medium access efficiency, cyclic times, responsiveness and bit error ratio.

1.4 Dissertation layout

The following chapters are included in the dissertation. Each chapter contributes to the design and implementation of the VHDL-based PROFIBUS DP protocol. Their contents are as follows:

Chapter 1 – Introduction

Chapter 1 forms an introduction of the document. Basic background on the study is discussed and the problem statement is determined. All the issues that need to be addressed in the study are discussed together with the methods on how to approach each objective.

Chapter 2 – Literature study

Chapter 2 presents a comprehensive literature study of all the relevant aspects with regards to this study. This includes communication in automation, the OSI reference model, communication standards, FPGAs and VHDL design, physical and data link layer implementation methods, medium access control, framing and error control, Fieldbus systems, the IEC 61158 and IEC 61784 standards and finally evaluation methods that will be used.

Chapter 3 – PROFIBUS standard

Chapter 3 examines and documents the PROFIBUS DP standard in detail. The standard is discussed with regards to the specifications and requirements that are needed to establish a PROFIBUS DP communication network.

Chapter 4 – Application platform

The hardware platform of the ADES environment is discussed in chapter 4, with special focus on the implementation of the PROFIBUS PMC module and work done prior to the implementation of the PROFIBUS DP protocol on the FPGA.

Chapter 5 – Fieldbus communication system design

Chapter 5 discusses the design of the PROFIBUS DP protocol and the implementation on the Xilinx® FPGA. The different aspects of the physical and data link layer design are presented. Finally the implementation of the developed PROFIBUS DP protocol on the ADES is discussed.

Chapter 6 – Verification of the designed protocol

Chapter 6 discusses the verification methods used to evaluate the designed protocol. Each aspect of the designed PROFIBUS DP protocol is verified through simulation. This chapter ensures that the designed PROFIBUS DP protocol meets the required standard before implementing the design on the ADES.

Chapter 7 – Validation of the implemented protocol

Chapter 7 discusses the validation methods used to evaluate the implemented PROFIBUS DP protocol. The same tests performed in chapter 6 are conducted, after implementation, with a digital oscilloscope. The characteristics of the implemented physical layer is also intensely validated with the use of eye diagrams.

Chapter 8 – Conclusion and recommendations

Chapter 8 gives a final overview of the developed PROFIBUS DP protocol. Based on chapter 6 and 7, a conclusion is drawn regarding the performance of the developed VHDL-based PROFIBUS DP protocol. Finally the developed protocol is compared to a commercial PROFIBUS DP module.

1.5 Conclusion

This chapter provided the necessary background knowledge of AMBs and the ADES. The problem statement was established in order to improve the ADES. The main issues that needed to be addressed together with the methods that will be used to address each issue were discussed. Finally a brief overview of the dissertation layout was given. Chapter 2 will commence by discussing the literature used for this study.

Chapter 2

Literature study

Chapter 2 presents an in-depth literature study on the various aspects of a digital communication system which relates to Fieldbus protocol design and implementation. The researched literature will be used to implement a communication protocol on a specified hardware platform that is discussed in chapter 4. The information regarding performance evaluation will be used in chapter 6 and 7 to verify and validate the implemented protocol.

2.1 Introduction

Information is considered to be power, but in fact power derives from having the information that you need when and where you need it. Modern society is much more mobile than our ancestors, and as we move about more, and trade over larger distances, the need for communication increases. On the other hand, the easier communication becomes, and the more accessible it is, the more people want to use it [7].

The route from analogue to digital communication over the last few decades has led to digital communication being a well researched and developed subject. The digital communication system and applicable components are discussed in this chapter. It is necessary to understand the entire communication system before it can be implemented. The following aspects of a digital communication system will be discussed:

- Communication systems and automation
- The OSI reference model which represent each layer of any communication system
- Physical bus characteristics and implementation possibilities
- FPGAs and VHDL programming language
- Line coding
- Medium access control of a communication network
- Error handling in communication systems

- Network topologies
- Different Fieldbus systems and the applicable standards
- Evaluation of the physical and data link layer of a communication system

2.2 Communication systems

Data communication is the problem of getting information from one place to another and ensuring no channel disruptions and deliberate interference while conforming to the user requirements [7]. The following specify some of the most common elements of a communication system [8]:

- Identification of communication traffic flows – source/destination/quality
- Overall system topology – star/mesh/ring/bus
- Device/processor capabilities
- Communication session/dialog characteristics
- Device addressing schemes
- Communication network traffic characteristics
- Performance requirements
- Timing issues
- Reliability/backup/failover
- Application service requirements
- Application data formats
- Operational requirements (directory, security and management of the network)
- Quantification of electromagnetic interference requirements

2.3 Communication systems and automation

For the last 20 years, digital communication has become a significant factor in automated distributed computer control systems within the factory as well as the process domain. This trend has to be supplemented by context awareness and is realized by location-based communication services and context-sensitive applications. The proprietary communication systems within supervisory control and data acquisition (SCADA) systems were complemented by the Fieldbus and sensor bus systems. Even though Ethernet has become a highly popular alternative for traditional Fieldbus systems, traditional Fieldbus systems still remain the most used communication systems for commercial control [9].

Communication technology is constantly evolving into more capable and efficient systems. Developing communication technologies requires the inventors to guarantee the fulfilment of the domain-specific requirements that include real-time behaviour, functional safety and security. The following requirements need to be considered [9]:

- Real-time behaviour that include diagnosis, commissioning, maintenance, slow mobile connections, process in manufacturing and process automation, data acquisition, control applications, fast mobile applications, machine tools and motion control among others.
- Guaranteeing functional safety that requires protection against hazards caused by incorrect functioning that include communication via heterogeneous networks.
- The guarantee for security requires a common security concept for distributed automation using heterogeneous networks with different security integrity levels.

Future scenarios of distributed automation will lead to desired mechanisms for geographically distributed automation functions. These functions can be summarized as follows [9]:

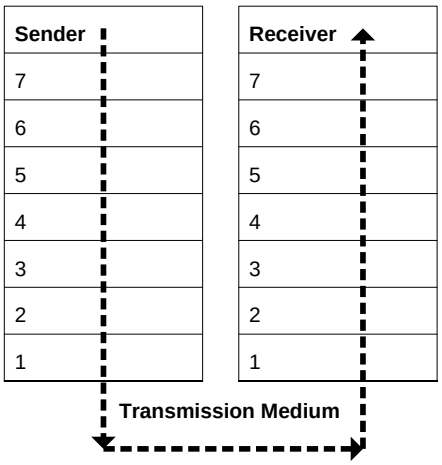
- Centralized supervisory and control of decentralized technological plants.
- Remote control, commissioning, parameterization and maintenance.
- Including remote experts or external machine-readable knowledge for the operation and maintenance of the plant.

Digital communication has been one of the most important driving forces of the computer control systems for the last 25 years. In order to make data accessible in various layers of an enterprise information system, it is needed to integrate the different digital communication systems within the plant level, control level and device level of an enterprise network. Each level has different requirements that are dictated by the nature and type of the exchanged information. The network size, number of devices supported, network speed, response time, frequency of exchange and payload size are only a few needed performance characteristics used to classify and group specific network technologies [9].

The development of industrial communication needs to address the improvement of the technology, system architecture, organization, engineering and standardization [9]. This view of industrial communication allows developers to address each characteristic separately, whilst keeping the main focal point in mind. Based on this view of industrial communication, this chapter continues by discussing various technology, architecture, engineering and standardization possibilities for communication in automation.

2.4 OSI reference model

The open systems interconnect (OSI) reference model was developed in 1979 by the international organization for standardization (ISO) in order to advance interoperability between manufacturers. By having a standard reference model that describes communications between stations of a communication system, any system which implements the OSI model will be able to connect to other OSI based systems. The aim of the OSI was to provide a foundation for standards in system communication [7]. The elements, structures and tasks required for a communication protocol are arranged in seven layers that have to fulfil specific functions within the communication process. All seven layers of the OSI model shown in figure 2.1 are not required by each communication protocol [10].



Layer	Function
Application layer	Interface to application program
Presentation layer	Representation of data for analysis and interpretation
Session layer	Synchronization of communication processes
Transport layer	Controls data transmission for layer 5
Network layer	Establishes and clears connections to avoid network congestion
Data-link layer	Handles the bus access protocol and data security
Physical layer	Definition of the hardware, speed, coding etc.

Figure 2.1 – OSI Reference Model [10]

Each layer of the OSI reference model plays a specific role in the development of communication standards. Each layer of this model will be discussed together with specific hardware and communication methods that apply to the applicable layer. Fieldbus systems normally utilize the physical, data link and application layer of the OSI reference model and therefore more emphasis is placed on the discussions of these layers.

2.4.1 Physical layer

The physical layer deals with specific mechanical and electrical means, required for data transmission [11]. This includes different application characteristics which include data rates, cabling systems and various network topologies [12] [13]. The purpose of the physical layer is to transport bit-information. The physical layer makes it possible for different manufacturers to be compatible if they use the same physical layer. The physical layer is usually discussed in relation to a specified communication system and is therefore linked to the data link layer where a certain protocol is used [11].

In order to implement a communication protocol on a certain physical layer it is necessary to understand all the components of the physical layer that will be used. The transmission medium and hardware platform form a basic part of the physical layer. The next section explores some of the available possibilities that apply to the physical layer.

2.4.1.1 Transmission media

Transmission mediums provide the platform for digital communication. A transmission medium has to transfer information in a communication network. The information is adapted to fit the specific requirements of the medium that is used. It is crucial to understand the physical properties and limitations of the transmission medium that will be used. This will ensure correctness of data transmission and device safety. The following transmission standards form part of the ADES environment.

RS 232 and RS 485

RS 232 is one of the oldest and most widespread standards for serial data transmissions. The electrical characteristics of this standard are widely known but have fundamental differences with the more popular RS 485 standard. These differences are illustrated in table 2.1. Table 2.2 shows the different characteristics of RS 485 at different data transmission speeds [11].

Table 2.1 – Differences between RS 232 and RS 485

	RS 232	**RS 485**
Transmission mode	Unbalanced (single ended)	Balanced (differential)
Driver output voltage	-15 V to +15 V	-5 V to +5 V
Receiver sensitivity	3 V	200 mV
Driver slew rate	< 30 V/µs	> 500 V/µs
Driver output impedance	> 250 Ω	< 50 Ω

Table 2.2 – Speed domains of RS 485

< 100 kbps	**100 kbps – 1 Mbps**	**> 1 Mbps**
With standard UART and non-return to zero coding, bit distortion is not disturbed by wave reflection	Wave reflection has to be avoided by adding terminating resistors equal to the line impedance. Line topology is preferred with a bus length less than 500 m.	The line length is limited by low-pass characteristics of skin effect. Line length will decrease with transmission speeds.

Compact PCI

Compact peripheral component interconnect (cPCI) is a variation of the peripheral component interconnect (PCI) specification. cPCI was developed for industrial computer applications that require a smaller, more robust mechanical form factor than the one defined for desktop computers. cPCI specifically suites small, high-speed industrial computing applications, where transfers occur between a number of high-speed cards. Compact PCI uses the Eurocard form factor, popularized by VersaModule-Eurocard (VME) [4]. Table 2.3 highlights the specifications of cPCI by comparing it to other system bus systems available.

Table 2.3 – System bus comparison [4]

Features	**VME**	**cPCI**	**PCI**	**Industry Standard Architecture (ISA)**
Address Space	4 GB	4 GB	4 GB	1 or 16 MB
Addressing method	Geographic	Slot sensitive or geographic	Slot sensitive	Board locator technology (BLT)
Time division multiplex	Bus and bandwidth American National Standards Institute (ANSI)/VME International Trade Association (VITA) 6-1994 2048 time slots std. 4096 time slots ext.	H.110 4096 time slots	H.100 up to 4096 time slots	SCbus up to 2048 time slots
Central controllers in system	Multiple	1 (master)	1 (master)	1
Max. number of channels	400 per CPU 800 with BLT	To be determined	To be determined	96
Hot swap/live insertion	VITA 1.4-199	PICMG hot swap	PCISIG hot plug	Not supported
Cabling options	SCbus and I/O ribbon cables, ANSI/VITA backplanes	Rear I/O transition modules, H.110 embedded backplanes	Bracket I/O, H.100/SCbus ribbon cable	Bracket I/O SCbus ribbon cable
System bus bandwidth	40 to 80 Mbps	132 Mbps	132 Mbps	8 Mbps
System bus bits	SCbus, CT Bus	SCbu, CT Bus	SCbus, CT Bus	Scbus

2.4.1.2 Line coding

In data transmission, the transmission medium usually cannot accept transmitted symbols in their natural form. Line codes therefore adjust digital data and transmit it in a form which is suitable for the transmission medium. The data can be conditioned in two main forms, namely baseband and modulated. Baseband refers to the direct transmission of the coded symbols in contrast to modulated transmissions where a carrier waveform is manipulated to send the signal. Each line code has the following requirements [7]:

- Timing – The waveform of the line code should be easy to synchronize, to avoid timing problems
- DC content – Line codes should be able to be transmitted through alternating current coupled media
- Power spectrum – The power spectrum should be as small as possible for transmission efficiency
- Performance monitoring – To detect errors
- Low error probability
- Transparency – The code should work for any pattern of symbols
- Complexity – Encoding and decoding should be easy to reduce cost
- Uniqueness – By decoding transmissions, data should be easy to be identified

Based on these requirements there are two basic forms of line codes [7]:

- Level codes – Information is carried in the voltage or current level of the signal
  - Non-return to zero (NRZ) – pulse level maintained throughout the signal
  - Return to zero (RZ) – the level returns to zero at the end of each symbol
- Transition codes – Information is carried in the change in level of the voltage or current of the signal.
  - Manchester coding – Each data bit has at least one transition during the duration of the bit

2.4.2 FPGA and VHDL

A field programmable gate array (FPGA) is an array of bit-processing units whose functions and interconnections can be reprogrammed after fabrication, contrary to application specific integrated circuits (ASICs) and digital signal processors (DSP). Small lookup tables that are wired together with a programmable interconnect are used as programmable computational elements. This allows

a user to program the FPGA after manufacturing to meet a specific need. FPGAs were originally designed as user-programmable alternatives to fixed gate arrays. Researchers have successfully used FPGAs for accelerated computing applications with impressive performance. Configurable computers have proven themselves to be the fastest and most economical way to solve problems. Examples have been seen where FPGAs are able to complete computations in 1 cycle, which took traditional processors tens of hundreds of cycles [5].

Using FPGAs is referred to as configurable computing since the computation is defined by the configuration bits in the device which inform each gate and interconnect how to operate. FPGAs are programmed after fabrication and can solve any computational task that fits into the device's finite state and operational resources. This allows the user to configure the FPGA in such a way that it will be dedicated to meet specific functions. FPGAs can complete more work than traditional processors or DSPs for the following reasons [5]:

- FPGAs have less instruction overhead allowing the device to pack more active computations onto the same silicon area as the processor. This allows the FPGA to perform more parallel computing per cycle compared to other processor devices
- With FPGAs, operations can be controlled at the bit level, where traditional processors can only control operators at word level. Traditional processors waste computational capacity when operating on narrow-width data.

VHDL

Very high-speed integrated circuits hardware description language (VHDL) was standardized by the American government in 1986 as a hardware description language for military systems. VHDL has been revised and is now a rich versatile language that can be used for synthesis, modelling and simulation [14].

VHDL is intended to be used for circuit synthesis as well as circuit simulation. The motivation for using VHDL lies in it being a standard, technology/vendor independent language that is portable and reusable. VHDL can be used in the field of programmable logic devices or in the field of ASICs. After the VHDL code has been developed, the code can be implemented as a circuit in a programmable device, or can be fabricated on commercial ASIC chips. A main distinction that VHDL has over regular computer programs lies in the fact that VHDL statements are inherently concurrent (parallel). Figure 2.2 illustrates the VHDL design flow from origin to placement on a physical device [15].

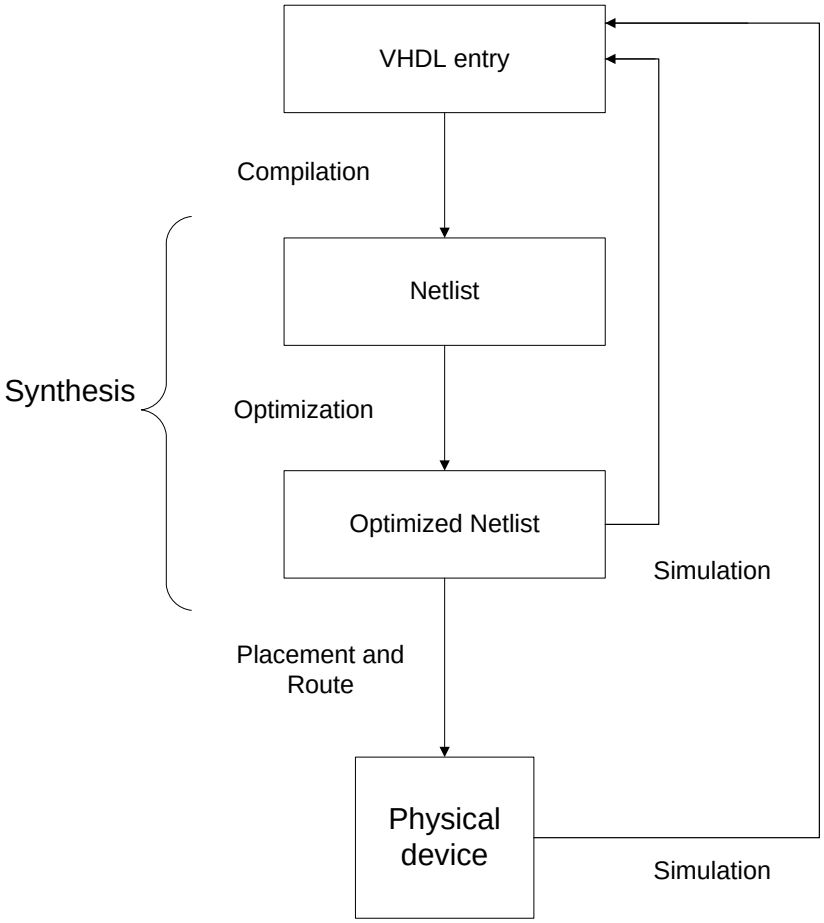


Figure 2.2 – VHDL design flow

2.4.3 Data link layer

In regards to the physical layer, which handles bit information, the data link layer can be considered as the layer that handles the combination of these data units. The data units that are transmitted are passed to the physical layer by the data link layer. Errors that may have occurred by signal interference is detected and corrected in the data link layer [11]. Different medium access control protocols are therefore implemented in different Fieldbus standards and are based on whether controlled access is used or not [12] [13]. Besides providing the network layer with a connection between two terminals, the data link layer has a number of functions that include medium access control, framing and error control [7].

2.4.3.1 Medium access control

In order to use a transmission medium at its full capacity, the transmission medium must be able to allow more than one device to use the medium. This is done by statistical multiplexing. Statistical multiplexing means the sharing of a common transmission medium in an efficient strategy. Since the medium is shared, terminals connected to the medium have to be given permission to use the medium when they need it, given the use of the other terminals. This task is handled by the medium access control. The requirements of any medium access control (MAC) include the following [7]:

- The medium access control method should be reliable
- The protocol should be stable
- Fair access should be given to each terminal in the system
- Access to the system should be controlled based on the needs of the services or any other specified reason
- The medium access control should be efficient and easy to manage

A number of different MAC methods are possible, based on the resources of the network. These possibilities are categorized under centralized and distributed schemes as well as contention-based (ALOHA, Slotted ALOHA, CSMA/CD) and reservation-based schemes (Polling, Token-based) [7]. In table 2.4 the main medium access control methods together with their distinctive features are listed.

Table 2.4 A – Medium access control methods

MAC	**Features**
ALOHA	Simplest possible contention-based scheme If a terminal wants to transmit data without any authorization, it transmits it. If another terminal transmits during this time, both transmissions will be corrupted since the transmissions collide on the medium. The transmitting terminal receives an acknowledgment if its transmission has reached the desired node. If no acknowledgement is received the terminal will wait a random period of time and re-transmit. ALOHA does not require any centralized control and is very easy to manage. It is however very inefficient since an average of only 18% of the channel capacity is used [7].

Table 2.4 B – Medium access control methods continued

MAC	**Features**
Slotted ALOHA	Slotted ALOHA divides transmissions into frames, to reduce the chance of collision. This reduces the chance of collision and increases the throughput to a maximum of 36% of the channel capacity. Part of the most medium access control schemes for mobile radio systems [7].
CSMA/CD	Carrier sense multiple access with collision detection (CSMA/CD) is used for the popular Ethernet and other networks. The transmitter will 'listen' before transmitting, not transmit while another terminal is transmitting, not transmit too long and stop transmitting when it is detected that another terminal is transmitting at the same time. Much higher throughputs are achieved. Collisions still occur because of the time it takes a frame to propagate the length of the bus. A minimum packet length on a CSMA/CD system that is equal to two times the propagation time over the bus length and some additional processing time is specified to avoid collision [7].
Polling	A central controller is used to permit access to the shared medium. Devices are permitted to use the medium for a certain period of time after being polled by the central controller. The time slot for transmission is limited to allow each host to have a fair chance to transmit. A disadvantage of this scheme is that each host is polled whether or not it has data to transmit. If the host has nothing to transmit, it immediately signals this to the controller which polls the next host in the queue [7].
Token-based scheme	Token-based schemes are based on the polling scheme that occurs in a distributed manner, thus removing the central controller. The token is passed from node to node where permission to transmit is based on which node has the token. The token time is limited to ensure fair access to each node. This process continues in a cycle until all nodes have received the token after which the cycle repeats. Most common token-based scheme is the IEEE 802.5 Token Ring MAC standard [7].

2.4.3.2 Framing

Framing refers to the function of the data link layer that allows different parts of the information and control signals to be recognized. Data is broken down into different frames that are sent over a communication network. The start and end frame must be marked, especially when the transmission medium is shared. Different framing possibilities are available. For example, the use of a field in the header of the frame, that specifies the number of characters in the frame, or marking the start and end of each frame with a specified value or a unique bit sequence [7]. Different standards apply different framing techniques based on their specifications and requirements. Table 2.5 illustrates the framing specification of the PROFIBUS DP standard [16].

Table 2.5 – PROFIBUS-DP framing specification

Character	**Character name**	**Purpose**
SD	Start delimiter	Identifies the beginning of a frame
LE	Data length	Length of the data in the telegram
LEr	Length repeated	For redundancy
DA	Destination address	Address of receiver
SA	Source address	Address of sender
FC	Function code	Specifies the type of telegram (request, response etc)
DSAP	Destination service access point	Tells the receiver what data is to be transmitted and what function is to be performed
SSAP	Source service access point	Tells the receiver what data is to be transmitted and what function is to be performed
DU	Data units	The specific data units of the frame
FCS	Frame check sequence	Sum of the bytes from DA to DU without the overflow. (Checks the correctness of the received frame)
ED	End delimiter	End of the frame

2.4.3.3 Error control

The data link layer aims to provide reliable data to the network layer. This requires the use of error correction and detection. If an error is introduced in a message, the received message will not be equal to the transmitted message. Error correction coding is included in a transmission system to protect against these errors. Error control can be implemented through the following methods [7]:

- **Error concealment**: Errors are detected and the corrupted information is identified and discarded. The remaining information is used and in some cases can be used to mask the missing data. This is done by repeating a previous message, muting the corrupt signal or by interpolating from the surrounding values.
- **Automatic repeat request (ARQ)**: Errors are detected and a request is sent to the transmitter to resend the data. The disadvantages of ARQ are that the transmitter has to store the sent data until it is informed that is was correctly transferred. A feedback channel is required to the transmitter and if an error occurs, a delay is introduced.
- **Forward error correction (FEC)**: FEC adds additional message symbols to the message being sent to allow the receiver to reconstruct the message if part of the message has been corrupted. Although FEC is computationally more complex and requires additional transmission overhead, it does not require a feedback path to the transmitter. FEC therefore adds little delay time in transmission.

Error detection

In order to respond to errors, a system has to first detect if an error occurred. The frame that was received can either be corrupted with errors or the frame may not arrive at all. The latter can be overcome by including sequence numbers in each frame that can be used to detect if a frame was 'lost'. Detecting errors is more complex since a function is calculated using the information in the message and added to the frame as a frame check sequence (FCS). Upon arrival of a frame, the receiver calculates the same function and compares it to the FCS. Errors within the frame can be detected in this manner. A problem with the FCS is than an error can occur in such a manner that the FCS function still consider the transmitted telegram as correct. This error can be overcome by careful design of the cyclic redundancy checks (CRC) which form the centre of the FCS. CRCs are discussed later in this section [7].

Probability of errors

The probability of errors in transmission can be calculated by considering information through a binary symmetric channel with transition capability *a.* The binary symmetric channel has no memory and the probability of an error is the same for both 0_2 and 1_2. The probability that *n* bits are received without an error is $(1-a)^n$. Furthermore, the probability that one error occurs is the probability that a given bit is erroneous with all the other bits (which is $a(1-a)^{n-1}$) times *n.* This is because there

are *n* possibilities for the specific bit, which is an error. A more general formulation is: the probability that *e* bits are in error from the *n* bits can be written as [7]:

$$\binom{n}{e} = \frac{n!}{e!\,(n-e)!} = \frac{n(n-1)\ldots(n-e+1)}{i!} \tag{2.1}$$

Parity checks

Parity check code is a binary code that adds an additional bit to each message, to ensure that there is an even or odd number of 1s in the message. When data is transmitted, the amount of 1s in the message is counted. With even parity a 0_2 is added to the message if an even number of 1s is present in the message. Otherwise a 0_2 will be added. Odd parity on the other hand will add a 1_2 when an even amount of 1s is present and a 0 if an uneven amount of 1s is present in the message. The code only detects the error and is not capable of any correction. The disadvantage of this method is when more than one bit in the message has the wrong value [7]. Parity checks can easily be implemented in design by counting the amount of outgoing and incoming messages. The sum of the 1s is modulated by 2 to detect even or odd parity.

Hamming code

In order to resolve an error, a more scientific method would be to have a codeword that delivers enough information to contain the message as well as the error. Consider the need to correct a single error in a three-bit code. There is a possibility that an error can occur in any of the three bits or no error at all. There are *3 + 1 = 4* possibilities of error messages that can occur. It takes two bits of information to encode this and the number of parity bits added to the code must equal at least two. There are three bits in total *(n=3)* with *k*, the number of information bits, equals *1*. This theory can be extended if there are *n* bits in the codeword. The amount of parity bits would be $log_2(n+1)$ with $k = n - log_2(n+1)$. Therefore codes exist with $n = 2^q - 1\ and\ k = n - q$. These codes are called Hamming codes. The Hamming distance is equal to the number of coefficients, where the bit streams differ and is called the weight of the two bit streams. The weight of a bit stream is the number of non-zero components is has. The higher the Hamming distance, the more transmission security is built into the communication network. [7].

Cyclic redundancy checks

A cyclic redundancy check (CRC) is a way of providing error control coding by introducing some redundancy in the data in a controlled fashion. It is commonly used and a very effective way of detecting transmission errors during transmissions in various networks [17].

Common CRC polynomials can detect the following types of errors [17]:

- All single bit errors
- All double bit errors
- All odd number of errors
- Any burst error if the burst length is less than the polynomial length
- Most large burst errors

The CRC procedure is described in (2.2) to (2.3) [17]:

$$V(x) = S(x) + x^{n-k}U(x) \tag{2.2}$$

V(x) is the transmitted data word with length *n*. It consists of the original data *U(x);* followed by a codeword *S(x)* called the CRC-sum. *S(x)* can be computed by using equation 2.3.

$$x^{n-k}U(x) = a(x)g(x) + S(x) \tag{2.3}$$

S(x) can be seen as the remainder that results from a division of the data stream and a generator polynomial *g(x).* The coding procedure will be the same on the receiving and transmitting end of the line. The CRC encoding/decoding principle is best illustrated by figure 2.3 [17].

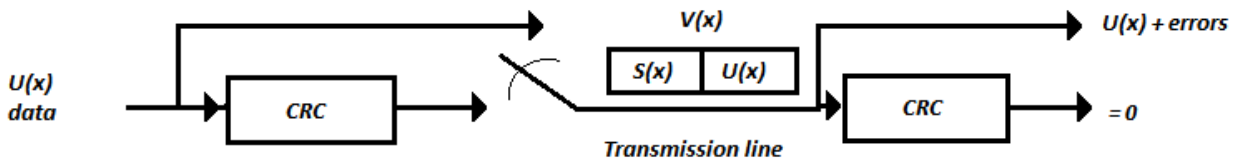


Figure 2.3 – Error detection principle using the CRC algorithm

In figure 2.3 the receiving side of the network performs a CRC-check on the incoming message. If the result equals zero, no error occurred. If the result is non-zero, an error occurred and the receiver will have to resend the message [17].

CRC is widely used in data transmission to detect errors. However, most CRCs do not have the inherent capability of correcting multi-bit errors because of their algorithm. A successful error correcting CRC depends on two aspects. Firstly, an efficient confidence declaration should be applied with a low misjudgement rate without degrading the performance. Secondly, the digital processor on the receiver side should have enough throughput to fulfil confidence declaration [18].

2.4.4 Network layer

“The network layer provides all the means you need to route data from one application to the other in an open communication system” [11]. Considering that often only single communication paths are available between Fieldbus station, the network layer is usually not included in most Fieldbus architectural models [12].

2.4.4.1 Network topologies

Different network topologies are available when connecting devices to communicate with each other over a network. The manner in which the devices of the network (nodes) are connected describes the topology. The four basic network topologies are mesh, star, ring and bus [7]. Each topology has different characteristics and application possibilities.

Mesh topology

Mesh topology, as seen in figure 2.4, is where the nodes of the network are directly connected to each other. The nodes can either be connected to each other node (full mesh) or only some of the nodes in the network (partial mesh). Partial mesh requires some nodes to communicate via other nodes in order to transfer data to the correct node on the network. When full mesh is used, no switching technique is used since each node can communicate directly to each other node on the network. However, this can drastically increase the cost of connection when a large mesh network is implemented [7].

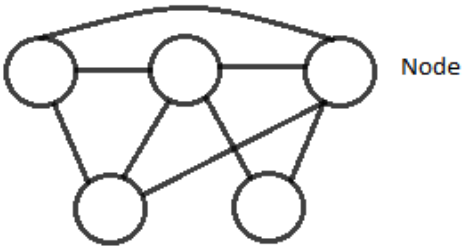


Figure 2.4 – Mesh topology

Star topology

Figure 2.5 represents star topology in which each node is directly connected to a central node. The central node is responsible for routing communications between nodes. This places all the demands of the nodes in the network on the central node, resulting in the need for a complex and expensive device. The reliability of the central node determines the reliability of the entire network. Redundant central controller devices are used to keep the network from failing [7].

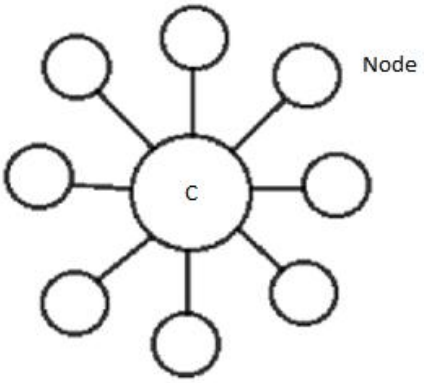


Figure 2.5 – Star topology

Ring topology

The ring network shown in figure 2.6 uses an arrangement where all the nodes of the network are joined point-to-point in a closed loop. Information is passed from node to node until it reaches the desired destination. The aim of the ring network is to distribute processing power over all the nodes in the network. Each node has the ability to make decisions regarding data transfer. Since there is no central controller, this topology also allows for complete failure should one of the nodes fail [7].

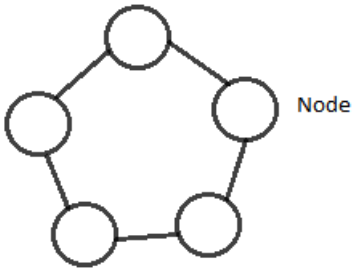


Figure 2.6 – Ring topology

Bus topology

Bus topology, as illustrated in figure 2.7, is an example of a fully distributed network. This bus-based network has no central processing, where the actual communication network is the transmission medium itself. Each node connects directly to the cable or bus by using the specified hardware interface. This network can be extended by adding an additional number of busses to form a tree-network. The network is independent from devices attached to it should a node fail. This requires an access control method to determine when the nodes can transmit data over the bus [7]. This was discussed previously in this chapter.

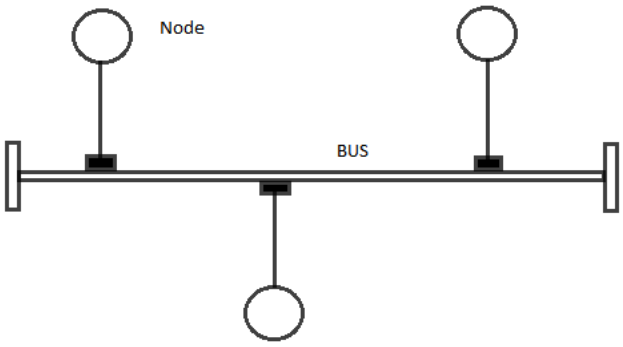


Figure 2.7 – Bus topology

2.4.5 Transport layer

The transport layer controls exchanges between two stations that is not dependant of network infrastructure [7] [11]. The messages that are transmitted are cut into small packets and transmitted individually. This layer deals with addressing, connection set-up, flow control and crash recovery [7]. Since the transport layer is not present in a Fieldbus system, the data link layer handles the necessary responsibilities [7].

2.4.6 Session layer

The session layer usually organizes and synchronizes the exchange of large messages. Nevertheless, this session layer has no role in most Fieldbus systems [12] [11].

2.4.7 Presentation layer

The presentation layer acts as an interconnection when different syntax is used for data transfer [11]. It can be said that the presentation layer provides a common language that can be used to exchange data between locations that uses dissimilar languages [12].

2.4.8 Application layer

The specific user interface for the communication system is usually seen as the application layer and sometimes referred to as the user layer [12] [7]. The term user layer is used to describe how the user will see the Fieldbus system and the corresponding communication [12]. It can be said that the application layer handles the specific coding of the data into variable structures [11].

2.5 Fieldbus basics

“Conceptually, a Fieldbus is a digital, serial, multi-drop, data bus for communication with industrial control and instrumentation devices such as — but not limited to — transducers, actuators and controllers” [19].

A general definition for a Fieldbus system has been established as a communication network used to connect field devices such as discrete and analogue sensors, transducers, controllers, actuators, field controllers such as PLC’s, regulators, drive controllers, multi-loop controllers, hand held

communicators and man-machine interfaces [12] [20]. Fieldbus systems were invented in the early 1970's, while the real development started in the 1980s [21]. From the mid 1960's, the 4-20 mA analogue current signal was used as the standard communication for process instrumentation [20]. Fieldbus systems were initially intended to replace the 4-20 mA analogue signal with an international digital communication standard for a distributed control system [20]. Currently a large number of global Fieldbus standards are available for consumers. In order to compare these different Fieldbus systems it is necessary to consider the history of the IEC Fieldbus standard, how each Fieldbus was developed and for which purpose was it intended.

2.5.1 Fieldbus history

In the 1980's a large amount of Fieldbus systems were being developed by companies in the automation business. Different approaches to the various applications led to an overwhelming number of different systems that were incompatible. The large companies realized that much more could be benefitted from making the Fieldbus specifications publicly available, in order for different vendors to manufacture compatible devices. The transition between creating an open Fieldbus specification, to producing a standard for Fieldbus systems in a formal way, ruled out the possibility of quick changes. The establishment of this standard ensured the customer of a more reliable and stable Fieldbus system that has commercial advantages over non-standardized rivals [21]. Table 2.6 shows the major activities that took place during the standardization of Fieldbus.

Table 2.6 – Fieldbus standardization timeline [20]

Duration	**Major activities**
1986-1990	Fieldbus systems were mainly developed by European countries with German PROFIBUS and French FIP as the main candidates.
1990-1994	WorldFIP and the Interoperable System Project (ISP) were developed to try and solve the global problem.
1995-1998	American companies responded to the struggle between the European companies and defined Foundation Fieldbus (FF) as a Fieldbus for the process industry.
1999-2000	The main Fieldbus contenders (Fieldbus Foundation, Fisher Rosemount, ControlNet International, Rockwell Automation, PROFIBUS user organization and Siemens) signed a "memorandum of Understanding" to end the Fieldbus war and create a comprehensive IEC 61158 standard that accommodates all the Fieldbus systems.
2000-2002	The standard is enhanced and specified in IEC 61784

This standardization introduced a new race for Fieldbus standardization, which caused a lot of difficulty in establishing international solutions. The period between 1986 and 2000 saw the development of the Fieldbus systems that were standardized in the IEC 61158 on 31 December 2000 [21].

2.5.2 Fieldbus basics

In order to compare different Fieldbus systems, it is necessary to understand the operation of their procedures and different communication aspects. This can be evaluated with the OSI reference model that was published in 1984 as the international standard reference model of open systems interconnection (OSI). The standard was published by the technical committee in the international organization for standardization (ISO). This standard was intended to allow newly developed standards to be placed into perspective with existing standards [11]. It is commonly said that Fieldbus utilizes only three layers of the OSI model namely the physical layer, data link layer and the application layer. The applicable Fieldbus layers of the OSI model will therefore be implemented and evaluated in detail. Table 2.7 shows an overview of the seven layer OSI reference model [7].

Table 2.7 – Seven layer OSI model

1	Application	Upper Layers
2	Presentation	
3	Session	
4	Transport	
5	Network	Lower Layers
6	Data Link	
7	Physical	

2.6 IEC 61158 standard

“The international electrotechnical commission (IEC) is a worldwide organization for standardization, comprising all national electrotechnical committees (IEC National Committees)” [19]. The IEC aims to promote international co-operation on all questions concerning standardization in the electric and electronic fields. The IEC provides various other services as well as publishing international standards. The preparation of the standards is done by technical committees as well as

international, government and non-governmental organizations liaising with the IEC. The IEC also collaborates with the ISO when developing standards. The standards are produces in document form and published under standards, technical specifications, technical reports or guides. Even though the IEC ensures the accurateness of the published standards, they are not liable for any errors in the standards. The main task of the IEC technical committees remains to prepare international standards [19].

Effective communication in global markets requires global understanding of a specification which may include a standard. The understanding between IEC and OSI provide a common basis for understanding between international manufactures and end-users. The IEC 61158 series provides specifications regarding Fieldbus systems. This standard works closely with the IEC 61784 which provides users and manufacturers with details of supported Fieldbus specifications, that are based on selected options, that are presented with the intent to work together consistently and correctly [19].

According to the IEC 61158 the benefits of using an international common and formal style to specify the communication system include [19]:

- The common look and feel of a specification reduces effort during evaluation
- The common structure helps users and manufacturers to identify and specify common parts and contents
- The universal approach ensures long-term quality and stability
- Identification of missing parts and items of any specification are more easily identified by comparison with other specifications, leading to simplified review and evaluation procedures
- These modular concepts support future enhancements, extensions and adjustment of new technologies

2.6.1 Components of the IEC 61158 standard

The IEC 61158 standard is used to define standards for Fieldbus systems, for use in industrial control systems. This standard falls under the general title of the IEC – Digital data communications for measurement and control. The IEC 61158 standard is divided into 6 parts [19]:

- Part 1: Overview and guidance for the IEC 61158 series
- Part 2: Physical layer specification and service definition

- Part 3: Data link service definition
- Part 4: Data link protocol specification
- Part 5: Application layer service definition
- Part 6: Application layer protocol specification

Each part of this standard addresses a different component of Fieldbus systems, with relation to the OSI reference model, that is defined in “ISO/IEC 7498-1, Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model”. These protocol types have been engineered to support information processing, monitoring and control system for any part or domain of the industrial sector. Figure 2.8 illustrates an example of the application of high-integrity low-level communication between sensors, actuators and local controllers in a plant that is interconnected with programmable controllers [19].

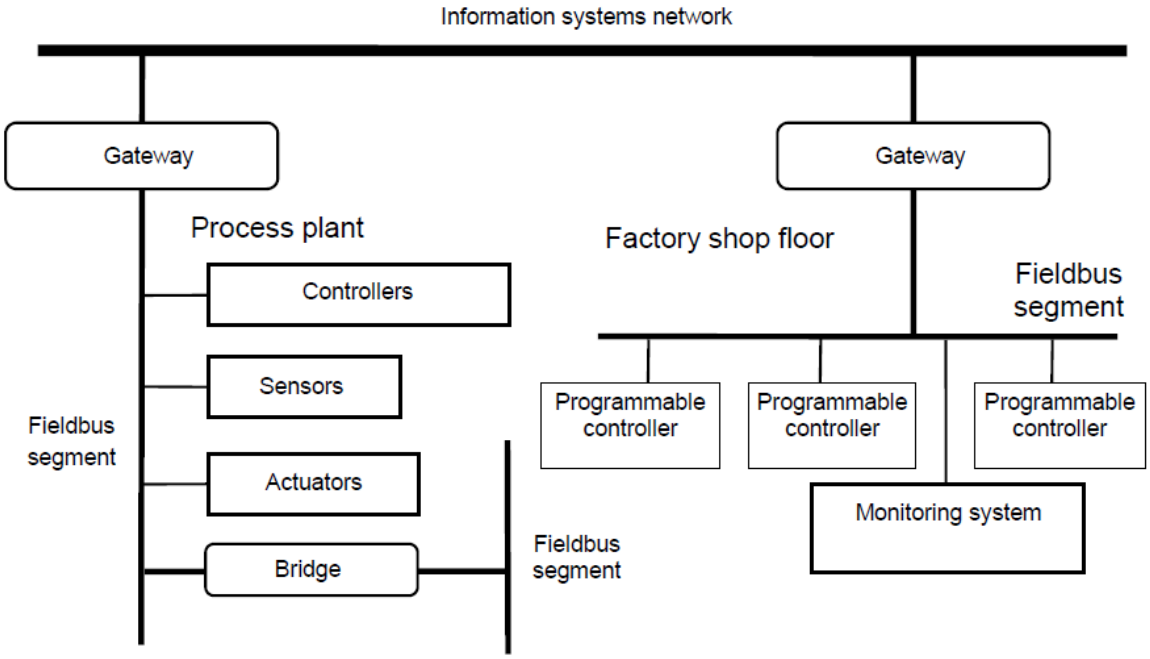


Figure 2.8 – Generic Fieldbus network [19]

The IEC 61158, as a standard, addresses three primary concepts for decomposition. Firstly, the complex communication task is divided into different layers according to the OSI reference model discussed earlier. This helps with the decomposition of complex tasks as well as modular structure that can be adapted to different technologies. Secondly, the Fieldbus type is composed of one or more layer specifications. Most Fieldbus types have various services and protocol options that require an appropriate selection. Selections that are compatible with these options and services are

specified as standardized communication profiles in the IEC 61784. The third concept includes the physical, data link and application layers that are described in complementary ways, in terms of the offer services and the protocol which provide the services [19].

2.6.2 Layers in the IEC 61158 standard

The protocol types that are described in the IEC 61158 are based on the principles, methodology and model of the ISO/IEC 7498 that describes the OSI reference model. As discussed earlier, the OSI model provides a layered approach to communication standards whereby the layers can be implemented and modified independently. Table 2.8 illustrates how the OSI model corresponds with the IEC 61158 layers [19].

Table 2.8 – OSI and IEC 61158 layers [19]

OSI layer	**Function**	**IEC 61158 layer**
7 – Application	Translates the demands placed on the communications stack into a form that is understood by the lower layers and vice versa	Application (IEC 61158-5, -6)
6 – Presentation	Converts data to or from standardized network formats	
5 – Session	Synchronizes and manages data	
4 – Transport	Provides transparent and reliable data transfer	Data link (IEC 61158-3, -4)
3 – Network	Performs message routing	
2 – Data link	Controls access to the communication medium and performs error correction	
1 – Physical	Encodes and decodes signals for transmission or reception in a form appropriate to the communications medium and specifies the communication media characteristics.	Physical (IEC 61158-2)

The IEC 61158 physical layer

The IEC 61158 physical layer receives data units from the data link layer and encapsulates them if necessary, by adding framing information and encoding the bits and framing information into signals. The resulting physical signals are transmitted via the transmission medium connected to the

transmitting node. The transmitted signals are received at one or more nodes and decoded by removing and checking framing information. The data units retrieved from the framing units are then passed to the data link layer of the receiving device [19].

The IEC 61158-2 consist of physical layer specifications to support the protocol types, specified in the IEC 61158 data link layer. The following services are provided [19]:

- The services provided to the various types of Fieldbus data link layers, at the boundary between the data link and physical layers of the Fieldbus reference model
- The services provide to systems management, at the boundary between the physical layer and systems management of the Fieldbus reference model.

The IEC 61158 data link layer

When persistent errors occur, the IEC 61158 data link layer provides time-critical support for data communications among devices in an automation environment. Time critical applications have a time-window, within which one or more specified action needs to be completed with a defined level of certainty. If the specified actions are not completed in the time-window, risk of failure or damage to applications or equipment exists. The IEC 61158-3 specifies the externally visible services provided by the Fieldbus data link layer that include the following [19]:

- The primitive actions and events of the service
- The parameters that are associated with each primitive action and event
- The valid sequences and interrelationship between these actions and events
- The services of the various Fieldbus application layer boundary between the application layer and data link layer of the Fieldbus reference model
- The services provided to the system management at the boundary between the data link layer and the system management of the Fieldbus reference model

The IEC 61158 application layer

The application layer of the IEC 61158 supports the conveyance of time-critical application requests, and responses between different devices in an automation environment. The IEC 61158-5 specifies the interactions between remote applications in terms of an abstract model for defining application resources, that can be manipulated by users with the use of Fieldbus application layer

(FAL) services. The primitives associated with each FAL service, the different parameters of each primitive and the interrelationship between the primitives for each service is also specified. The following services are included in the IEC 61158-5 [19]:

- The services for the various users of the Fieldbus application layer, at the boundary between the users and the application layer of the Fieldbus reference model
- The services to the system management, at the boundary between the application layer and the system management of the Fieldbus reference model

Various different Fieldbus systems are discussed in the IEC 61784. Each profile provides a different solution based on the IEC 61158 standard. The following section will discuss the most popular Fieldbus systems and draw a comparison between their similarities and features.

2.7 Fieldbus systems

Fieldbus is considered as the lowest level industrial network in computer communication hierarchy of factory automation and process control. The benefits of the Fieldbus approach to inter-device communications in an industrial environment include lower costs, as a result of multiple point-to-point links that are replaced by a shared bus and the enhancement of the performance of field devices by removing the demand for central computing [22]. The lower cost of Fieldbus and the evolution of this technology has led to Fieldbus gaining more popularity in communication between any type of controller and controlled devices where point-to-point links and analogue signals were traditionally used [23]. Table 2.9 draws a comparison between Fieldbus systems and other networks. In table 2.10 the properties of a few popular Fieldbus profiles of the IEC 61158 and newer additions are compared. For a more detail discussion of the Fieldbus profiles, refer to appendix E.

Table 2.9 – Comparison between Fieldbus and other networks [11]

	Fieldbus	**Other network (LAN)**
Typical application	Measurement and control, automation, control loops	Office, management, visualization
Typical user data	Strings < 100 bytes	Files > 1 Kbyte
Typical reaction time	<< 1 second	>> 1 s
Typical type of station	Sensors, actuators, devices	Computers, printers, multi-media
Real-time requirement	strong	weak

Table 2.10 A – Fieldbus profiles comparison [22] [24] [25] [26] [27] [28]

Fieldbus	**Transmission rates**	**Data transfer size**	**Network topology**	**Error checking**	**Medium access control**	**Physical media**	**Maximum devices (per segment)**	**Maximum distance (without repeaters)**
Foundation Fieldbus	H1: 31.25 Kbps HSE: 100 Mbps	128 bytes	Star, bus	16 bit CRC	Scheduling, multiple backup, CSMA/CD	Twisted-pair, optic-fibre	H1: 32, HSE: unlimited	H1 < 1900 m @ 31.25 kbps HSE: 100 m @ 100 Mbps
ControlNet	5 Mbps	0 to 510 bytes	Star, tree, bus, or combinations	Modified CCITT with 16-bit polynomial	Concurrent TDMA	Coaxial cable, optic-fibre	99	1000 m with 2 nodes. 250 m with 48 nodes
PROFIBUS	PA: 31.25 kbps DP/FMS: 9.6 kbps to 12 Mbps	0 to 244 bytes	PA: mixed bus/tree DP/FMS: bus	CRC, Hamming distance 4	Token passing and Master/Slave	Shielded twisted pair, optic-fibre	PA/DP/FMS - 127 nodes	PA < 120 m DP/FMS – 100 m @ 12 Mbps
Industrial Ethernet	10, 100 Mbps	46 to 1.5 kb	Bus, star, daisy-chain	CSMA/CD	CRC 32	Coaxial, twisted pair, optic-fibre	1024	Twisted pair -4 km Fibre – 2.5 km
WorldFIP	31.25 Kbps, 1 Mbps, or 2.5 Mbps, 5 Mbps	128 bytes	Bus	16-bit CRC	bus contention, centralized polling	Twisted pair, optic-fibre	256	40 km

Table 2.10 B – Fieldbus profiles comparison [22] [24] [25] [26] [27]

Fieldbus	Transmission rates	Data size	Network topology	Error checking	Medium access control	Physical media	Maximum devices	Maximum distance
ARCNET		19.53 kb to 10 Mb	Star, bus	16 bit CRC	Token passing	Coaxial, twisted pair, optic-fibre	255	Coax 2000 ft, Twisted pair 400 ft feet, Fibre 6000 ft
Interbus-S	500 Kbps	1 to 64 bytes	T-drops	16-bit CRC	not required	Twisted pair, optic fibre	256	400 m
CANopen	125 kbps to 1 Mbps	0 to 8 bytes	Bus terminated	Elaborate error handling scheme	Non-destructive CSMA/CD, bit-wise arbitration	Twisted pair	127	1000 m
Devicenet	125 kbps to 500 kbps	8 bytes	Bus	CRC check	Prioritized, Peer-to-Peer communication	Separate twisted pair for signal and power	64	500 m
Seriplex	200 Mbps	7680 b	Tree, loop ring, star	End of frame and echo check	Sonal multiplexing	4 wire shielded cable	500+	500 ft
MODBUS plus	1 Mbps	Variable	Linear	LRC or CRC	Token passing	Twisted pair	32	2500 m

2.8 Communication systems evaluation

A communication system can be evaluated in order to determine its physical and protocol properties. This section discusses methods that can be used to evaluate a communication system at both the physical layer and data link layer.

2.8.1 Physical layer evaluation

The properties of the physical layer can be analyzed with various methods. Jitter analysis and eye diagrams will be discussed as the main evaluation techniques for this layer.

2.8.1.1 Jitter

Jitter is a term used in industry for timing uncertainties in digital transmission systems. Controlling jitter is one of the most challenging design factors in high-speed digital systems, where timing is of utmost importance [29]. Furthermore, the variation in bit rate is called burstiness and is defined in terms of the difference between the main bit rate and the peak bit rate. The variation in delay is called jitter [7].

Jitter can be categorized in many different ways. The most important categorization is based on the phenomenological properties of the jitter itself. Jitter can be divided into two categories, namely random jitter and deterministic jitter. Random jitter is very unpredictable and is usually caused by physical noise processes. Deterministic jitter differs in the fact that it is predictable. Deterministic jitter can further be divided into periodic jitter, which is repetitive with a certain frequency and is caused by electromagnetic pickup from periodic signals, such as oscillators or switching power supplies, and data-dependent jitter. Data-dependent jitter is correlated to the actual data content. Finally, data-dependent jitter can be categorized as intersymbol interference (ISI) and duty cycle distortion. The instantaneous jitter value at each point depends on the current data bit and its bit history with ISI. Jitter caused by duty cycle distortion, depends only on the value of the current data bit [29].

2.8.1.2 Eye diagrams

An eye diagram is a time folded representation of an electrical or optical signal that carries digital information. With regards to eye diagrams, the term *bit-time* is used to represent the time needed to

transmit one bit of data. Data is overlapped with the interval of a bit time (time interval of one bit of information). By overlapping the bits, visual inspections and statistical methods can be used to determine rise times, fall times, jitter, amplitude, amplitude noise and overshoot present in the communication [29]. A simple eye diagram is shown in figure 2.9.

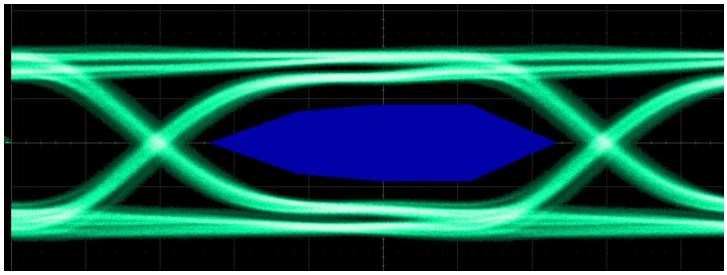


Figure 2.9 – Eye diagram example

Eye diagrams can also be used to determine aggregate performance of data patterns. The most common method to determine the performance of a signal is seen by the openness of the eye diagram. An open eye will indicate a quality signal, while a closed eye indicates signal impairments. By constructing an eye diagram of a signal it is also possible to extract the average power, crossing-point levels, signal-to-bit ratio, bit rate, duty cycle distortion etc. from the eye diagram [29].

2.8.2 Data link layer evaluation

Besides the physical characteristics of the communication network, it is necessary to analyze the protocol that is implemented on the network. This can be evaluated with the software simulation package Modelsim®, or by using the following characteristics of the physical layer [27] [30]:

- Throughput
- Data coding efficiency
- Medium access efficiency
- Responsiveness
- Cyclic times
- Bit error rate

The developed PROFIBUS protocol needs to meet the requirements of the standardized protocol. The developed protocol will therefore firstly be verified in the Modelsim® environment. When all the requirements have been met, the protocol will be implemented on the Xilinx® FPGA en validated. The following specifics regarding the PROFIBUS DP protocol will be verified and validated:

- Bit rate
- Character frame
- Telegram sequence
- State machine
- Parity and cyclic redundancy checks
- Timing and synchronization

Modelsim®

Modelsim® is a software simulation package that enables a user to simulate and verify hardware description languages. Modelsim® simulates real-time implementation of hardware description languages, by displaying the digital waveforms of the simulated program code. The different signals that are used in the hardware description language can be visually displayed, allowing the visual inspection of the applied protocol.

The real-time environment that is needed to simulate the signals of the hardware description language is defined in a test bench. A test bench is a program that simulates real-time environments by defining inputs to and from the developed programming code. This information is used by Modelsim® to simulate the environment in which the program code will be implemented. Figure 2.10 illustrates the waveform outputs of Modelsim® that is created using the test bench.

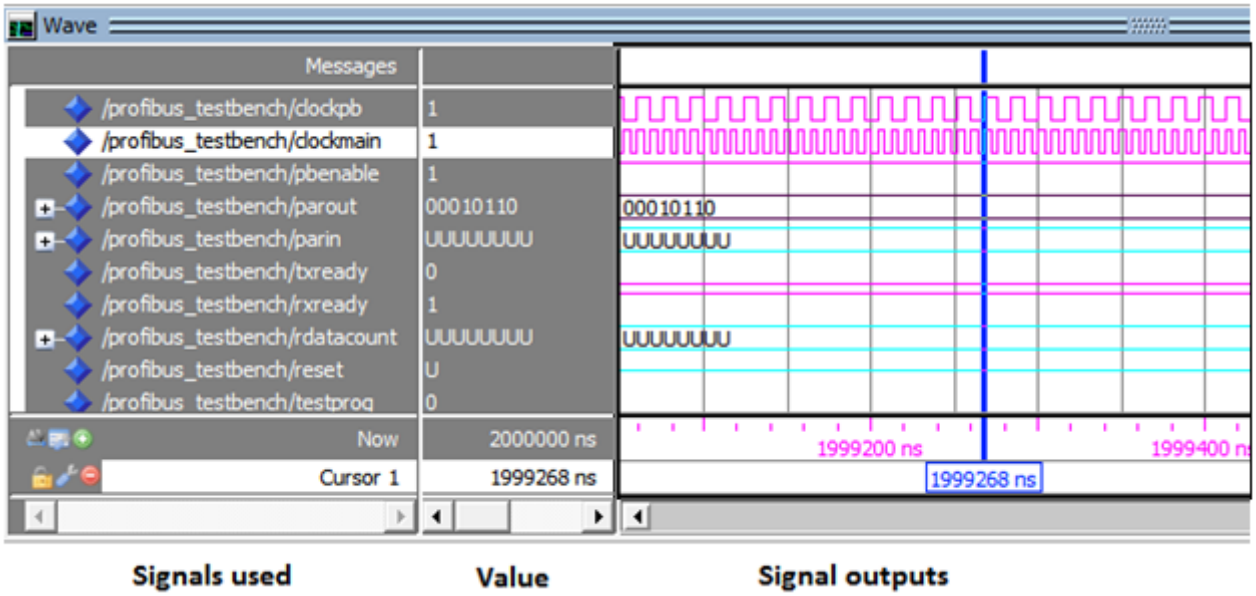


Figure 2.10 - Modelsim® environment

Throughput

Throughput can be defined as the user data rate and is related to the networks nominal bit rate multiplied by the communication efficiency. Throughput is calculated with (2.4) [27].

$$U^{(\max)} = \eta_{dc} . \eta_{ma} . R \tag{2.4}$$

In (2.4), *R* is the nominal bit rate, η_{dc} is the efficiency of the data coding scheme, with η_{ma} representing the medium access control (MAC) efficiency.

Data coding efficiency (η_{dc})

The quotient between the size of the user data that is transmitted in one transaction and the time needed to execute the service is called the data coding efficiency of a data link. This is represented in (2.5) [27].

$$\eta_{dc} = \frac{8d}{T_{ms}} \tag{2.5}$$

d represents the number of user data octets and T_{ms} is the message service time or message cycle time in bit times. T_{ms} is seen as the time needed for the completion of one read or write signal. This time is composed of several transmission services when the number of data octets, *d,* is higher than the maximum amount of data octets per frame that is allowed by the protocol [27].

Medium access efficiency (η_{ma})

Medium access efficiency can be defined as the time that is not wasted by the stations in the management of the bus. This will differ for each protocol, based on their method of MAC. In the case of PROFIBUS DP, where a token scheme is used, an amount of overhead exists and the medium access efficiency is represented by (2.6) [27].

$$\eta_{ma} = \frac{T_c - N_M T_{TC}}{T_c} \tag{2.6}$$

N_M is the number of master stations, with T_c equal to the mean cycle time of the network and T_{TC} is the token time [27].

Responsiveness

Responsiveness refers to the system response time T_R. Responsiveness is the maximum time allowed between a high priority request being made and the time when the first bit of that frame is being transmitted. This function is also dependent on whether a protocol uses a token passing scheme or not [27].

Cyclic times

Cyclic times differ for each protocol implemented on a communication system. Basically, the cycle time of a protocol is defined as the time it takes for one telegram cycle to complete. As an example consider the PROFIBUS-DP standard in (2.7) [16].

$$T_{MC} = \left(\left(T_{S/R} + T_{SDR} + T_{A/B}\right) * T_{TD}\right) + T_{ID} \tag{2.7}$$

Where:

T_{MC} - Time of one telegram cycle

$T_{S/R} = a \times 11bit,$ where $a = \text{No. of frame characters in Send/Request frame,}$

T_{SDR} - The reaction time of the slave to respond to a message

T_{TD} - Maximum time interval that occur on the transmission medium between a transmitter and receiver when a frame is transmitted

T_{ID} - Amount of time the initiator must wait before the next telegram can be sent

$T_{A/B} = b \times 11bit,$ where $b = \text{No. of frame characters in Acknowledge/Response frame}$

Bit error rate

Bit error rate (*P*) is the amount of erroneous bits in relation to the number of all the transmitted bits in a communication system. This simple equation is shown in equation 2.8 [30].

$$P = \frac{\text{number of erronous bits}}{\text{sum of all transmitted bits}} \tag{2.8}$$

The worst-case scenario would have bit error rate of 0.5, meaning every second bit is wrong. Practical implications as low as $P = 10^{-4}$ is feasible. A twisted-pair telephone cable has a bit error

probability of 10^{-5}. Error detection mechanisms are used to detect these errors, but sometimes not all errors are captured. The residual error rate (*R*) can then be defined in equation 2.9 [30].

$$R_e = \frac{\text{number of undetected erronous bits}}{\text{sum of all transmitted bits}} \tag{2.9}$$

The integrity of an error detection method can be calculated as the relation between *P* and *R.* Integrity class 1 is used for cyclic data exchange; class 2 for event driven communication; and class 3 for remote control systems [30].

2.9 Critical literature review

This section will aim to establish a relevant connection between the research presented in the literature study and the key objective of developing a VHDL-based Fieldbus system for the ADES. The complete literature study forms the essential information, obtained from research and development trends, that aims to establish the background and technological possibilities to develop this project. Each research topic will be discussed with its particular relevance to the study.

The literature chapter commence by discussing the current development in the field of digital communication with special regards to the automation field. This project will be developed by keeping in mind the current trends in the industry shift with regards to automation communication. The need for standardization in the industry is stressed together with the benefits of industrial standardization.

The next section of the literature focuses on the OSI reference model, established for digital communication in the ISO/IEC 7498. This standard describes communications between stations of a communication system, with the aim of providing a foundation for standards in system communication. The OSI reference model is used to design communication protocols and will construct the foundation for the VHDL-based Fieldbus design. The different layers of the OSI reference model are discussed in detail with special focus on the physical and data link layer. This is done with the aim of implementing the proposed PROFIBUS DP Fieldbus protocol, which mainly utilizes these two layers.

The literature regarding the physical layer of the OSI reference model is concerned with different physical implementations and standards, which are applicable to different Fieldbus standards. With regards to the particular Fieldbus implementation of this study, FPGAs and VHDL programming is

discussed to form a basic understanding for the proposed implementation. The data link layer is discussed in detail to establish an understanding of how protocols are designed with regards to medium access control, framing and security. This literature is important for the understanding of the PROFIBUS DP protocol which is discussed in chapter 3. The network, transport, session, presentation and application layer are briefly discussed for the completeness of the OSI reference model study and is not applicable to all Fieldbus profiles.

In the next section, the focus shift towards the history and structure of Fieldbus systems. The history of the Fieldbus systems and the IEC 61158 standard is profound for this study to establish the need for standardization in industrial communication. A comparison between the OSI reference model and the Fieldbus implementation of this reference model is presented to illustrate the relationship between Fieldbus systems and theoretical digital communication. The following section compares a few popular Fieldbus systems with each other. This comparison is done to distinguish between properties and application possibilities of the different Fieldbus profiles. PROFIBUS DP is selected from the comparison, to be implemented in this project, based on its reputation and application possibilities in the ADES.

After the implementation of the VHDL-based Fieldbus system on the ADES it will be crucial to measure the technical performance of the implementation as well as the correct implementation of the PROFIBUS DP profile. Thus the last section in this chapter is concerned with evaluation techniques which will be used to evaluate each aspect of the Fieldbus profile implementation. This will ensure that the PROFIBUS DP standard is fully met, as well as draw a comparison between commercial off-the-shelf PROFIBUS DP products and the FPGA implementation on the ADES.

Chapter 3 specifies the PROFIBUS DP profile and is also considered as critical literature for this study. The profile is discussed in detail to establish fundamental understating of the profile with regards to designing and implementing this profile on the ADES.

2.10 Conclusion

Chapter 2 discussed all the applicable theoretic principles and possibilities of a digital communication system. By using the information presented in this chapter it is possible to develop or implement a communication protocol on a system. Chapter 3 continues by doing an analysis of the PROFIBUS DP protocol and discusses the reason why PROFIBUS DP was chosen as the desired Fieldbus.

Chapter 3

PROFIBUS standard

PROFIBUS DP was chosen as the Fieldbus protocol that is used for data communication. Chapter 3 focuses on the detail aspects of the PROFIBUS DP protocol and why the standard was chosen as the desired Fieldbus. This chapter discusses the PROFIBUS DP protocol with regards to the physical and data link layer of the OSI reference model.

3.1 Introduction

PROFIBUS, or process Fieldbus, is an open, digital communication system with a wide range of applications, particularly in the fields of factory and process automation. PROFIBUS is suitable for both fast, time-critical applications and complex communication tasks [10].

PROFIBUS is described in the IEC 61158 as being a rugged open bus system that can guarantee problem-free communication. The PROFIBUS Fieldbus is mostly used in automation systems; human machine interfaces (HMI) or management systems to connect sensors and actuators. The Fieldbus system is fully standardized, that enables standard components from different manufacturers to be connected without problems [31].

3.2 Fieldbus selection

The criteria for selecting a Fieldbus is based mainly on the desired ADES Fieldbus application as well as general criteria used for Fieldbus systems. Examples of general criteria include openness, connectivity, interference protection, throughput, data efficiency, cycle times etc. A variety of Fieldbus systems are available, that are designed differently to meet specific application needs. In chapter 2, popular Fieldbus profiles were compared. There is however a large amount of Fieldbus systems that can be used apart from the Fieldbus systems discussed in chapter 2.

In order to choose a Fieldbus, the specific criteria need to be established. The criteria for this project are based on the requirements of the ADES, on which this Fieldbus will be implemented. The following Fieldbus requirements are defined for the ADES:

- Low cost
- High data rates
- At least 20 bytes of data per message transfer
- High interconnectivity
- Expandability
- Robust, low noise interference
- High data security
- RS 485 standard

Nearly every Fieldbus profile discussed in chapter 2 meets these requirements. The implementation cost of each Fieldbus profile does however differ greatly. PROFIBUS DP was chosen as the Fieldbus profile for the ADES based on the cost, characteristics and performance reputation. The following features were the main reason for choosing PROFIBUS DP:

- Low comparative cost
- Open standard
- Reputation of a rugged bus with low noise interference
- Up to 12 Mbps data transfer rates
- Up to 126 devices per segment (extra segments can be added)
- RS 485 physical layer
- Up to 244 bytes of data per transmission
- Hamming distance = 4

3.3 PROFIBUS background

PROFIBUS was developed in 1987 and has since been established as the world market leader in Fieldbus technology. The initial focus was set on communication technology and later moved to be comprehensive providers in factory and process automation by focusing on system integration, engineering and application profiles. The open standard used by PROFIBUS user organization and the continuous development of the PROFIBUS technology, ensure that users are equipped with a long-term solution [10].

PROFIBUS user organization was initially formed around the PROFIBUS Fieldbus message specification (FMS) and the faster PROFIBUS DP protocols in factory automation. In 1995, process automation was introduced and in 2002, PROFIBUS was the world market leader with a 20% market share and more than 5 million devices installed [10]. By the end of 2008, PROFIBUS was still the world's most successful Fieldbus with more than 28 million devices installed and is expected to grow to 50 million in 2012 [32].

PROFIBUS offers a single solution for plant wide and worldwide use by covering process, factory and a wide range of different applications. THE PROFIBUS protocol consists of three versions namely PROFIBUS process automation (PA), PROFIBUS DP and PROFIBUS FMS [10] [22].

3.3.1 PROFIBUS PA

PROFIBUS PA is designed with the emphasis on process automation, where high-speed and reliable communication is required. The protocol allows the link between sensors and actuators on one common bus line, even in potentially explosive areas. The transmission technique ensures intrinsic safety in accordance to the IEC 61158 standard and can easily be integrated in PROFIBUS DP networks [33] [22].

3.3.2 PROFIBUS DP

PROFIBUS DP is developed for high speed and inexpensive applications especially for communication between control systems and distributed input/output devices. Layer 1 (physical), layer 2 (data link) of the open systems interconnect (OSI) reference model is utilised [22] [33]. The protocol easily replaces the 24 V or 0 to 20 mA parallel signal transmission. The protocol uses the RS 485 or fibre optic standard with transmission rates between 9.6 kbps to 12 Mbps at 100 m transmission length [22].

3.3.3 PROFIBUS FMS

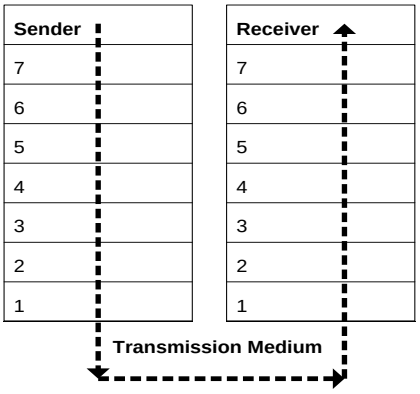
PROFIBUS FMS is a general-purpose data communication solution at the cell level. This protocol version has a wide range of application and produces great flexibility for extensive and complex communication tasks. Layers 1, 2 and 7 are utilized in this version where the application layer consists of FMS and lower level interface (LLI). The LLI provides FMS with access to layer 2, independent from the device in the system by representing the FMS services on the data transmission protocol of layer 2 [22] [33].

3.4 PROFIBUS DP technical overview

As mentioned, PROFIBUS is an open standard based on the IEC 611568. It was for a long time the fastest available Fieldbus with data rates up to 12 Mbps and input/output message sizes of up to 244 bytes. PROFIBUS has plug and play operation with the ability to connect 126 stations on a single bus with up to 32 stations per bus segment. To achieve bus access control, PROFIBUS uses a master-slave system [16] [26]. The original PROFIBUS version, DP-V0, offers cyclic data exchange between masters and slaves. Version DP-V1 was developed to offer acyclic exchange between masters and slaves, while version DP-V2 allow direct communication between slaves [16]. This section focuses on the various characteristics of PROFIBUS with special regard to PROFIBUS DP.

3.4.1 OSI layers

PROFIBUS DP is based on international standards and utilizes layer 1 and 2 of the OSI reference model. Each layer handles specific tasks that relate to the PROFIBUS and OSI standard. Layer 1 is defined in the OSI reference model as the physical layer and characterizes the physical transmission characteristics, while layer 2 explains the bus access control in the data link layer. Although the application layer (layer 7) is not directly associated with PROFIBUS DP, this layer is associated with the graphical user interface of the communication standard, used for the initial PROFIBUS PMC implementation. PROFIBUS PA uses the application layer, while PROFIBUS DP only includes the first two layers and a user interface [33] [16]. The various layers of the OSI reference model is shown in figure 3.1. The layers used by PROFIBUS DP are discussed in detail in the next section.



Layer	Function
Application layer	Interface to application program
Presentation layer	Representation of data for analysis and interpretation
Session layer	Synchronization of communication processes
Transport layer	Controls data transmission for layer 5
Network layer	Establishes and clears connections to avoid network congestion
Data-link layer	Handles the bus access protocol and data security
Physical layer	Definition of the hardware, speed, coding etc.

Figure 3.1 – OSI Reference Model [10]

3.4.2 Master and slave principle

PROFIBUS DP systems use a bus master to poll slave devices. The system in figure 3.2 shows a master-slave system that has one active master station and multiple passive slave stations. Figure 3.3 shows a system that has multiple masters and slaves. When more than one master is present on a bus, a token-passing scheme is used to control the bus access rights. Only the master that has the token has the right to address the slaves, send messages to the slaves and request messages from the slaves. In version DP-V0 the master exchanges data with the slaves cyclically. DP-V1 allows acyclic data exchange while DP-V2 allows direct communication between slaves [33] [16].

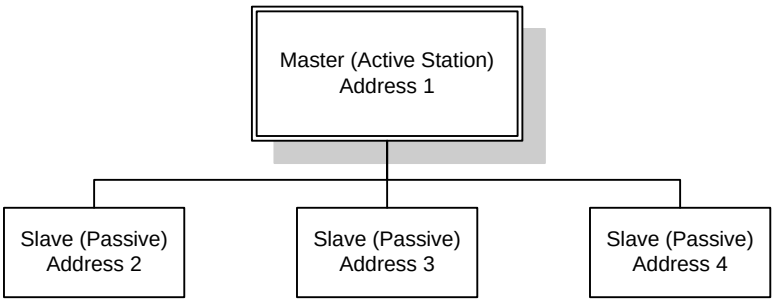


Figure 3.2 – Single Master with multiple slaves

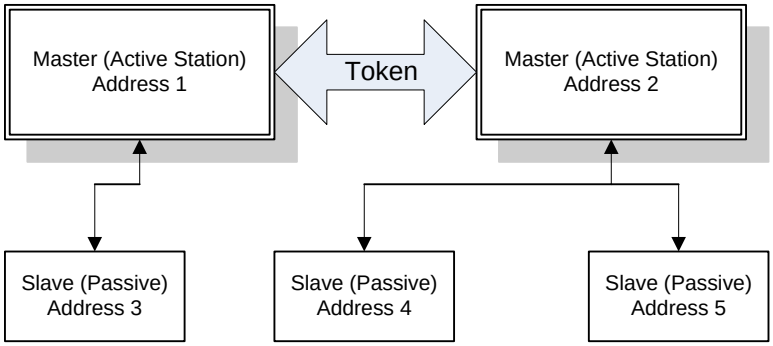


Figure 3.3 – Multiple Master with multiple slaves

Masters are active stations on the bus network which control all the communication on the network. There are two types of masters defined for PROFIBUS DP. Class 1 masters are responsible for communication and data exchange with slaves, while class 2 masters are specifically used for diagnostics and for commissioning slaves. A class 1 master is normally a central programmable controller. It sets the baud rate for the bus and slaves automatically detect this during start-up of the system. Master devices handle the data exchange with slaves and handle I/O information according

to a defined message cycle. When multiple masters are present, the masters are responsible for token transfers between masters. When only one master is used in a system, it has the token the whole time [10] [31] [33].

In PROFIBUS version DP-V0 and DP-V1, slaves form passive stations since they do not have bus access rights. A slave can only respond to requests from a master or acknowledge that a message was received. Each slave module developed has a GSD file associated with it. A GSD file can be seen as an electronic data sheet for the device and is needed by the master to determine the parameterization and configuration of the network [16].

3.4.3 GSD files

GSD files are mandatory ASCII files that specify Input/output data, parameterization and configuration data, address location list and bus parameters for all connected stations. A master uses this information to set up communication with slaves and is fundamental for the master parameter record on start-up. Usually a PROFIBUS configuration tool is used to read the GSD file and allows plug and play interoperability amongst different manufactured devices [10].

A GSD file is divided into three sections, namely general, master and slave specifications. In the general specifications, information regarding device vendors together with hardware and software release versions is specified. The master and slave sections contain information about the applicable parameters respectively. By reading the GSD file, the user can set up the PROFIBUS network for optimal performance according to the different devices on the network [10]. Appendix C.1 gives an example of the GSD file of the EM 277 slave module used in this study.

3.4.4 Physical layer

In the basic PROFIBUS version, where shielded, twisted pair cables are used, data transmission is implemented in accordance with the RS 485 standard. With this standard, data rates of 9.6 kbps to 12 Mbps can be achieved. The shielded, twisted pair cable used as the bus line, is terminated at both ends as seen in figure 3.4. PROFIBUS is based on a semi-duplex communication system where a binary high is represented by a positive voltage on line B with respect to line A. Line A and B are also commonly referred to as the RxD/TxD-N and RxD/TxD-P signals respectively. Non-return to zero (NRZ) data is transmitted on a PROFIBUS line, meaning that the transmitted signal does not return to zero between bit transitions. Figure 3.5 shows the form of line A and B with respect to the non-return to zero transmission [33].

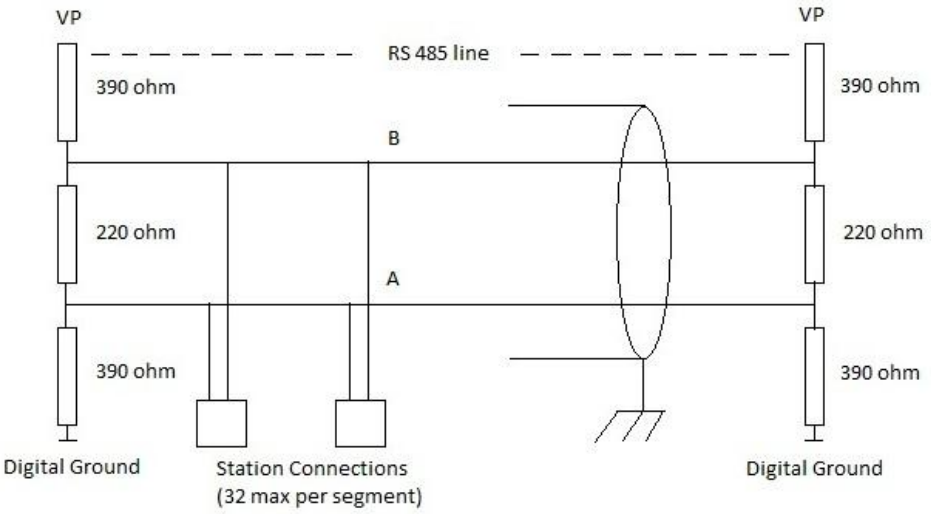


Figure 3.4 – RS 485 bus segment

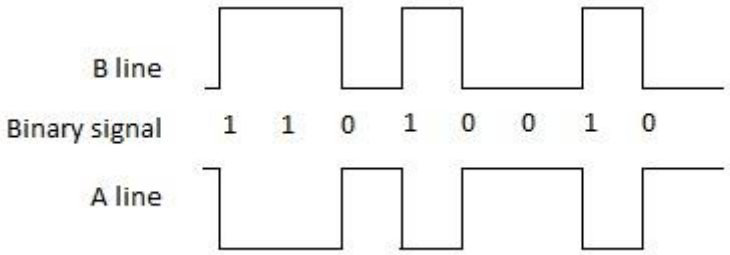


Figure 3.5 – Non-return to zero transmission

In order to ensure specific PROFIBUS transmission speeds, maximum allowable cable lengths are defined based on the baud rate. The defined cable used in PROFIBUS networks has to meet specific characteristics. When the length of a PROFIBUS network exceeds the allowable cable length, repeaters can be used to amplify the signals on the bus line. These allowable cable lengths and characteristics are defined in tables 3.1 and 3.2 [33].

Table 3.1 – Allowable PROFIBUS cable length based on the baud rate

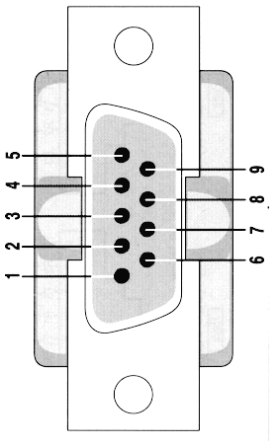
Baud rate (kbps)	9.6 to 187.5	500	1500	12000
Segment length (m)	1000	400	200	100

Table 3.2 – PROFIBUS RS485 type A cable specifications

Surge impedance	135 to 165 Ω, at a measuring frequency of 3 to 20 MHz
Cable capacitance	< 30 pF per meter
Core cross section	> 0.34 mm²
Cable type	Twisted pair, 1 x 2 or 2 x 2 or 1 x 4 conductors
Loop resistance	< 110 Ω per km
Signal attenuation	9 dB maximum over the entire length of the cable section
Shielding	Braided copper shield or braided shield and foil shield

According to the PROFIBUS standard specified in the IEC 61158, a nine pin connector is recommended to connect a bus station and cable. Table 3.3 shows the pin assignment of the connector with the mandatory signals shown in bold. In order to ensure a defined idle potential difference on the bus line when no data is transmitted, it is necessary to terminate the data lines A and B of the bus. Pull-down resistors as well as a pull-up resistor are connected to the digital ground and positive voltage input respectively. This is shown in figure 3.4. It should be noted that when transmission speeds higher than 1.5 Mbps are needed, specialized connection plugs with additional longitudinal inductance should be used since the connected stations will have a higher capacitive load [33].

Table 3.3 – Pin assignment of 9-pin connector [33]

Connector View	**Pin Number**	**Signal Name**	**Designation**
	**1**	**Shield**	**Shield, protective ground**
	2	M 24V	Minus 24 output voltage
	3	**RxD/TxD-P**	**Receive/Transmit data (line P)**
	4	CNTR-P	Control for line P
	5	**DGND**	**Data ground**
	6	**VP**	**Voltage plus**
	7	P 24V	Plus 24 output voltage
	8	**RxD/TxD-N**	**Receive/Transmit data (line N)**
	9	CNTR-N	Control for line N

The PROFIBUS standard also allows for fibre optic cable to be used based on the guidelines of the PROFIBUS Nutzerorganization. Fibre optic cables are not sensitive to electromagnetic interference and permit transmission distances up to 15 km between stations. These advantages have made this technology very popular for PROFIBUS and other Fieldbus devices. Another transmission technology is available for PROFIBUS PA but will not be discussed in detail in this document since it does not apply to the application of this study. This specification uses the Manchester code protocol on a shielded or unshielded twisted-pair cable. Resistor and capacitor passive line terminations are used at the ends of cable segments for termination [33]. This specification is described in the IEC 61158-2 standard.

3.4.5 Data link layer

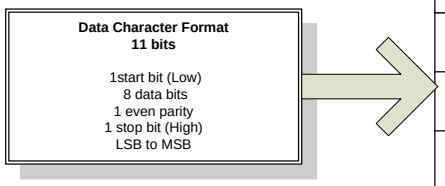
Layer 2 of the OSI reference model describes the security and transmission protocols regarding data transmission. In PROFIBUS, layer 2 is called the Fieldbus data link layer (FDL). The telegram specified in this layer, provides high transmission security. A hamming distance of 4 is used for security. This is achieved by applying the specifications of the IEC 61784 PROFIBUS standard, by selecting special start and end identifiers for the telegrams, by using gap-free synchronization, and by using a parity bit and a control byte. Hamming distance is discussed in detail in chapter 2. The different start and stop delimiters as well as the parity and control bits are used to ensure error free communication. Any error that can occur in the character format, protocol, start and end delimiter, frame check byte, telegram can be detected. If errors are found in the telegram, it is re-sent between one and eight times. PROFIBUS DP uses two transmission services namely *Send and Request Data with Acknowledge* (SRD) and *Send Data without Acknowledge* (SDN) [33]. SRD and SDN are discussed in the next section.

PROFIBUS meets two crucial bus access control requirements for automated industrial and manufacturing processes. These requirements are the basis of Fieldbus technology. This includes the necessity for equal programmable controllers to have sufficient opportunity to perform its necessary communication within a defined period, as well fast communication with little protocol overhead [33]. PROFIBUS allows for master-to-master systems, master-to-slave systems and a combination of the procedures [16] [33]. When multiple masters are used in one system a token passing scheme is needed to grant each master communication rights. The system on which the PROFIBUS protocol will be implemented has only one master and a token bus procedure is not necessary.

The PROFIBUS protocol determines the structure of the data, i.e. frame or packet size, and deals with aspects such as flow control, error detection and error recovery [7]. In the PROFIBUS standard, the data link layer is commonly referred to as the Fieldbus data link (FDL) layer [26].

3.4.5.1 PROFIBUS DP data character format

The data character format of the PROFIBUS standard is explained in figure 3.6. A PROFIBUS character consists of 11 bits. A single start bit (low), 8 data bits, 1 even parity bit and 1 stop bit (high) is included in a PROFIBUS character. Transferring data is done from least significant bit (LSB) to most significant bit (MSB) and the non-return to zero line coding method is used. In each cyclic message transfer a telegram is sent consisting of up to 256 bytes of information. Figure 3.6 gives a detail description of the PROFIBUS DP telegram [26].



PROFIBUS DP telegram format

SD	1 data character	**Start Delimiter** * Identifies beginning of telegram
LE	1 data character	**Data Length** * DU + DA + SA + FC + DSAP + SSAP
LEr	1 data character	**Length repeated** * For redundancy
DA	1 data character	**Destination Address** * Valid between 1-127 (00H..7FH)
SA	1 data character	**Source Address** * Valid between 0-127 (00H..7FH)
FC	1 data character	**Function Code** – Specifies - Type of telegram (request, response, acknowledgement) - Type of station (passive, active/slave, master) - Priority and telegram acknowledgement (successful or not)
DSAP	1 data character	**Destination Service Access points** *Tells what data is to be transmitted or which function is to be performed
SSAP	1 data character	**Source Service Access points** *Tells what data is to be transmitted or which function is to be performed NB – only telegrams with data fields use SSAP and DSAP
DU	1 to 32 data characters (or 1 – 244)	**Data Units** * Usually 32 bytes but up to 244 bytes.
FCS	1 data character	**Frame Check Sequence (00F...FFH)** * Sum of ASCII bytes of from DA to DU mod 256 ((DA+SA+FC+DU) mod 256)
ED	1 data character	**End Delimiter** **End** of PROFIBUS telegram - 16H

Figure 3.6 – PROFIBUS data character format

3.4.5.2 PROFIBUS operating states

In order to exchange data on a PROFIBUS network, the masters and slaves in the network have to be configured according to the PROFIBUS specification and the current network. There are four major operating states in a PROFIBUS specification namely; *Power On/Reset*, *Parameterization,*

I/O configuration and *Data Exchange* [16]. During these operating states, the different PROFIBUS DP devices have different communication responsibilities and capabilities. These are listed in table 3.4 [16] [33].

Table 3.4 – Communication responsibilities of PROFIBUS DP devices [16] [33] [26]

Function	**DP Slave Class 1**		**DP Master**		**SAP* number**	**Layer 2 Service**
	Requester	**Responder**	**Requester**	**Responder**		
Data_Exchange		Mandatory	Mandatory		Default SAP = 0	SRD*
RD_Input		Mandatory			56	SRD
RD_Output		Mandatory			57	SRD
Slave_Diagnostics		Mandatory	Mandatory		60	SRD
Set_Parameters		Mandatory	Mandatory		61	SRD
Check_Configuration		Mandatory	Mandatory		62	SRD
Get_Configuration		Mandatory			59	SRD
Global_Control		Mandatory	Mandatory		58	SRD
Set_Slave_Address		Optional			55	SRD
M-M-Communication			Optional	Optional	54	SRD
DP V1 Services		Optional			51/50	SRD

*SRD – Send and request data with acknowledgement

*SDN – Send data without acknowledge

SAP – Service access point (indication of function that need to be executed)

When a master station in a PROFIBUS network controls the network, the following telegram sequences have to be followed to ensure correct parameterization and configuration before data transmission can occur [16].

- Request diagnostics
- Change station address – Class 2 master only
- Parameterize the slaves
- Configure the slaves

- Request diagnostics again before data exchange to ensure start-up was executed successfully
- Data exchange
- Global control – optional

After the initial power up of the system, the PROFIBUS master evaluates the slave parameters and configures the device for data exchange. The master requests the slave's diagnostic data after the slave has been detected on the bus. Once the slave is ready for parameter characterization, the configuration data and parameter-set is transferred to the slave by the master. The master can only start the cyclic data exchange once the diagnostics data of the slave is correct. Figure 3.7 gives an overview of the start-up sequence of the PROFIBUS protocol [33].

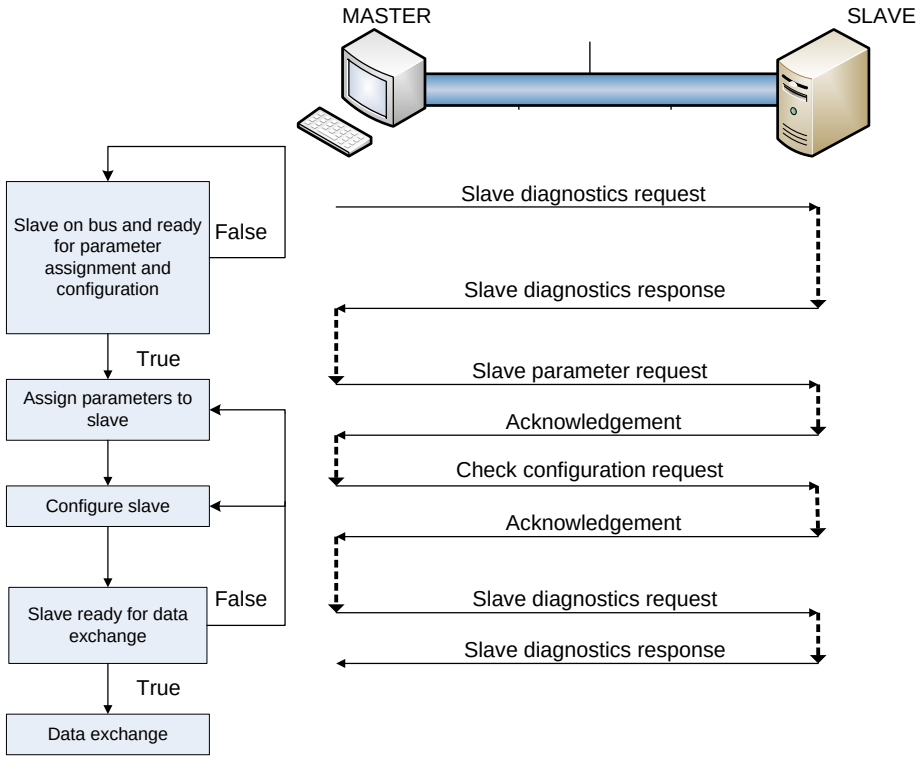


Figure 3.7 – PROFIBUS start-up sequence

The operations that accompany the PROFIBUS sequence shown in figure 3.7 is summarized in tables 3.5 to 3.8. The protocol specific functions that have to be executed in the various states are shown. Each function has a unique service access point (SAP) number that is necessary for the slave to recognize the command or request from the master.

Table 3.5 - Power ON/Reset

Master operation	**Slave operation**
• Slave diagnostics – SAP 60 • Set slave address – SAP 55 • (class 2 master station)	• Initiate itself • Detects BAUD rate • Respond with slave diagnosis response

Table 3.6 - Parameterization

Master operation	**Slave operation**
Set parameter – SAP 61	Reply with source service access point function

Table 3.7 - I/O Configuration

Master operation	**Slave operation**
Check configuration – SAP 62	• Acknowledge configuration telegram • Respond with response telegram • Check configuration against stored configuration. • If conflict occurs, check with GSD file and report error to master

Table 3.8 - Data Exchange

Master operation	**Slave operation**
• Read input – SAP 56 • Read output – SAP 57 • Data exchange – No SAP • Global control – SAP 58	• Automatically checks transferred output data, respond with input data and generate a message if difference or failure is detected • Reconfiguration of I/O data is allowed but not parameterization

3.4.5.3 Bus timing parameters

PROFIBUS was designed to guarantee and precisely predict the behaviour of the system over time [16]. There is a variety of timing parameters that have to be coordinated with each other to ensure a

faultless PROFIBUS network. These parameters must be the same for all devices on a network and depend on the selected data transfer rates. These bus parameters are specified with the unit *Tbit* (time bit). A *Tbit* depends on the data transfer rate and is the bus rotation time for one bit and is calculated as shown in (3.1) [16] [33]:

$$Tbit = \frac{1}{\text{data transfer rate}} \text{ bps} \tag{3.1}$$

Table 3.9 explains the bus parameters that are crucial to the correct functioning of the PROFIBUS network – but not all are mandatory [16] [33].

Table 3.9 A – PROFIBUS bus parameters [16] [33] [26]

Timer	**Symbol**	**Description**
Sync. time	T_{SYN}	Minimum time that a station must remain in idle state before another request can be accepted
Sync. interval time	T_{SYNI}	Serves the supervision of the maximum allowed time interval between two consecutive synchronization times or receiver synchronizations
Station delay time	T_{SDx}	The elapsed time between the last bit of a transmitted or received frame and the first bit of the next frame
Station delay of initiator	T_{SDI}	Delay of the station transmitting an request or a token frame
Min. station delay of responder	$\min T_{SDR}$	The minimum time a slave takes to respond to a message
Max. station delay of responder	$\max T_{SDR}$	The maximum time a slave takes to respond to a message
Quiet time	T_{QUI}	The time interval that will occur when the non-return to zero signals are transposed into a different signal coding. This will be the transmitter fall time occurring after the transmitter is switched off
Ready time	T_{RDY}	The time a master needs between transmitting a request and being ready to receive a response or acknowledgement
Safety margin	T_{SM}	Safety margin time
Set-up time	T_{SET}	Set-up time interval between an event and until the required reaction is performed

Table 3.9 B – PROFIBUS bus parameters continued [16] [33] [26]

Idle time	T_{ID}	The time that elapses at the transmitter after a frame's last bit is received on the transmission medium until it sends the next telegram
Transmission delay time	T_{TD}	Maximum time interval that occur on the transmission medium between a transmitter and receiver when a frame is transmitted
Slot time	T_{SL}	The time the master waits for receipt of the first frame character (11bits) of the response after the last bit of an action frame is sent out
Time-out	T_{TO}	The time-out monitors the activity on the bus and the Idle time. If the idle time reaches the limit on time-out the bus will be considered inactive
Message cycle time	T_{MC}	The cycle time consists of the frame transmission times, the transmission and station delay times
System reaction time	T_{SR}	The number of possible message cycles per second

Based on table 3.9, table 3.10 addresses the calculation methods and requirements for the timing characteristics of the PROFIBUS DP protocol.

Table 3.10 A – PROFIBUS bus parameter functions [16] [33] [26]

Timer	**Function**
Sync. time	$33\ Tbit$
Sync. interval time	$2 \times \left(2\ \times (33\ bit + 255\ \times 11\ bit)\right) +\ 33\ bit = 11385\ bit$
Station delay of initiator	$T_{SDI} =\ t_{SDI}\ \div\ t_{BIT}$
Min. station delay of responder	$\min T_{SDR} = \min t_{SDR} \div\ t_{BIT}$
Max. station delay of responder	$\max T_{SDR} = \max t_{SDR} \div\ t_{BIT}$
Quiet time	$T_{QUI} < \min T_{SDR}$ $T_{QUI} < T_{RDY}$
Ready time	$T_{RDY} < \min T_{SDR}$

Table 3.10 A – PROFIBUS bus parameter functions continued [16] [33] [26]

Safety margin	$T_{SM} = 2bit + 2 \times T_{SET} + T_{QUI}$
Set-up time	$T_{SET} = t_{SET} \div t_{BIT}$
Idle time	$T_{ID1} = \max(T_{SYN} + T_{SM}, \min T_{SDR}, T_{SDI})$
Transmission delay time	$T_{TD} = t_{TD} \div t_{BIT}$
Slot time	$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + 11bit + T_{SM}$
Time-out	$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL}$, Masters: n = station address, Slaves: n = 130, independent of its station address
Message cycle time	$T_{MC} = T_{S/R} + T_{SDR} + T_{A/B} + T_{ID} + 2 \times T_{TD}$, $T_{S/R} = a \times 11bit$, $T_{A/B} = b \times 11bit$, $a = No.of\ frame\ characters\ in\ Send/Request\ frame$, $b = No.of\ frame\ characters\ in\ Acknowledge/Response\ frame$
System reaction time	$R_{SYS} = \frac{1}{t_{MC}}$, $message\ rate$ $t_{MC} = T_{MC} \times t_{BIT}$, $T_{SR} = np \times T_{MC} + mp \times RET\ T_{MC}$, $np = number\ of\ slave\ stations$, $mp = number\ of\ message\ retry\ cycles\ per\ Poll\ Cycle$, $RET\ T_{MC} = message\ retry\ cycle\ time$

3.5 Conclusion

PROFIBUS DP is considered to be a robust and one of the most popular Fieldbus standards for industrial automation. Chapter 3 discussed the protocol and characteristics of this standard. Based on chapter 3, this protocol can be implemented on any platform that meets the physical layer characteristics of PROFIBUS DP. This implementation is presented in chapter 5. However, it is necessary to understand the specific platform on which this protocol will be implemented. Chapter 4 continues by presenting the hardware platform for the protocol implementation.

Chapter 4

Application platform

Chapter 2 and 3 have presented all the characteristics of a digital communication system with specific focus on the PROFIBUS DP protocol that was chosen as the desired Fieldbus standard for the ADES. Chapter 4 presents an overview of the hardware platform on which the Fieldbus protocol will be implemented. This chapter together with chapter 2 and 3 summarizes all the information needed to implement a Fieldbus protocol on a certain platform.

4.1 Introduction

In order to understand the requirements of a Fieldbus communication system, it is necessary to understand the application platform on which the Fieldbus will be implemented. A rotor delevitation system (RDS) that is controlled by an active magnetic bearing drive-electronic system (ADES) was developed at the North-West University in 2009 for the purpose of high-speed turbo-machine applications as well as an AMB research platform.

The PROFIBUS DP standard is implemented on this system in order to produce a system that has different operating options as will be discussed further. Hardware and software changes were made on the system to implement the PROFIBUS communication standard.

4.2 Rotor delevitation system (RDS)

The RDS was specifically designed as a platform for high-speed turbo-machinery in next-generation nuclear reactors as well as to be a platform to specifically evaluate backup bearings and digital control of AMBs. The system uses two radial AMBs and one axial AMB instead of conventional bearings to suspend a rotor. The absence of conventional bearings allows for low mechanical resistance which permits high-speed operation. The RDS was designed for an operational speed of 19,500 r/min when rotated by a permanent magnet synchronous machine. The AMBs are actuated by power amplifiers and the position of the axis measured by inductive sensors. The RDS conforms to all the necessary requirements to function in a next-generation nuclear reactor. The control of the

AMBs proposed a big challenge and the need arose to develop a robust controller that met industrial requirements. The ADES was specifically designed for this purpose to interpret the sensor measurements and control the power amplifiers of the RDS.

4.3 AMB drive electronic system (ADES)

The initial ADES system constituted of different units as seen in figure 4.1. An FPGA PMC module (U6) was programmed to be the central controller of the ADES. The FPGA receives sensor signals from the inductive sensor unit (iSensor) (U10) situated next to the AMBs (U9). The controller interprets these signals and adjusts the power amplifiers (PAs) (U8) in order to keep the rotor of the RDS levitated. A PROFIBUS PMC card installed on the single-board computer (SBC) is configured to communicate between the SBC and the programmable logic device (PLC) (U4).

The system was designed to have two possible operating stations. A maintenance station by means of a graphical user interface (GUI) is directly connected to a single-board computer (SBC). The SBC forms the backplane for the PCI modules used in the controller. This station has an interface that graphically displays the rotor's position with regards to the horizontal and vertical plane, as well as the power amplifier currents. The GUI can change certain control parameters needed for PID control and change between the different operating states of the system. A second operating station was added to have a PLC interface to externally control the system. The PLC station has a user interface by means of buttons that can levitate and delevitate the rotor. The PLC can be expanded to have a graphical user interface that has the same operating capabilities as the GUI of the SBC.

The system was developed for industry and therefore the PLC station was chosen as the method to operate the RDS. Since the controller is usually situated at a different location as the RDS, a communication standard was needed to communicate between the PLC and SBC. This was initially implemented by means of PROFIBUS PMC module attached to the SBC, and connected to a Siemens® PROFIBUS slave. PROFIBUS uses RS 485 as the data transfer standard, which was also used to establish the communication link [34].

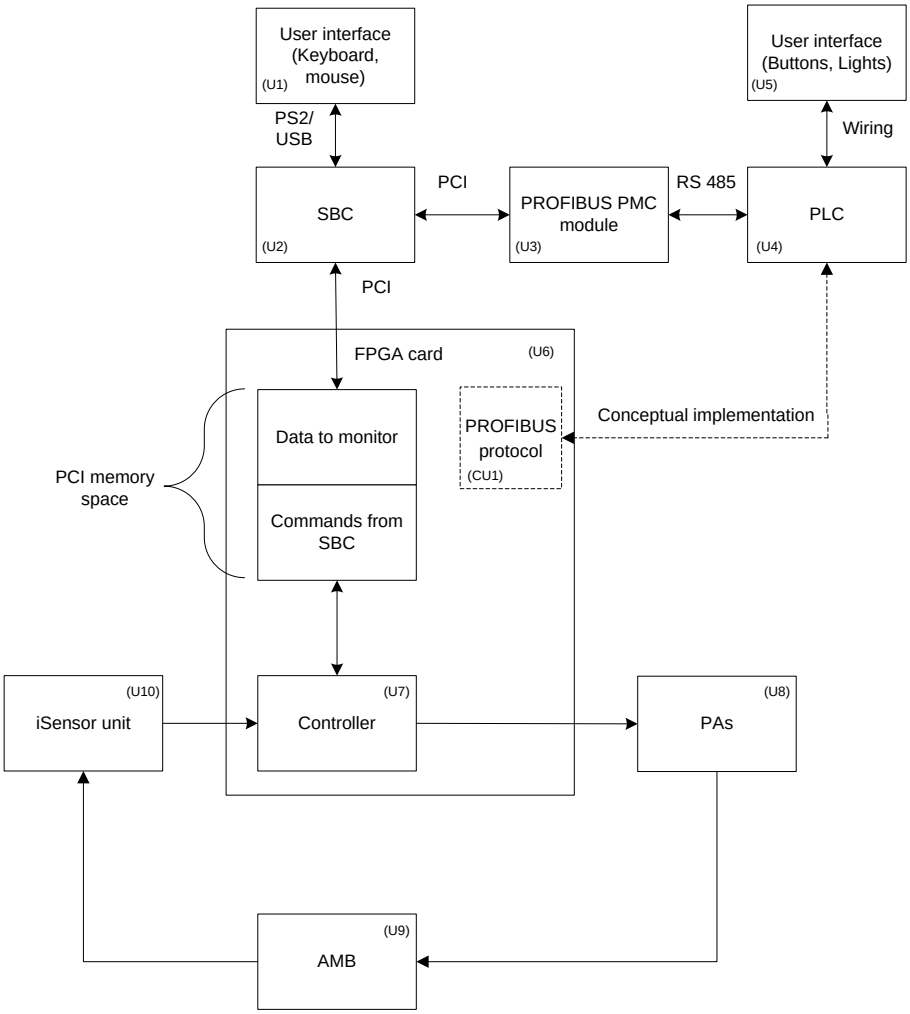


Figure 4.1 – ADES system overview

4.3.1 ADES hardware

Most of the hardware purchased for the ADES was commercial off-the-shelf (COTS) items that were chosen to ensure a robust system that has lower cost and development time. The different hardware items will be discussed in this section.

4.3.1.1 Single-board computer

The SBC, shown in figure 4.2, forms the backbone on which the entire ADES relies. The specific PP 412/03x module from Concurrent Technologies® supports the Intel® Core™ 2 Duo processor and the Intel® E7520 server class chipset. This dual PMC compact PCI board is PC-compatible, making this the perfect solution for the ADES requirement [35].The single-board computer has the following features:

- 2.16 GHz Intel® Core 2 Duo processor T7400
  - Dual-core processor
  - 667 MHz front side bus
  - 64 kbytes L1 cache per core
  - 4 Mbytes L2 cache shared between cores
  - Intel® 64-bit computing support
- Up to 4 Gbytes of dual channel DDR2-400 ECC SDRAM
- 2 x PMC module interfaces, with front and rear I/O
- Compact PCI controller
- Support for Linux® and Windows®

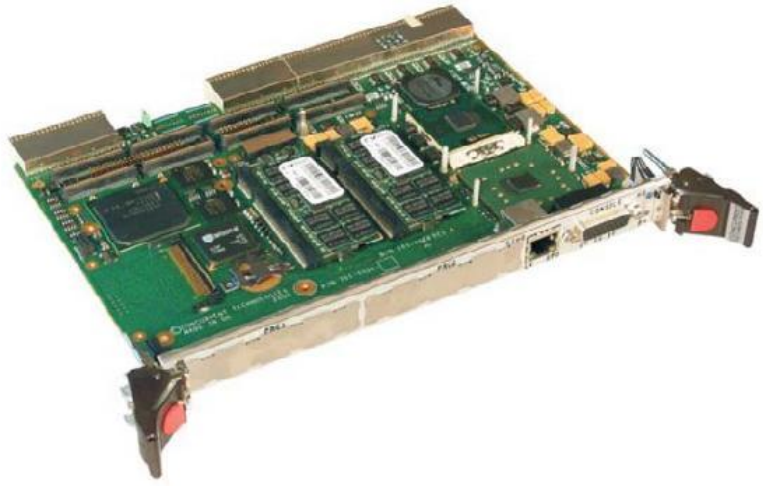


Figure 4.2 – Single-board computer

4.3.1.2 Xilinx® FPGA

The PMC-VFX70© device is a user-configurable Virtex©-5 FPGA module with plug-in I/O. This reconfigurable FPGA is enhanced with multiple high-speed buffers and a high-throughput PCI interface. The PMC module allows it to be compatible with the SBC. The on-board FPGA has a hard-core PowerPC block to handle the most complex and memory-intensive computing applications. This device is specifically programmed to produce dedicated control throughout the ADES [36]. The Xilinx® Virtex©-5 FPGA has the following characteristics:

- Reconfigurable Xilinx® Virtex©-5 FPGA
- 71680 logic cells
- PCI bus, 100MHz, 64-bit interface

- Front and rear I/O connections
- 64 I/O lines
- Dual-port SRAM and DDR2 SDRAM memory

4.3.1.3 Power Amplifiers

The power amplifiers are designed to amplify the currents of the AMBs based on a reference value that is received from the main controller. The power amplifier boards were manufactured in-house to meet the specific needs of the RDS. The Virtex©-5 FPGA is programmed to communicate the necessary control signals to the power amplifiers.

4.3.1.4 Inductive sensor

In conventional AMB systems, eddy-current sensors are used to measure the position of the rotor inside the AMBs. For the ADES an inductive sensor was developed in-house for this purpose. This reduces the cost and adds local expertise to the project team [37].

4.3.1.5 Siemens® S7-200 PLC

The Siemens® S7-200 PLC and EM 277 PROFIBUS DP module are used as an industrial standard controller for the complete active magnetic bearing system. The S7-200 is the basic PLC device, while the EM 277 module is an additional unit.

S7-200 PLC

The S7-200 is a micro-programmable logic controller that monitors inputs and changes outputs according to the control program specified by the user. The S7-200 can control a large amount of devices and is suited for automation processes. The STEP 7™-Micro/WIN Programming Package from Siemens® is used to program the device according to the user requirements [38]. This specific CPU 224 version of the S7-200 PLC has the following features:

- 12288 bytes program memory
- 8192 bytes data memory
- 14 digital inputs

- 10 digital outputs
- Built-in real-time clock
- RS 485 communications port

EM 277 PROFIBUS DP module

The EM 277, shown in figure 4.3, is an expansion module for the S7-200 PLC from Siemens®. This device is certified by the PROFIBUS Nutzerorginization by meeting the specifications of the IEC 61158 and IEC 61784. This device acts as a slave on any PROFIBUS network. A master device can communicate with the EM 277 over the RS 485 port of the device. By configuring the PROFIBUS network, the master detects the configuration settings and address of the EM 277 slave device. The EM 277 makes use of the memory space of the S7-200 PLC for data transfer on the PROFIBUS network [38]. The following features of the EM 277 module allows for PROFIBUS application:

- RS 485 physical layer
- Baud rates from 9.6 kbaud to 12 Mbaud supported
- PROFIBUS DP slave protocol
- 6 multilateral interoperability programme (MIP) connections

Figure 4.3 – EM 277 PROFIBUS-DP module

4.3.1.6 PROFIBUS PMC 253

The PMC 253 in figure 4.4 is a dedicated PROFIBUS DP V1 controller for Fieldbus applications manufactured by Hilscher®. This device was selected for its form factor which is the same single-

board computer used. The device complies with the IEC 61158 standard and is built for rugged industrial environments. This device can be configured to be a master or slave device that allows for transfer rates of 9.6 kbaud to 12 Mbaud [39]. The following features make the PROFIBUS PMC 253 module suitable for the ADES:

- 32-bit PMC module
- PROFIBUS DP V1 master or slave device
- Easy to use software user interface to set up PROFIBUS network
- Opto-isolation between the controller and the Fieldbus interface

Figure 4.4 – PROFIBUS PMC 253 device

4.4 Graphical user interface (GUI)

A GUI was needed to display certain parameters of the RDS and to operate the system directly from the SBC. It had to be designed to utilize the software drivers of the SBC. The GUI had to meet the following requirements:

- A two dimensional, real-time representation of the radial position of the rotor
- A one dimensional, real-time representation of the axial rotor position of the rotor
- A one dimensional, real-time representation of the ten power amplifier currents
- Various buttons to operate the system in different operating states
- An input method to change the different PID-control parameters of the system

4.4.1 Implementation

The SBC communicates with the various PMC modules and devices over the compact PCI backplane. The software drivers used to gain access to the various memory spaces on the SBC were available in Microsoft® Visual C++ code. This language was chosen as the universal language for the GUI programming in order to use the supplied software drivers for the SBC.

An open-source graphical development package named QT® was used as the GUI building block. QT® is a library package that can be included into Visual C++. By combining the visualization capabilities of QT® and the drivers of the SBC, a program was developed that can communicate with the FPGA and give visual presentation of the RDS.

4.4.2 Software Design

The various requirements of the GUI were met by dividing the GUI into 5 separate procedures. These procedures and their interdependence are illustrated in figure 4.5. Each procedure was developed individually and afterwards combined to create the required GUI. The following discusses each procedure that forms part of this implementation.

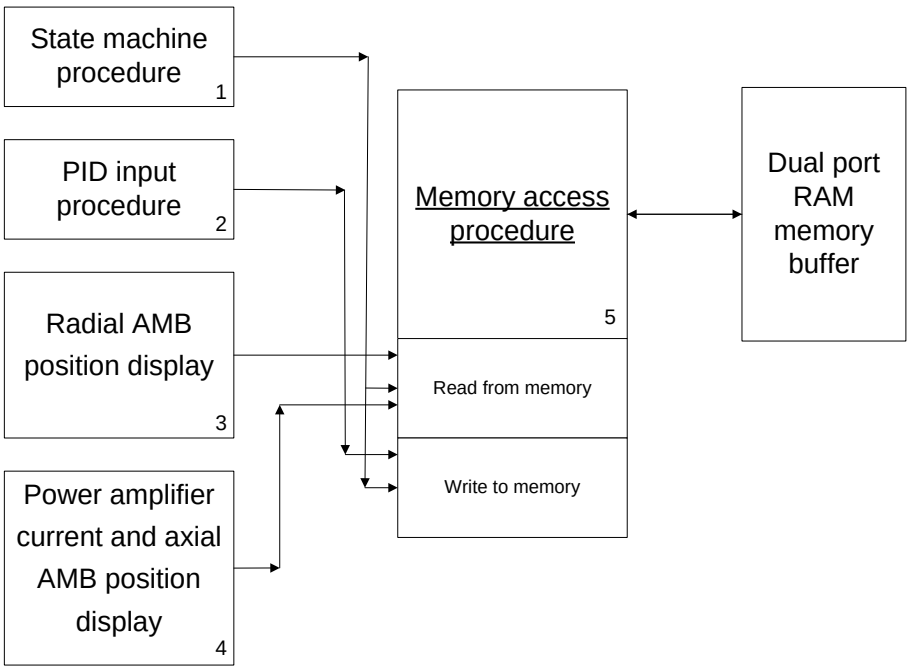


Figure 4.5 – GUI program layout

4.4.2.1 State machine

The RDS operates in different states. Each state has a specific purpose regarding the initialization of the system and different situations that the RDS need to handle. The developers of the RDS specified the following states. Figure 4.6 gives a visual presentation of the state machine of the ADES.

- No Power – RDS is completely shut down. This state is only a virtual state used to simplify the design
- Power Down – Initial configuration
- Standby – RDS is ready to be operated but not levitated
- AMB On – The AMB coils are excited by the power amplifiers to levitate the rotor
- Normal Operation – The motor-drive is operational and the rotor is driven at a nominal speed of 19,500 r/min
- Non Critical Stop – If a small error occurs
- Critical Stop – If a critical error occurs that completely shuts down the RDS

For the purpose of graphical representation of the state machine, each state was implemented by means of a GUI button. When a state is chosen – by clicking with a mouse on the button – the accompanying button changes the operational state and changes colour from orange to green. In order to truly relate the GUI to the state machine, the user is not allowed to change to an illegal state. This is visually represented by displaying the allowed states the user may choose in orange colour. The illegal states are displayed in red.

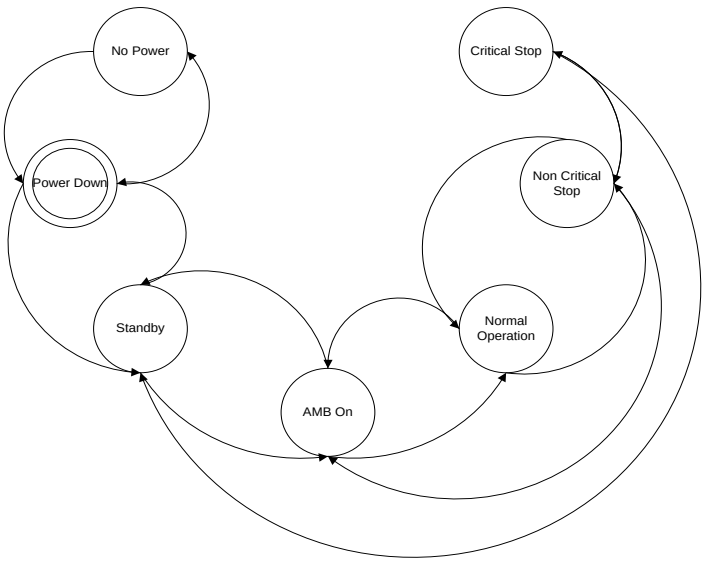


Figure 4.6 – ADES state-machine

4.4.2.2 PID input procedure

The AMBs are operated by controlling the reference currents to the power amplifiers. PID is used in this system as the control method and has specific parameters that can be adjusted for different control responses. The user therefore has to be able to adjust the PID parameters of the control. For this reason, this functionality was added to the GUI by means of a textbox input. The PID parameters for the radial and axial AMBs can be changed by entering the values into the specific textboxes and updating the system. The parameters can only be changed when the AMBs are not levitated and thus this feature of the GUI is only available when the system is in standby.

4.4.2.3 Radial AMB position display

The radial AMBs have two degrees of freedom in the X and Y-axis. By displaying the X and Y positions of the AMBS, the operator has a graphical representation of the rotor with regards to the AMBs. To present the AMB's position correctly, it was necessary to have a two-dimensional display of the AMB position. A two-dimensional graph was developed that continuously updates the X and Y positions in real-time by displaying the value as a small dot on the graph. This gave the user a graphical representation of the position of the rotor between the AMBs. Each radial AMB have its own two-dimensional graph.

4.4.2.4 Power amplifier current and axial AMB position display

The same method that was used for the radial AMB representation was used for the power amplifier currents and the axial AMB position. By visually displaying these values, the operator can evaluate the performance of the system and make control adjustments if necessary. These values are only one-dimensional and therefore was only presented as bar graphs that continuously updates the values in real-time.

4.4.2.5 Memory access

The AMB currents as well as the position measurements of the RDS are sent to the FPGA for control purposes. Figure 4.7 shows how the internal memory relates to the different sections of the SBC. In order to access the memory spaces, the C++ drivers of the SBC were used. The drivers were used in a procedure that reads the dual port memory space of the system. A read and write procedure was programmed to access memory data and send values to the SBC from the GUI.

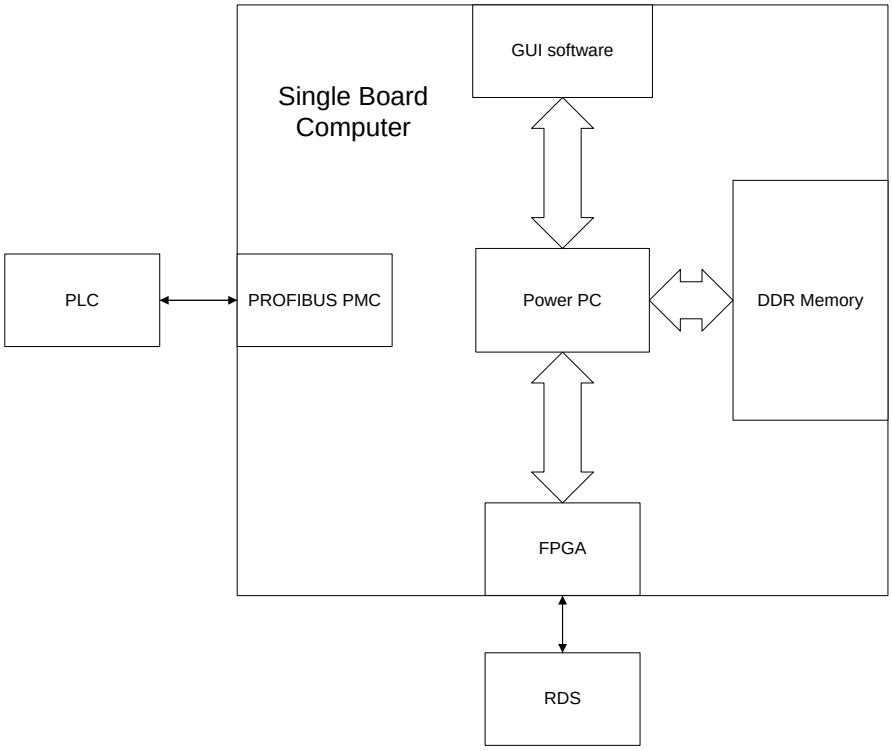


Figure 4.7 – Single-board computer internal memory path

Each of the GUI procedures either uses the read or write procedure for operation. Specific memory locations were designated for each value to be read or written in the DDR block of the SBC. Table 4.1 gives an overview of the memory spaces used by the GUI.

Table 4.1 – Location of DDR control registers in the SDRAM

Offset	**Register**	**Use**
805C – 805F	DDR-SDRAM Control/Status Register	Control read and write operations
8060 – 8063	DDR-SDRAM Address Register	Specify read/write address
8064 – 8067	DDR-SDRAM Read Register	Location of read memory value
8068 – 806B	DDR-SDRAM Write Register	Location of value written to memory
806C – 806F	DDR-SDRAM Mask Register	Determines size of read/write value

The read and write functions as well the specific section of the DDR memory map that was allocated for the RDS parameters are listed in appendix F. By combining these read and write operations with the different GUI procedures, the GUI operates in real-time by changing states, sending PID parameters and displaying specific values of the AMB system.

4.4.3 Verification and validation

Microsoft® Visual C++ was used as the development environment for the GUI. Each procedure was programmed and tested in the environment. The AMB environment was not used in the development phase of the GUI. A test program was developed that accurately simulates the data flow and memory space of the ADES. The GUI was thoroughly tested with this program before implementing it on the actual ADES and RDS.

After the GUI was developed it was implemented on the SBC of the ADES. The SBC uses a Microsoft® Windows® XP™ operating system. The GUI is run from the desktop of the display terminal connected to the SBC. Figures 4.8 and figure 4.9 illustrates the actual GUI as implemented on the SBC of the ADES.

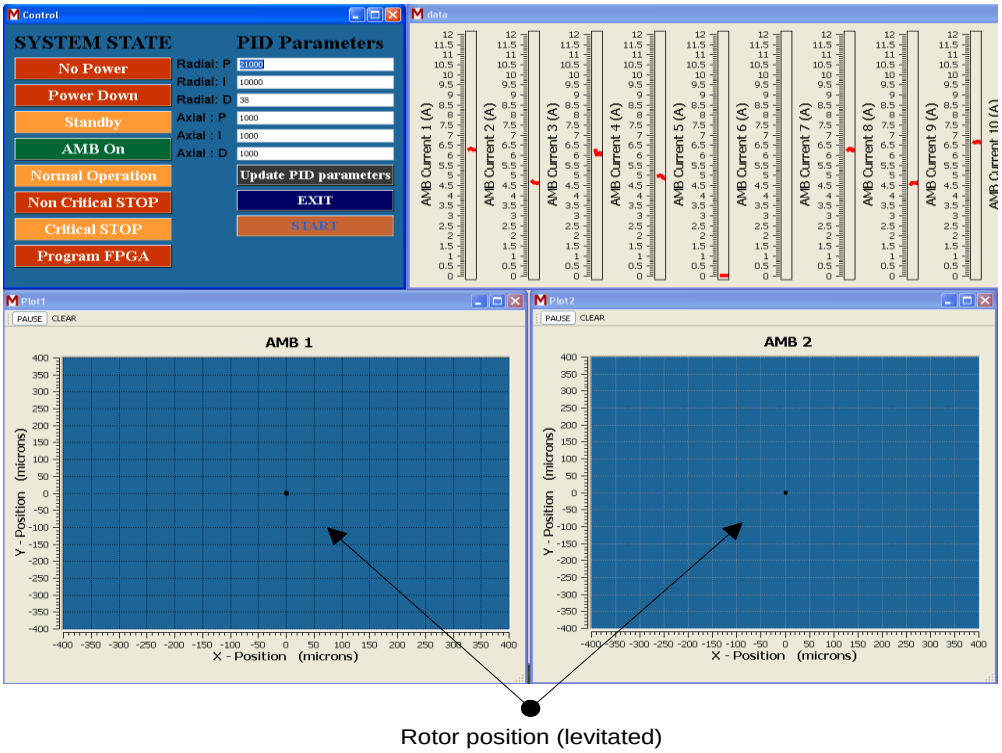


Figure 4.8 – ADES graphical user interface (system levitated)

The main control GUI section in the top, left hand corner of figure 4.8 and 4.9 shows the different state buttons and the PID parameter textbox inputs. Another feature is included in the GUI that allows the user to reprogram the FPGA before operation. Start and exit buttons are also added to initiate the program and safely close all the communication handles before the program shuts down. The radial and axial positions displays of the AMBs, as well as the power amplifier currents are seen

in the remaining section of figure 4.8 and 4.9. The operator of the system can monitor the entire system and make control adjustments if necessary. Figure 4.8 is captured while the rotor of the RDS was levitated. The positions marked on the figure represent the position of the rotor. Figure 4.8 indicates that the rotor is levitated in the middle of the RDS. The corresponding “AMB On” is activated and the AMB currents can be seen on this figure. All of these displays are captured in real-time. Figure 4.9 illustrates the ADES when the rotor is delevitated. The “Standby” state is activated and the AMB currents are zero. This as well as the position of the rotor displayed in the figure indicates the system is delevitated.

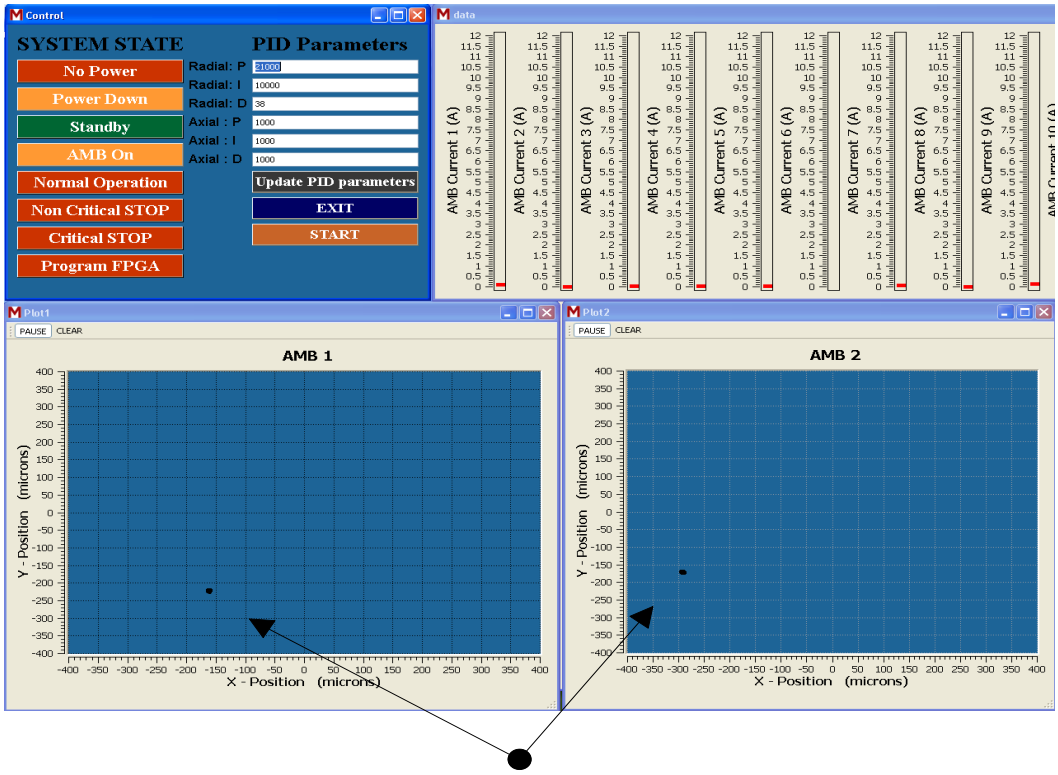


Figure 4.9 – ADES graphical user interface (system delevitated)

Various tests were performed to operate the RDS in all the different operating states. The buttons, textboxes, and graph presentations all worked according to the design and met all the requirements of the GUI. This GUI is used as maintenance station for the RDS.

4.5 PROFIBUS PMC implementation

A Siemens® 200 PLC with a PROFIBUS EM 277 module is used in the system as the alternative operating station. The PLC is responsible for controlling the motor drive of the RDS and has various safety alarms that continuously monitor the system. The PLC is connected through an RS 485 PROFIBUS A-type cable to the PROFIBUS PMC module connected on the compact PCI bus of the SBC, which allows the user to operate the RDS through the PLC. Figure 4.10 shows the PLC device that was used.

Figure 4.10 – Siemens S7-200 PLC

4.5.1 Implementation

The software environment SyCon® from Hilcher® Company was used to initialize the PROFIBUS network. This software allows the user to set up a PROFIBUS network by adjusting the various parameters regarding data rates, slave addresses and memory spaces used. Once the network was established, it was synchronized with the SBC GUI to communicate with the PLC by including the PROFIBUS drivers in the GUI software.

The PLC is programmed with the Siemens® STEP-7 environment tools. The PLC had to be programmed to allocate specific memory positions in the PLC, which the SBC can use to send data to and from the PROFIBUS PMC module. The PLC has various buttons that are used to levitate and delevitate the RDS rotor. A key switch is also connected to the PLC that changes control of the RDS between the SBC GUI and the PLC. If the key is turned in favour of the PLC, the SBC GUI hands over control of the system to the PLC interface (buttons and key). All this information is communicated over the PROFIBUS network. Figure 4.11 shows the SBC GUI when the PLC controls the system.

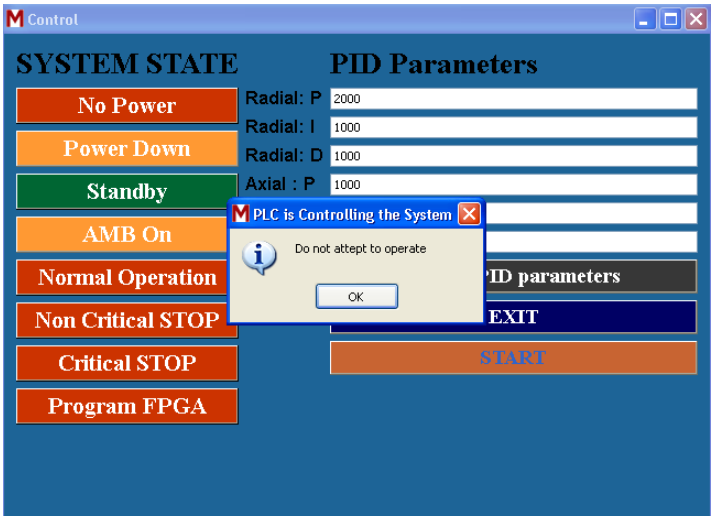


Figure 4.11 – GUI view when PLC has operational control

4.5.2 PROFIBUS DP physical layer implementation

PROFIBUS DP uses the RS 485 standard as discussed in chapter 2 and 3. Figure 3.4 showed the RS 485 bus segment and the specified termination resistors of the PROFIBUS DP standard. The termination resistors establish the necessary voltage and ground levels for the RS 485 differential signal.

The PROFIBUS A type cable, shown in figure 4.12, is used for the physical medium to conform to the PROFIBUS DP standard. The connectors are constructed with the termination resistors specified in figure 4.13. This figure illustrates the physical implementation as it was created for the PROFIBUS DP network.

Figure 4.12 – PROFIBUS A-type cable

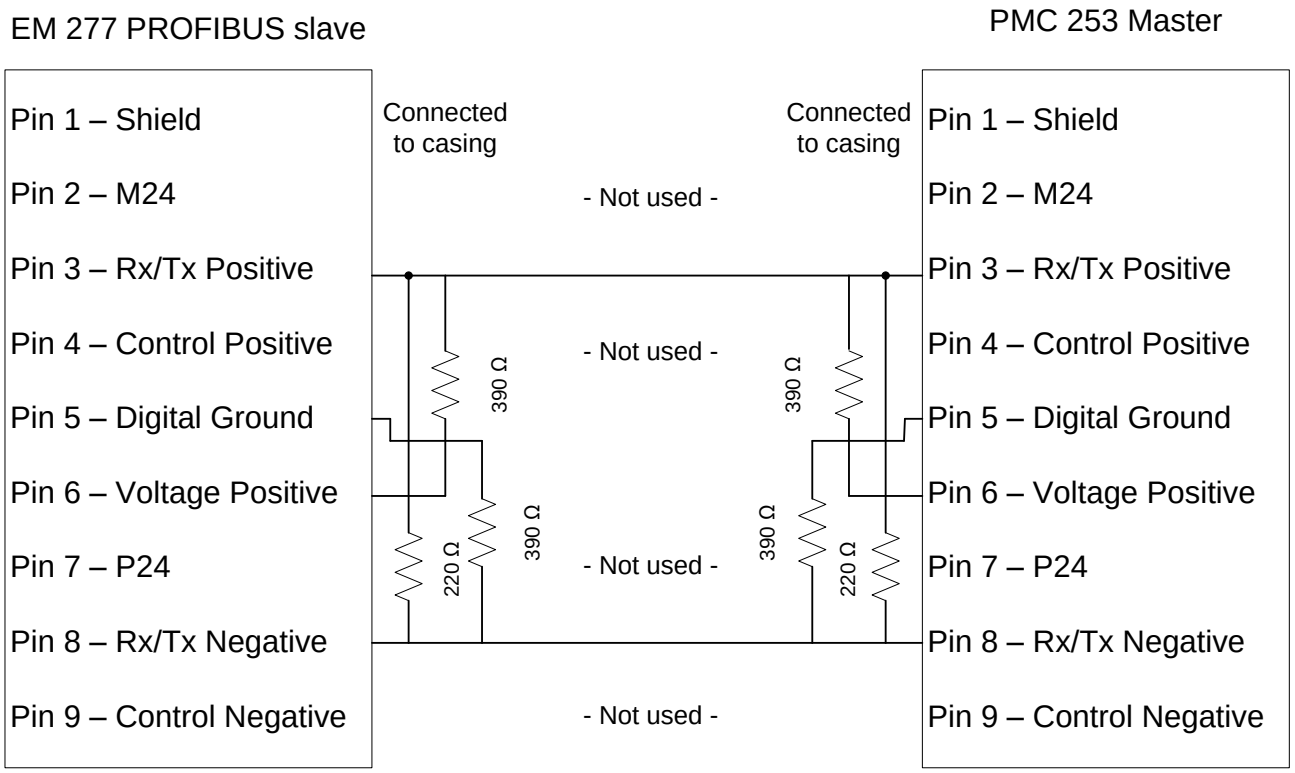


Figure 4.13 – PROFIBUS DP network physical setup

4.5.3 Verification and validation

The various software environments in which each section of the network was programmed were used to verify the entire network before it was tested in the actual RDS environment. Microsoft® Visual C++ was used to develop and test the new GUI software with the PROFIBUS drivers. SyCon® environment was used to set up the PROFIBUS network and do the initial parameterization. Finally the Siemens® STEP 7 environment was used to program the PLC to allocate specific memory spaces on the PLC for the PROFIBUS communication. This programming is shown in appendix C.

The RDS system was operated with the GUI on the SBC and the PLC by changing control between the stations with the key switch on the PLC. Both stations were successful in controlling the RDS and was able to operate the system in real-time. The PLC and GUI were synchronized with each other, which allow the user to change between operating stations any time during the RDS operation. The 1.5 Mbps data transfer speed of the PROFIBUS network allows this transition to happen in real-time, without an interference of the rotor position.

4.6 Conclusion

The ADES can be controlled with the GUI on the SBC or the PLC over the PROFIBUS network. Various software programs were developed, verified and validated to meet the specific requirements of the ADES for a Fieldbus communication network. The GUI on the SBC will mostly be used for maintenance and development while the PLC adds an industrial type operating system to the ADES.

It is however possible to implement the PROFIBUS protocol on the FPGA of the ADES. This will reduce the cost of the system by still keeping the PROFIBUS communication standard. Figure 4.14 displays the concept of how the ADES will function with the PROFIBUS protocol implemented on the FPGA. By comparing this figure with figure 4.1, the different implementation methods are clearly seen with regards to the PROFIBUS protocol. The next chapter will discuss the design and implementation of a PROFIBUS protocol on the FPGA of the ADES.

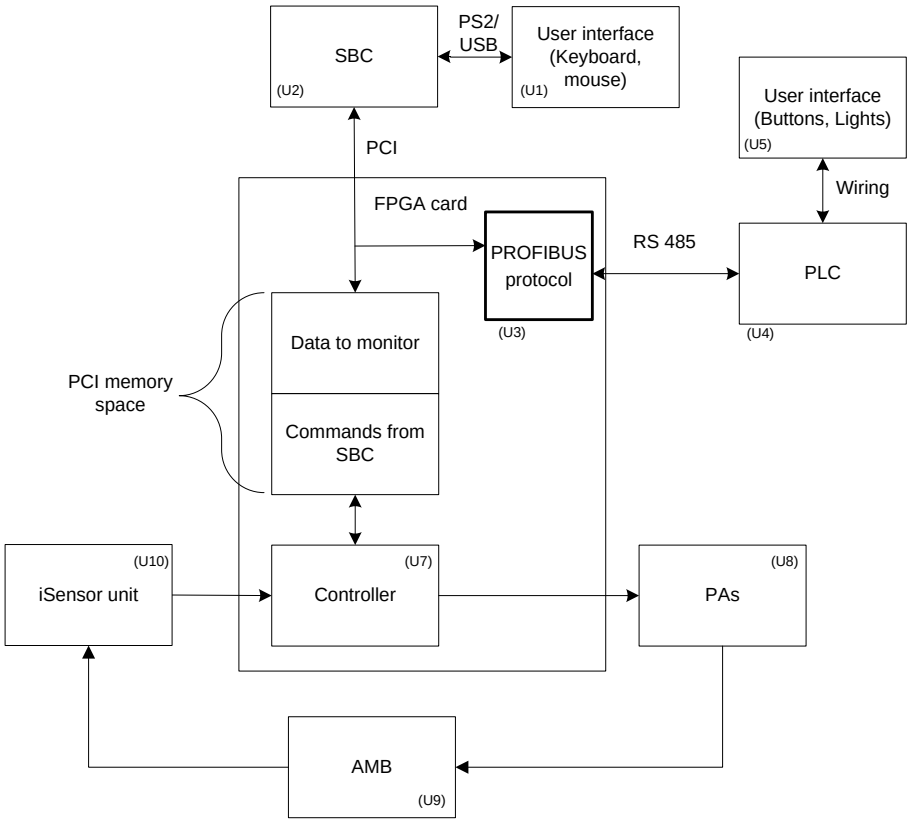


Figure 4.14 – Preliminary PROFIBUS protocol implementation

Chapter 5

Fieldbus communication system design

Chapter 5 begins by defining the requirements for the design of the PROFIBUS DP protocol, based on the PROFIBUS DP standard, as well as the ADES requirements. Once the main design flow is established, the focus shifts to the detail design and implementation of the PROFIBUS protocol on the ADES. The design of the PROFIBUS protocol will be conducted with regards to the applicable layers of the OSI reference model.

5.1 Introduction

In 1951, Maurice Wilkes of Cambridge University proposed a systematic technique to design and implement control units as an alternative to random logic. In this technique, the control unit's overall sequence was issued to the computer as a primitive instructional set encoded in the memory array. This technique was called microprogramming, and the instructional sequence became known as a micro program. The micro program's structure avoided much of random logic's arbitrariness and greatly facilitated computer design. Prior to 1970, the micro program was held in read-only memory until the advance in storage technology. This advance lead to writable-control-store availability. This enabled micro programmers to overwrite the contents of a micro program. This advance in technology allowed users to change a computer's behaviour by changing the contents of the writable-control-store. From this point, the term *firmware* became a synonymous for micro programs and emphasized the role of micro programs relative to software and hardware [40].

The implementation of VHDL on a FPGA can therefore be seen as a classic example of firmware, where a micro program (VHDL) is directly implemented on hardware (FPGA), with the possibility of overwriting the micro program. With this in mind the Fieldbus communication protocol will be designed. The first step in any software or firmware project is the detailed specification of what is to be accomplished [41].

Chapter 1 discussed the Fieldbus requirements for the active magnetic bearing drive electronic system (ADES). Based on the literature discussed in previous chapters, this chapter starts by defining the detail requirements needed to design a PROFIBUS protocol for an FPGA. The internal ADES requirements and environment will also be discussed to demonstrate the implementation of the PROFIBUS protocol on the system. The detail design process follows, illustrating how the PROFIBUS protocol is designed in VHDL and implemented on the FPGA.

5.1.1 PROFIBUS DP requirements

Chapter 3 discussed all the various aspects of the PROFIBUS DP protocol. However, not every aspect of the PROFIBUS DP protocol is mandatory to establish a communications network. Based on the requirements of the ADES, tables 5.1 to 5.3 briefly give the requirements of the PROFIBUS DP protocol that are mandatory. Table 5.1 gives the standard for a PROFIBUS character frame. Table 5.2 gives the PROFIBUS telegram format, while table 5.3 presents the different operating states of the PROFIBUS DP standard. Besides the PROFIBUS DP requirements, the following specifications with regards to the ADES also need to be implemented:

- High bit rate – to ensure fast reaction of control communications
- At least 16 bytes of cyclic data transfer – used for control and measurement data transfer (with expansion possibilities up to 32 bytes)
- Error free data transmission – to ensure safety of the rotor delevitation device
- Implementation on the Xilinx® Virtex©-5 FPGA of the ADES
- Watchdog timer – to ensure stability of the Fieldbus communication
- The same features as the PMC 253 PROFIBUS module

Table 5.1 – PROFIBUS character frame specifications [26]

Requirement	**Specification**
Non-return to zero data coding	Data stays active low or high during each bit
1 Start bit	Logic ‘0’
8 Data bits	Data byte to be transferred
1 Parity bit	Even parity
1 Stop bit	Logic ‘1’

Table 5.2 – PROFIBUS telegram frame specifications [26]

Telegram A character requirement	**Character specification**
SD - Start delimiter	10H (hexadecimal) value – indicates the beginning of telegram
DA - Destination address	Address of receiver
SA - Source address	Address of transmitter
FC - Function code	Used to identify the type of telegram
FCS - Frame check sequence	Cyclic redundancy check – calculated by adding the binary values of the telegram characters
ED - End delimiter	16H value – indicates the end of the telegram
Telegram B character requirement	**Character specification**
SD - Start delimiter	68H value – indicates the beginning of telegram
LE - Length	Net data length of the telegram
LEr -Length repeated	Net data length of the telegram (for redundancy)
SD - Start delimiter	68H value (for redundancy)
DA - Destination address	Address of receiver
SA - Source address	Address of transmitter
FC - Function code	Used to identify the type of telegram
DSAP - Destination service access point (Optional)	Used by the receiver of the telegram to determine which service should be executed
SSAP - Source service access point (Optional)	Used by the transmitter of the telegram to determine which service should be executed next
DU - Data units	1 to 244 bytes
FCS - Frame check sequence	Cyclic redundancy check – calculated by adding the binary values of the telegram characters
ED - End delimiter	16H value – indicates the end of the telegram

Table 5.3 – PROFIBUS state machine specifications [26]

Requirement	**Specification**
Initialization state	• Master detects slave devices on the network • Slave initialize itself and automatically detect the bit rate
Request diagnostic 1 state	• Master request the diagnostic data of the slave • Master determine if parameterization and configuration must occur before the data exchange state
Parameterization state	• The master specifies the mode in which the slave must operate
Configuration state	• The master specifies the amount of input and output bytes that are to be exchanged in each telegram cycle with the slave
Request diagnostics 2 state	• Master re-request the diagnostic data of the slave • Master determine if parameterization and configuration has completed successfully before the data exchange state
Data exchange	• The master and slave cyclically exchanges I/O data as specified in the configuration state

During the data exchange state, data is transferred between the master and the slave on the PROFIBUS network. In the ADES environment these data units will be specified and have specific purpose. These data values are discussed in the detail design section. Before the requirements and specifications of the PROFIBUS DP protocol can be implemented, it is necessary to understand the platform on which this protocol will be implemented.

5.1.2 ADES platform specification

The ADES platform was discussed in detail in chapter 4. This section briefly discusses the ADES platform with special focus on the PROFIBUS implementation on the ADES. Figure 5.1 displays the interaction of the ADES functional units. In this figure the original PROFIBUS PMC module (U3) can be seen. Unit CU1 represents the conceptual implementation proposition for the VHDL PROFIBUS protocol. Once the PROFIBUS protocol is correctly implemented on the FPGA, the PROFIBUS PMC module will be removed from the system. As shown in figure 5.1, the PLC will be directly connected to the FPGA after correct implementation.

The ADES consists of 10 main functional units. The purpose of the ADES is to operate and control the power amplifiers (PAs), which in turn control the active magnetic bearing (AMB) system. Two user interface options are available; by means of the graphical user interface (GUI) attached to the single-board computer, or the buttons connected to the PLC. The PROFIBUS network is necessary to communicate with the control of the power amplifiers and levitate or delevitate the rotor of the RDS. In table 5.4 the functional units of the ADES are discussed with regards to their specific function.

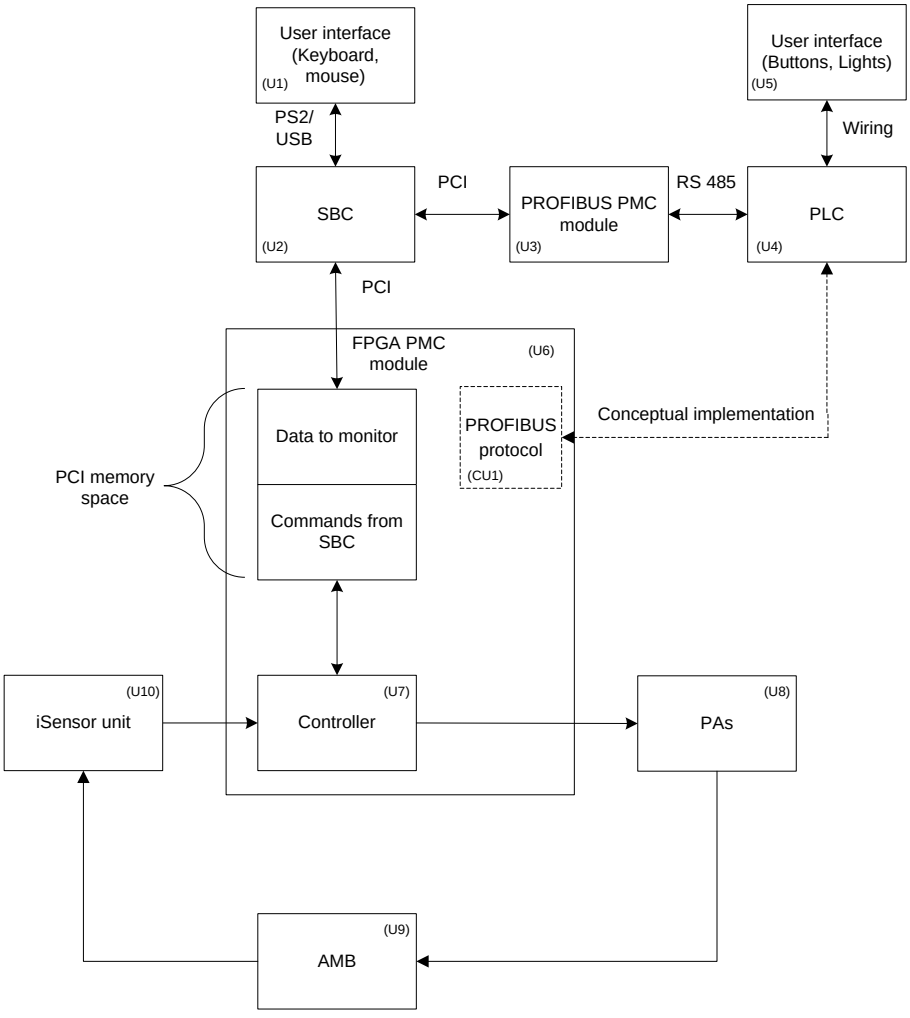


Figure 5.1 – ADES functional units

Table 5.4 – ADES functional units

	Unit	**Function**
U1	Graphical user interface	Allows a user to operate the rotor delevitating system (RDS).
U2	Single-board computer	Functions as the connection backbone for the entire system. The FPGA, PROFIBUS PMC module, keyboard, mouse and display screen all connect through the single-board computer.
U3	PROFIBUS PMC module	Functions as a PROFIBUS master to control the communication with the slave PLC device. Control data is transmitted between the PLC, single-board computer and the FPGA through the PROFIBUS network.
U4	PLC	The programmable logic controller transfer and receive control data over the PROFIBUS network. The PLC is also used to measure the temperatures in the ADES and triggers alarms during faulty operation.
U5	PLC interface	The PLC interface consists of buttons and lights – connected to the PLC – to operate the AMBs.
U6	FPGA	The Xilinx® FPGA is programmed to control the entire internal communication flow of the ADES. Data is received from the inductive sensor unit and the power amplifiers are adjusted to control the AMBs. PID control is implemented on the FPGA to control the AMB system.
U7	Controller	PID controller. Implemented on a PowerPC core embedded on the FPGA
U8	Power amplifiers	Generates the electromagnetic forces to levitate and delevitate the rotor
U9	Active magnetic bearings	Suspends a rotor with magnetic forces, allowing for a frictionless bearing system.
U10	Inductive sensor unit	Measures the displacement of the rotor in the rotor delevitating system and transfers the data back to the PID controller.
CU1	PROFIBUS protocol	The PROFIBUS protocol will incorporate the entire PROFIBUS standard to establish a PROFIBUS network between the PLC and the FPGA without the PROFIBUSPMC module.

In order to replicate the PROFIBUS PMC module on the FPGA, the same functionality of the original system must be incorporated together with the PROFIBUS DP specifications for an efficient design. The following section briefly discusses the proposed ADES with the VHDL PROFIBUS protocol implemented directly on the FPGA.

5.1.3 Final Fieldbus communication system

Figure 5.2 illustrates the ADES after implementing the PROFIBUS protocol on the FPGA. In this illustration, the PROFIBUS PMC module is completely removed from the system. The developed PROFIBUS protocol will operate as a PROFIBUS master device to control the PLC slave. To achieve this, the PLC will be directly connected to the FPGA by incorporating the specified RS 485 standard as the physical layer. The PROFIBUS protocol will be instantiated on the FPGA together with internal communication control on the FPGA. The parallel processing of the Xilinx® Virtex©-5 FPGA allows various operations to be executed simultaneously.

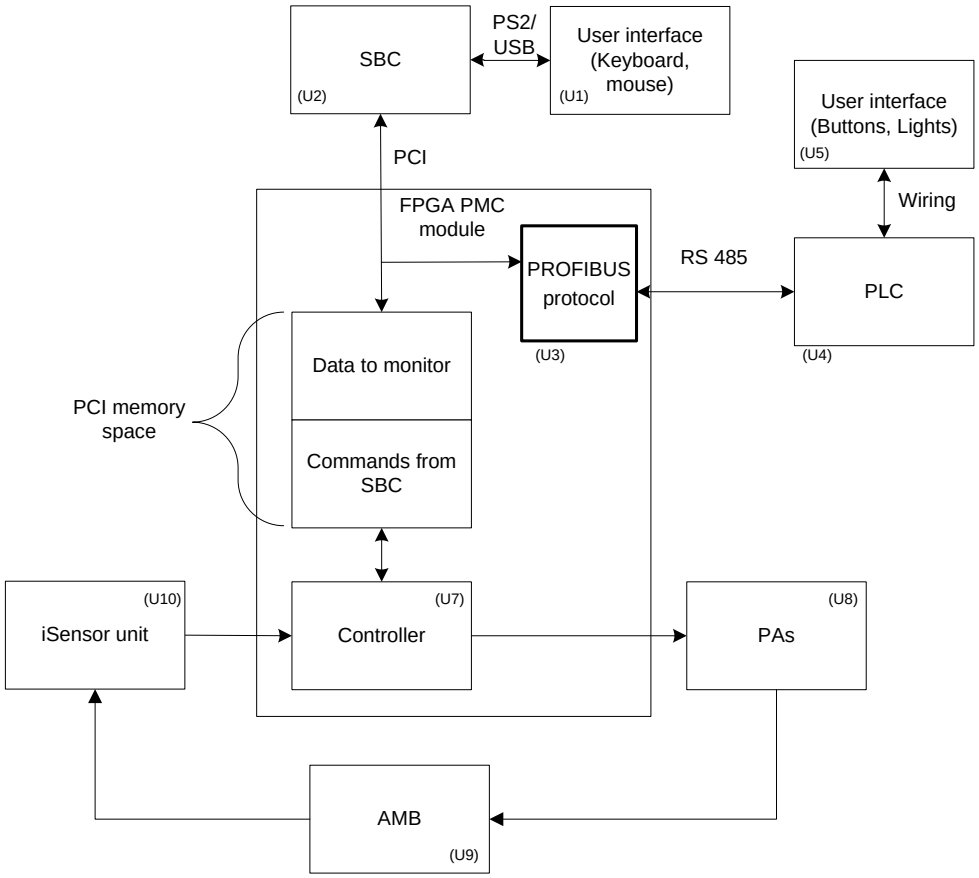


Figure 5.2 – Conceptual PROFIBUS protocol design interface

Besides the PROFIBUS requirements that need to be implemented, the PROFIBUS protocol must be interfaced with the controller (U7 in figure 5.2) on the FPGA and the PCI bus. This will allow control and measurement data to be sent and received from and to the PLC. The PCI bus allows data to be transferred between the FPGA and the single-board computer. All these requirements will be used to design the specific PROFIBUS DP protocol and interface.

5.2 PROFIBUS DP communication system design

The previous section discussed the requirements for a Fieldbus network on the ADES together with the specific PROFIBUS specifications that are needed to implement the protocol. By using this information, the design and implementation of the PROFIBUS protocol on the FPGA are discussed in detail in this section.

5.2.1 Design process

Before designing the protocol, a design process must be established. This ensures that the correct engineering approach is followed to achieve a high-quality design. Figure 5.3 illustrates the development process followed in this project. This figure can be used to summarize the entire project from the requirement and analysis phase up until the validation of the design. The left side of the figure represents the major flow of procedures followed together with the corresponding sub-processes followed in each section. Each chapter in this dissertation provides an explicit process needed to develop the PROFIBUS protocol for the Xilinx® FPGA. The development process shown in figure 5.3 illustrates how each part of the study in indispensable in engineering design.

It is essential to see the entire study from the initial requirements up until verification and validation as an interconnected sequence of events that cannot be separated from each other. This ensures that each part of the engineering approach is done with specific purpose and motivation.

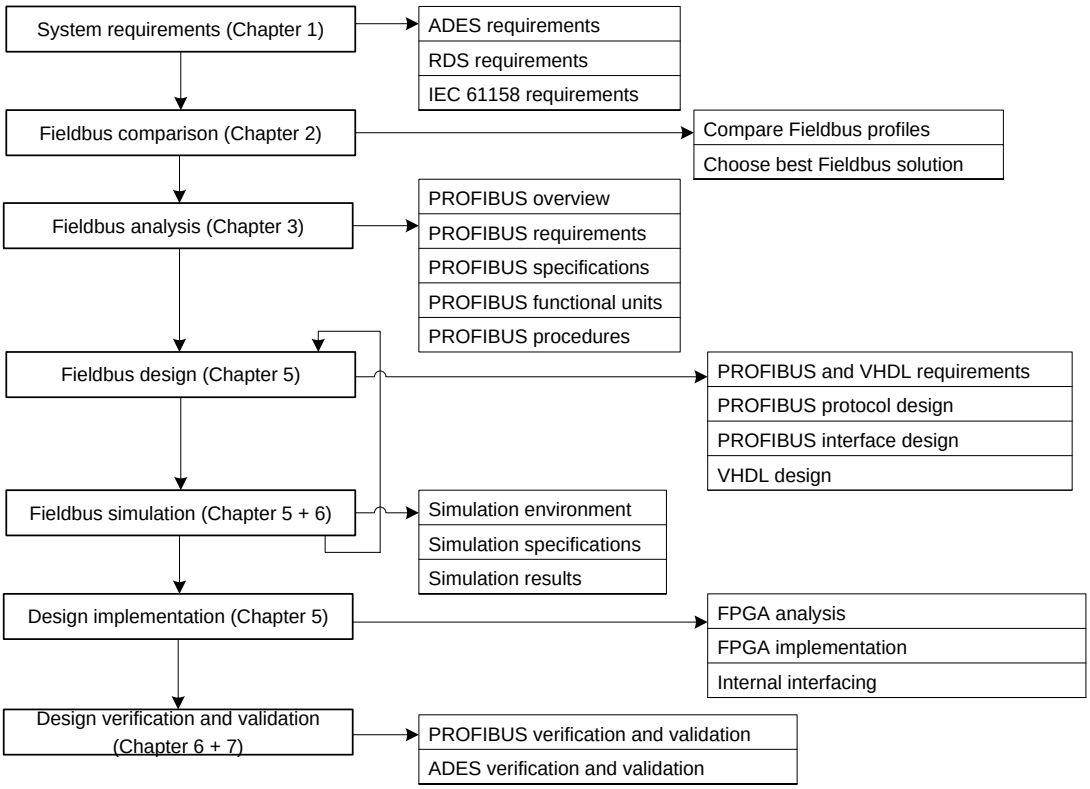


Figure 5.3 – PROFIBUS protocol development process

5.2.2 PROFIBUS protocol design process

In figure 5.4, the design process of the PROFIBUS protocol is illustrated. Each part of the design process progresses towards the final implementation on the ADES. This design flow is used to establish the design methodology and order. This figure represents how the initial requirements are derived and analysed from the ADES and PROFIBUS DP standard. By using these requirements, functional units and processes are determined for the physical and data link layer. Finally each layer is designed and interfaced to be implemented on the ADES.

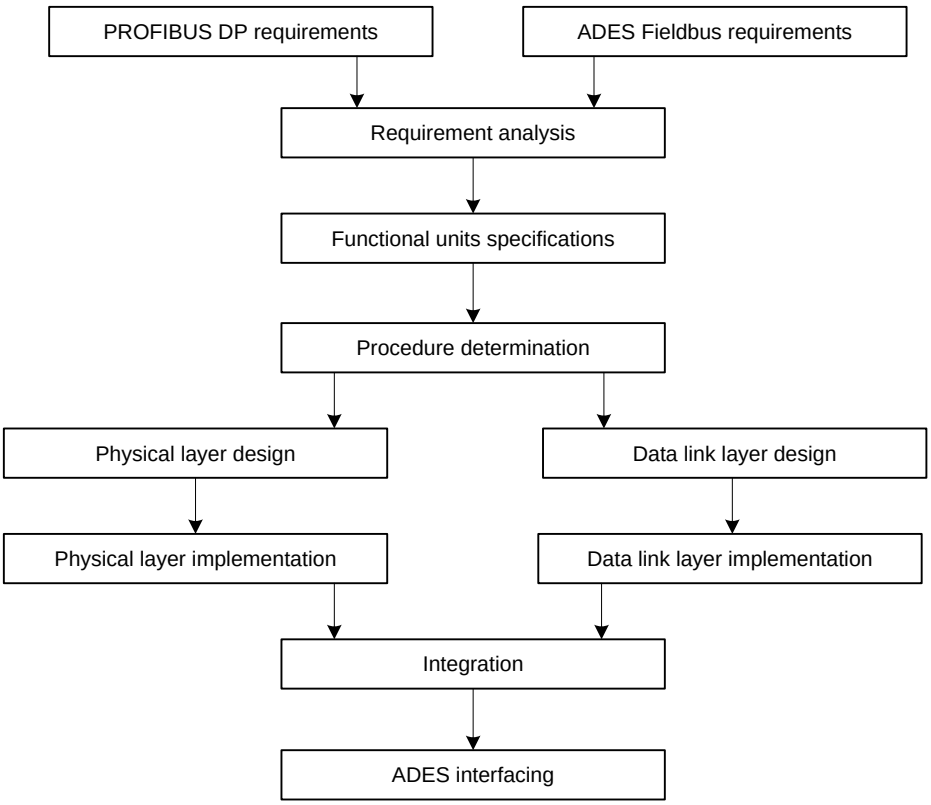


Figure 5.4 – PROFIBUS protocol design and implementation flow

5.2.3 Protocol unit specifications

The requirements and specifications of the ADES and the specific PROFIBUS protocol are analyzed in the previous sections. The next step in the design process is to create specific functional units that will be used to address each requirement and specification. These units will interface with each other to form the final design model. The protocol units with their application are discussed in table 5.5. Figure 5.5 demonstrates how the units will interface with each other.

Table 5.5 A – PROFIBUS protocol units specifications

Functional unit	**Function**
Main controller	• Interface for the functional units • Control execution of events
UART transmitter	• Receive parallel transmitting data from main controller • Add start, stop and parity bits to the data • Convert parallel data to serial data and transmit over serial channel to the PLC device

Table 5.5 B – PROFIBUS protocol units specifications continued

UART receiver	• Receive serial data from PLC over serial channel and convert to parallel data • Remove start, stop and parity bits from the data • Alert the UART transmitter if parity or CRC errors are present in the received data • Transfer the parallel data to the main controller
State machine	• Produce state-related data to the main controller • Interpret data received from the PLC • Determine the next state to be executed
Clock generation	• Generate a clock signal to synchronize all the functional units
ADES interface	• Read data on the PCI bus that is designated for the PLC
FPGA main control interface	• Transfer control data from the PLC to the PID control unit of the FPGA
Physical Unit – RS485	• Establish physical connection with the PLC and the FPGA

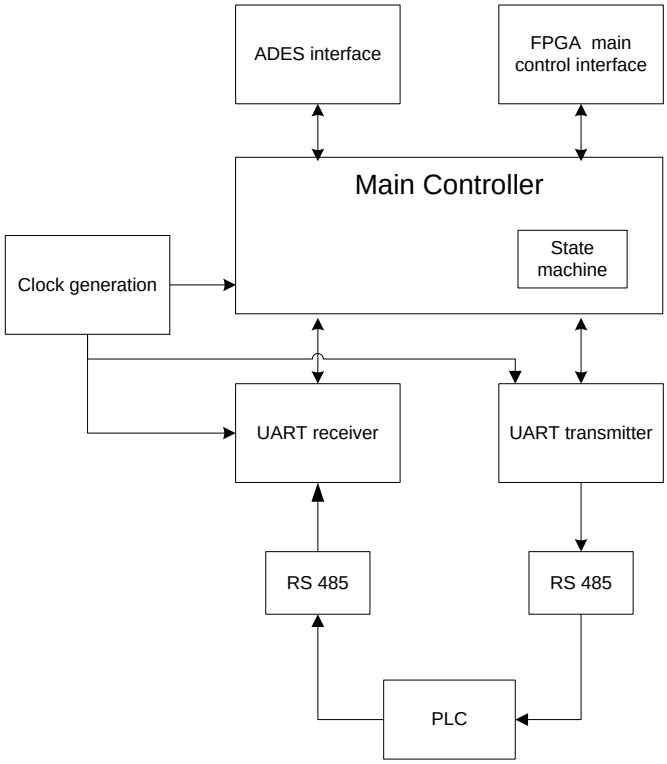


Figure 5.5 – PROFIBUS functional units design

5.2.4 PROFIBUS DP procedures and internal signals

The next step in the design process is to determine the procedures that each functional unit has to execute and how the units are connected internally. By using the functional units that were discussed in the previous section together with the detail PROFIBUS and ADES requirements, the procedures are determined and allocated to specific units. Table 5.6 to 5.11 shows how the different procedures relate to the different functional units together with their specified functions.

Table 5.6 – ADES interface procedures

Procedure	**Function**
Transmit data to ADES	Designated data from the PLC is sent to various sections of the ADES
Receive data from ADES	Designated data from the ADES is sent to the PLC

Table 5.7 – Main controller procedures

Procedure	**Function**
Parallel data transfer	Parallel data that is designated for the PLC, is transmitted to the UART transmitter
Parallel data capture	Parallel data that is received by the UART receiver, from the PLC, is obtained
State machine control	The complete PROFIBUS state machine is controlled by the main controller
State machine procedures:	
Data evaluation	Received data is evaluated to determine if the previous state was executed successfully
Determine state	Based on the data evaluation, the next state to be executed is determined
Determine data for transfer	Once the next state is determined, the related data to be transmitted is determined

Table 5.8 – FPGA interface procedures

Procedure	**Function**
Main clock synchronization	The main clock of the FPGA is used to synchronize the PROFIBUS activities with that of the FPGA
Transmit data to PID controller	Control data received from the PLC is sent to the PID controller implemented on the FPGA

Table 5.9 – Clock generation procedures

Procedure	**Function**
Main clock synchronization	The main clock of the FPGA is used to synchronize the PROFIBUS activities with that of the FPGA
PROFIBUS clock synchronization	By using the main clock received from the FPGA, a clock signal used for the PROFIBUS operations is generated

Table 5.10 – UART transmitter procedures

Procedure	**Function**
Parallel data capture	Data that need to be transmitted to the PLC is received from the main controller
Parity transmission	Parity bits are added to the received data
CRC transmission	Cyclic redundancy checks are added in the ‘Check Frame Sequence’ character of the telegram data
Parallel to serial conversion	Parallel data is converted to serial data at a specified bit rate and transmitted over the serial channel
Error handling	If an error is received from the UART receiver, the previous data will be resent to the PLC
Bit rate generator	Using the PROFIBUS clock signal, the bit rate is calculated for transmission
Direction control	The direction of the half-duplex PROFIBUS channel is controlled. This signal is sent to the RS 485 driver in the physical layer

Table 5.11 – UART receiver procedures

Procedure	**Function**
Serial to parallel data conversion	Serial data received from the PLC over the PROFIBUS channel is converted to parallel data
Parity evaluation	The parity bits of all the received data is determined and checked against the received parity bits
CRC evaluation	The frame check sequence character of the received data is evaluated to detect transmission errors from the PLC
Parallel data transfer	The converted parallel data is transmitted to the main controller for evaluation
Bit rate detector	Using the PROFIBUS clock signal, the serial data from the PLC is sampled with conjunction to the specified bit rate
Watchdog timer	If no data is received from the PLC for a specified amount of time, an error will be flagged
Error detection	When an error is detected by the parity evaluation, crc evaluation or the watchdog timer, the UART transmitter will be notified in order to resent the previous data.

By using the different functional units together with their specified procedures, the detail design of the developed PROFIBUS protocol is done. Figure 5.6 displays a diagram of how the protocol units are connected to fulfil the requirements of the PROFIBUS standard as well as the ADES. In figure 5.6, each protocol unit is presented together with the corresponding procedures. The main functional units of the design can still be seen as designed to acquire the PROFIBUS protocol units and the various procedures. The physical layer is discussed in detail in the next section.

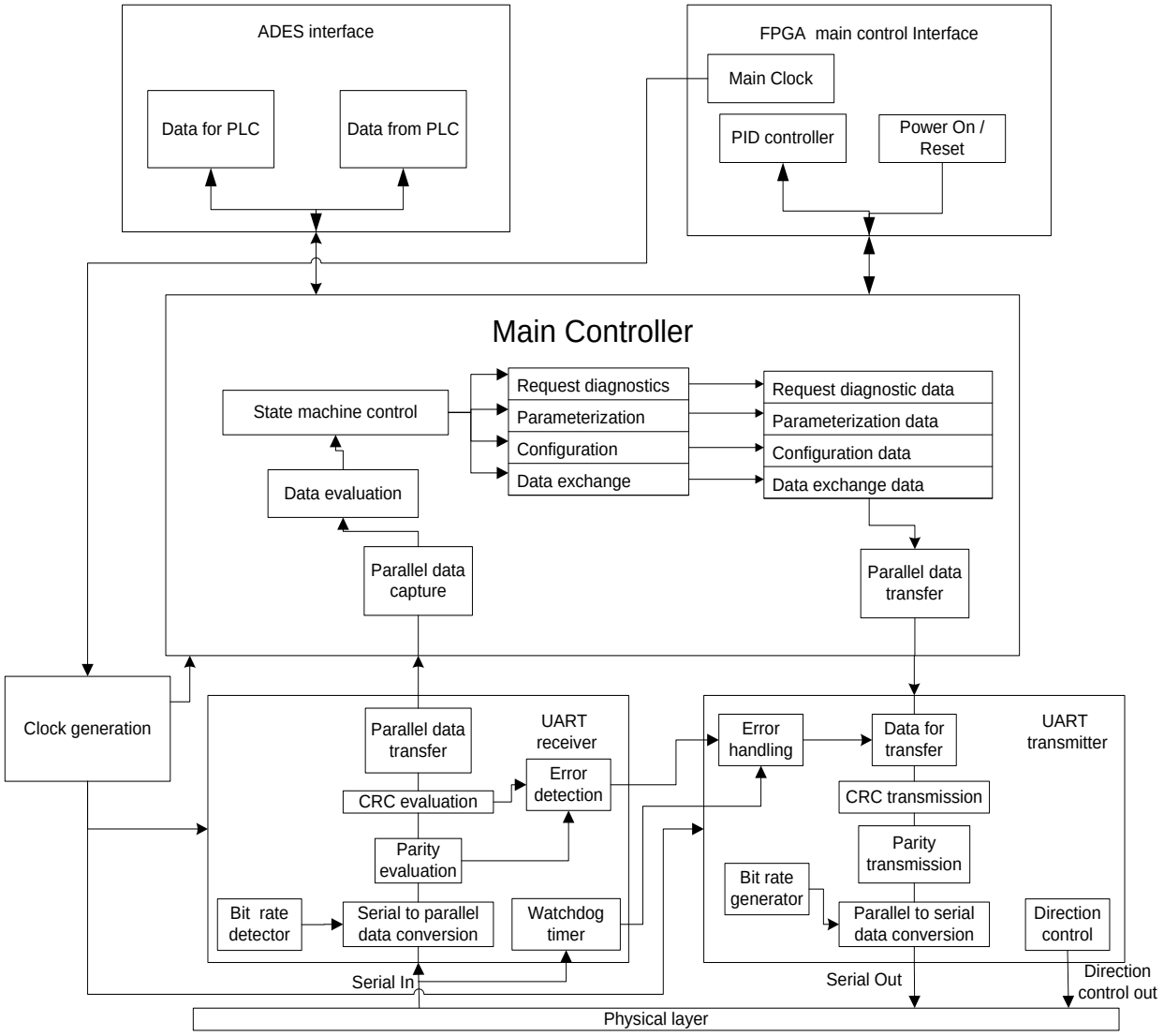


Figure 5.6 – PROFIBUS protocol detail design

5.2.5 PROFIBUS DP protocol design

The next step in the design process, illustrated in figure 5.4, is the detail design of the various layers of the OSI reference model, which is applicable to the PROFIBUS protocol. As discussed in chapter 3, PROFIBUS DP utilizes the physical and data link layer. This section explains how the defined functional units and their processes are designed and implemented with regards to the OSI reference model.

5.2.5.1 Physical layer design

The physical layer addresses the mechanical and electrical aspects of the communications link and provides the means to transmit bits of data across a continuous communications path [7]. The physical layer aspects of the PROFIBUS protocol include the bus line, transmission procedure, bus connection, bus control and bus termination.

Bus line

The PROFIBUS standard specifies the type A cable (shown in figure 5.7) to be used as the bus line for PROFIBUS networks. The properties of this copper, shielded twisted pair cable is shown in table 5.12. Using twisted pair implies that half-duplex communication is achieved when transmitting data as RS 485 differential signals. This cable is installed on the ADES as the PROFIBUS bus line.

Figure 5.7 – PROFIBUS type A cable

Table 5.12 – PROFIBUS RS 485 type A cable specifications

Surge Impedance	135 to 165 Ω, at a measuring frequency of 3 to 20 MHz
Cable Capacitance	< 30 pF per meter
Core Cross Section	> 0.34 mm²
Cable Type	Twisted pair, 1 x 2 or 2 x 2 or 1 x 4 conductors
Loop resistance	< 110 Ω per km
Signal Attenuation	9 dB maximum over 100m of the cable section
Shielding	Braided copper shield, braided shield or foil shield

Transmission procedure

PROFIBUS uses the RS 485 transmission procedure, which is based on half-duplex, asynchronous gap-free synchronization. Non-return to zero (NRZ) line coding is used to transmit the 11-bit character frame, which implies that during transition from binary ‘0’ to ‘1’, the shape of the signal does not change while the bits are being transmitted [33].

Figure 5.8 displays how data bits are transmitted over the two lines of the PROFIBUS cable. Line B and A represent the two channels in the PROFIBUS cable, while the binary signal represents the corresponding data bit transmitted at that moment in time. The voltage levels of line A and B are inversed with a voltage level of $|V_B| - |V_A| > 0.2\,V$. At the receiver end the resulting voltage level are determined by determining the voltage difference between the two lines. Since $|V_B| - |V_A| > 0.2\,V$, the receiver can determine the binary value with the difference between the two channels. A resulting voltage difference larger than 0.2 V indicates a binary ‘1’, while a resulting voltage difference smaller than -0.2 V indicates a logic ‘0’.

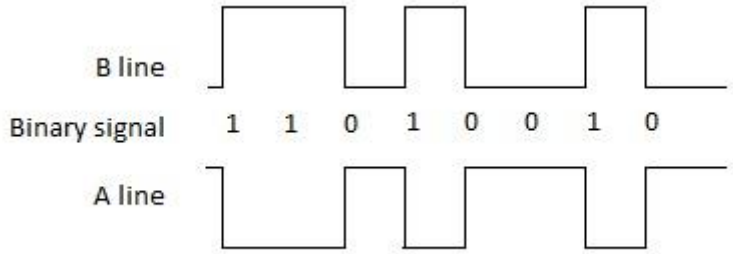


Figure 5.8 – Non-return to zero transmission

Bus control, connection and termination

In order to transmit data over the PROFIBUS network, the serial data has to be converted to RS 485 differential signals as discussed in the previous section. An RS 485 transceiver from Sipex® is used to convert the serial data to RS 485 differential signals. The Sipex® SP491E driver is seen in figure 5.9. This driver uses two operational amplifiers to convert the input serial signal from the FPGA into two ‘mirror’ signals that meet the RS 485 differential standard. When data is received from the PLC slave, the differential signals are converted back to a single serial signal and transmitted to the FPGA via the RS 485 driver. A direction control signal from the FPGA is used to control the driver to transmit or receive data.

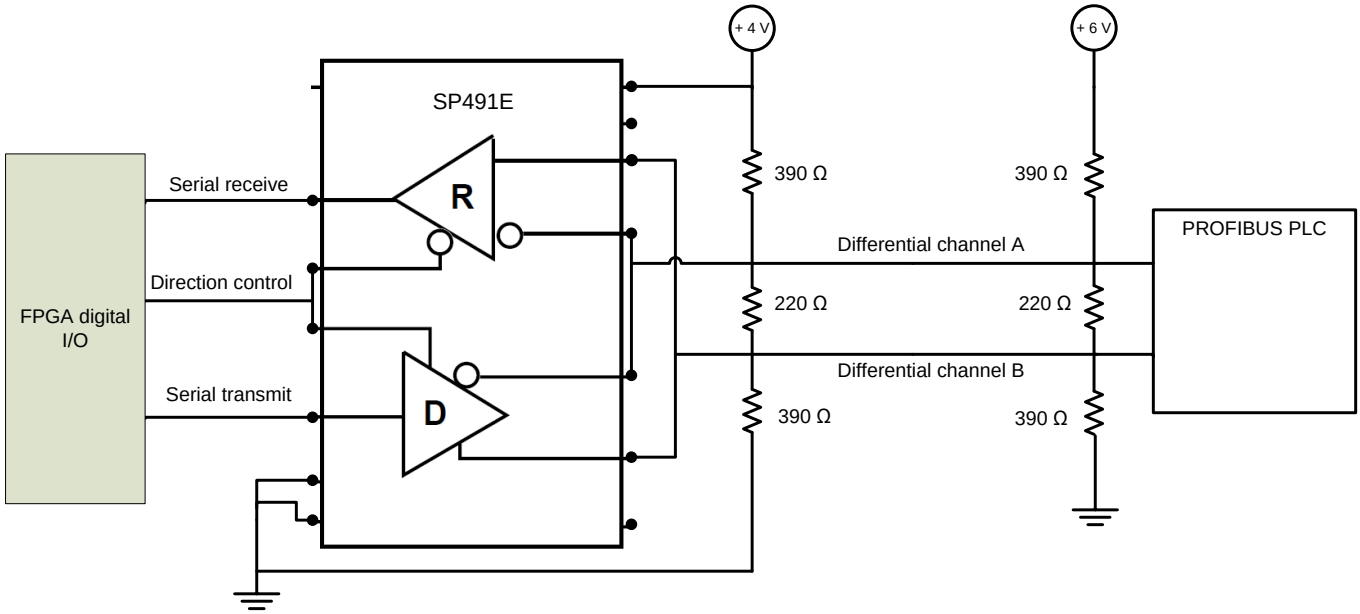
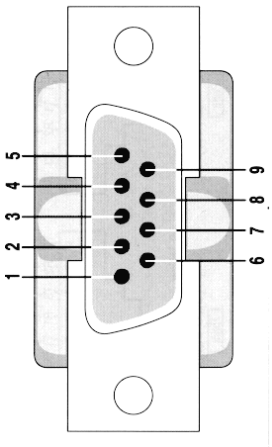


Figure 5.9 – Physical layer design

In figure 5.9, the bus termination can be seen as implemented through a two-sided bus terminating resistor, as well as a pull-up resistor connected to the supply voltage and a pull-down resistor connected the digital ground data reference potential. These three resistors are implemented at the transmitter as well as the receiver end. These resistors ensure a defined idle potential on the bus line when the bus line is in the idle state between two telegrams. The PROFIBUS standard defines the specific values of the resistors as shown in figure 5.9.

Table 5.13 – Pin assignment of 9-pin connector [33]

Connector View	**Pin Number**	**Signal Name**	**Designation**
	1	Shield	Shield, protective ground
	3	RxD/TxD-P	Receive/Transmit differential (line P)
	5	DGND	Digital ground
	6	VP	Voltage plus
	8	RxD/TxD-N	Receive/Transmit differential (line N)
	2,4,7,9		Not connected

To connect two PROFIBUS devices, the PROFIBUS standard recommends a 9-pin connector for the interconnection of bus stations through the bus line. Table 5.13 displays the 9-pin connector with the pin assignment. This figure was discussed in chapter 3; however in table 5.13 the implemented pin connections are displayed.

5.2.5.2 Data link layer design

This section discusses the design and implementation of the PROFIBUS data link layer. The entire data link layer is designed in VHDL and implemented on the Xilinx® FPGA. Firstly, the design of the PROFIBUS specifications are discussed where after the implementation for the FPGA with regards to the ADES are discussed. All the PROFIBUS specifications discussed in this section are acquired from “PROFIBUS specification” and “Introduction to PROFIBUS” [16] [26].

PROFIBUS character frame

Each PROFIBUS character comprises 11 bits, which include 1 start bit, 8 data bits, one parity and one stop bit. When no data is being transmitted, the idle state potential on the line is ‘1’. The start bit causes the line to go to ‘0’ and indicates a new character for the receiver. An even parity bit is calculated and also added to the character. The stop bit causes the line to go to ‘1’ while the receiver waits for the next low start bit. The eight bits of data in each character are transmitted in the order of least significant bit (LSB) to most significant bit (MSB) [16]. Figure 5.10 illustrates how this character specification is implemented in VHDL for the FPGA.

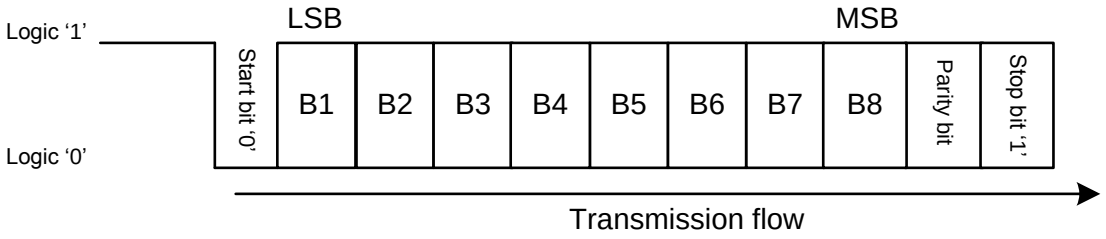


Figure 5.10 – PROFIBUS character design

PROFIBUS telegram frame

By combining the PROFIBUS characters in the previous section, telegram frames are constructed. Table 5.2 lists the various PROFIBUS characters used to construct different telegrams. There are 5

different PROFIBUS telegrams that are used for different purposes, but not all are mandatory. Figures 5.11 A, B and C display the three telegrams used in this PROFIBUS protocol design.

In each telegram, a frame check sequence (FCS) character is added. PROFIBUS specifies this character for cyclic redundancy checks of each telegram. The transmitter generates the data for the FCS, based on the characters in the telegram and transmits it to the receiver. Upon reception, the receiver will also determine the value of the FCS character and check it against the received FCS character. If an error is detected, the transmitter is notified or the telegram is resent.

To illustrate the calculation of the FCS character, consider the following telegram, which represents telegram frame 1. Six characters are transmitted which include a start and end delimiter character. In this telegram, the FCS character is determined by calculating the binary values of the characters excluding itself as well as the start and end delimiters:

DA – Destination address – 00000001

SA – Source address – 00000000

FC – Function code – 00111000

The resulting FCS character has data value 00111001. The FCS is always calculated by adding the telegram characters, excluding the start delimiter, stop delimiter and length characters. It should be noted that when the accumulating result causes a carryover of the eighth bit, the carryover is ignored.

Figure 5.11 A illustrates a telegram with a fixed information section and no data field. A master station uses this telegram to look for new stations on the bus line. The characters in figures 5.11 A, B and C are discussed in table 5.2.

Figure 5.11 A – PROFIBUS telegram frame 1

Figure 5.11 B and C illustrate telegrams with variable information section and data field lengths. Data lengths of 1 to 244 bytes can be transmitted in a single telegram. The variable information sections are also seen in these two figures. In figure 5.11 B, the characters DSAP (destination

service access point) and SSAP (source service access point) are included but not in figure 5.11 C. The telegram in figure 5.11 B is used in parameterization and configuration, while the telegram in figure 5.11 C is mainly used in the data exchange state. These states will be discussed in the next section.

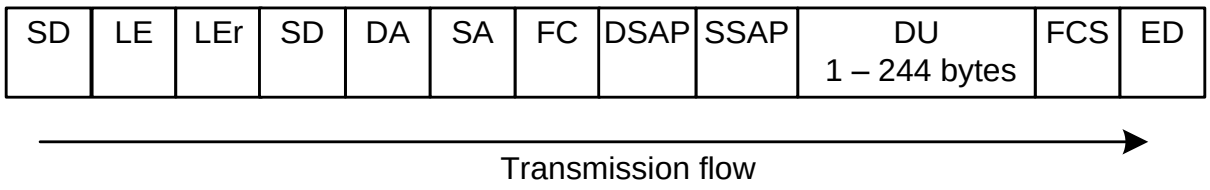


Figure 5.11 B – PROFIBUS telegram frame 2

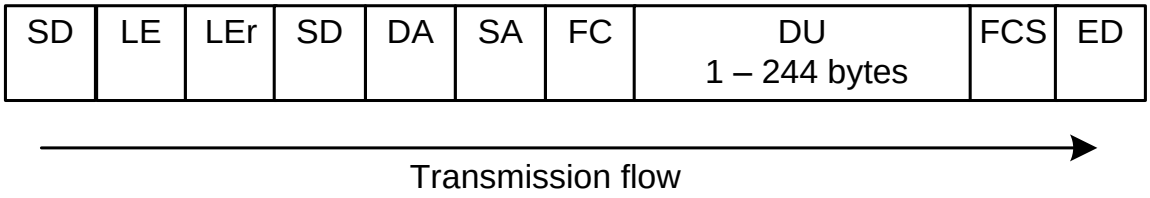


Figure 5.11 C – PROFIBUS telegram frame 3

State machine

PROFIBUS specifies eight states in which a PROFIBUS network can operate. These states are dependent on the specific application and not mandatory to establish a PROFIBUS network. The three optional states in the PROFIBUS state machine is not implemented in this design. Figure 5.12 illustrates the interconnection and flow of the implemented states. The PROFIBUS states include the following:

1. **Power on/Reset**
2. **Request diagnostics 1**
3. Change station address (optional)
4. **Parameterization**
5. **Configuration**
6. Request diagnostics 2
7. **Data exchange**
8. Global control (optional)

Each implemented state forms an essential part in establishing communication between the master and slave device. During each state, specific telegrams are transmitted between the master and slave to perform definitive functions. Each of these states is discussed in detail in the next section.

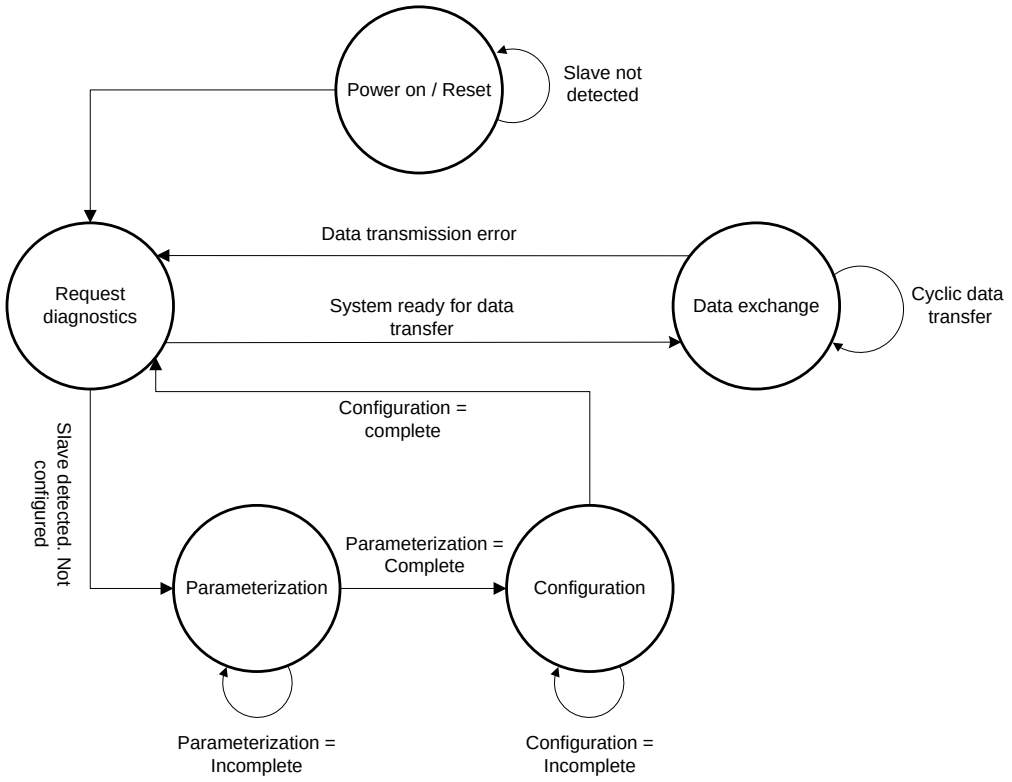


Figure 5.12 – PROFIBUS state machine

Power on / Reset

The power on / reset state is the initial state following power up or a reset of the entire system. The master transmits a telegram frame 1 continuously to detect slave devices on the network. The slave will initiate itself and detect the correct bit rate for communication automatically. The station address of the master is physically set on the PLC to be '00000001'. After the slave is fully initiated, it will respond to the initiation telegram sent by the master device. When the master receives the response telegram, it will continue to the request diagnostics state to determine the configuration of the slave. In table 5.14 the initialization telegram transmitted by the master is illustrated together with the value of each character. A slave device will respond with the same telegram, with the exception of the function control character. The function control character with a positive acknowledgement data value, received from the slave, will specify a slave device on the PROFIBUS network.

Table 5.14 – Power on / Reset telegram

SD	**DA**	**SA**	**FC**	**FCS**	**ED**
00010000	00000001	00000000	01001001	01001010	00010110
FC: Specifies the function of the telegram as – “Request data link layer status with reply”					

Request diagnostics

The request diagnostics state is used by the master to request diagnostic data from the slave after initialization. This state typically occurs after start up, before sending the parameterization telegram, and then again after configuration, before the data exchange state can occur. The master will evaluate the diagnostic information received from the slave to determine if the parameterization and configuration information is correct. If no further diagnostic service is needed, the master will proceed to the data exchange state. If the received configuration data from the slave in incorrect, the master will continue to the parameterization and configuration states [16]. Telegram frame 2, illustrated in figure 5.11 B, is transmitted in this state. Table 5.15 displays the specific telegram values for this telegram

Table 5.15 – Request diagnostic telegram frame

SD2	**LE**	**LEr**	**SD**	**DA**	**SA**	**FC**	**DSAP**	**SSAP**	**FCS**	**ED**
68H	00000101	00000101	01101000	10000001	10000000	01110111	00111100	00111110	11110010	16H

- LE and LEr = decimal value of 5. (DA + SA + FC + DSAP + SSAP)
- FC: Specifies the function of the telegram as – “Request diagnostic data”
- DSAP: Read diagnostic data
- SSAP: Check configuration data
- FCS = DA + SA + FC + DSAP + SSAP

The slave device will respond to the “request diagnostics” telegram with the same telegram as shown in table 5.15, with the addition of 6 data characters. The master uses these data characters to evaluate the diagnostic information of the slave and to determine the next state to be executed. The specific data units received in the request diagnostics state are discussed in detail in appendix G.

Parameterization

After the request diagnostic state, the master will determine if parameterization and configuration must occur before the data exchange state. The parameterization state is used to set the parameters of a slave by the master. A parameterization telegram will contain at least 7 bytes of specific information required by the PROFIBUS standard [16] [26]. Table 5.16 displays the telegram used in the parameterization state.

Table 5.16 – Parameterization telegram frame

SD2	**LE**	**LEr**	**SD**	**DA**	**SA**	**FC**	**DSAP**	**SSAP**	**DU**	**FCS**	**ED**
68H	00001100	00001100	68H	10000001	10000000	01101101	00111101	00111110	7X	00001110	16H

- LE and LEr = decimal value of 12. (DA + SA + FC + DSAP + SSAP + Data units)
- FC: Specifies the function of the telegram as – “Send and request data (high priority)”
- DSAP: Send parameterization data
- SSAP: Check configuration data
- FCS = DA + SA + FC + DSAP + SSAP + Data units

In the telegram specified in table 5.16, 6 data units are transmitted to the slave to parameterize the device. Each data unit represents a specific parameter that can be set. The specific values of the six data units are discussed in detail in appendix G.

A slave will respond to this parameterization telegram with a short acknowledge telegram consisting of one data character with value ‘E5H’. This will inform the master that the parameterization has taken place on the slave device. After this response, the master will continue to the next configuration telegram.

Configuration

After a slave has been parameterized by a master, the master will specify the configuration with the telegram displayed in table 5.17. This telegram specifies the amount of input and output data that will be exchanged between the slave and master in the data exchange state. Only one data byte is transmitted in this telegram. This binary value of the byte is displayed in appendix G.

Table 5.17 – Configuration telegram frame

SD2	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	00000110	00000110	68H	10000001	10000000	01101101	00111110	00111110	1X	01100001	16H

- LE and LEr = decimal value of 6. (DA + SA + FC + DSAP + SSAP + DU)
- FC: Specifies the function of the telegram as – “Send and request data (high priority)”
- DSAP: Check configuration data
- SSAP: Check configuration data
- FCS = DA + SA + FC + DSAP + SSAP + DU

The data unit is used to specify the amount of I/O data that will be exchanged in the data exchange state. The designed value of this byte specifies that 16 bytes of data will be exchanged between the master and slave during each data exchange cycle. The slave will respond to this configuration telegram with a short acknowledge telegram of one data character with value ‘E5H’. This will inform the master that the configuration has taken place on the slave device.

Data exchange

Once the parameterization and configuration have completed successfully, the request diagnostics state will once again be executed to verify if the configuration and parameterization have completed successfully. Figure 5.12, which displays the state machine, illustrates this behaviour. When the parameterization and configuration are complete, the data exchange state will be executed. Table 5.18 displays the telegram used in this design for data exchange. The 16 data units will differ for each data exchange cycle, and therefore the frame check sequence byte will be calculated before each transmission, based on the values of the 16 data units.

Table 5.18 – Data exchange telegram frame

SD2	LE	LEr	SD	DA	SA	FC	DU	FCS	ED
68H	00010011	00010011	68H	00000001	00000000	01111101	16X	x	16H

- LE and LEr = decimal value of 19. (DA + SA + FC + (16 x DU))
- FC: Specifies the function of the telegram as – “Send and request data (high priority)”
- FCS = DA + SA + FC + DSAP + SSAP + Data units

The PROFIBUS network is established to transmit data between the PROFIBUS master (FPGA) and the slave (PLC). During configuration, 16 bytes of I/O data are specified to be transmitted in each data exchange cycle. These data bytes are used to communicate specific control data

between the PLC and FPGA. Table 5.19 lists the 16 bytes of data that will be sent to the PLC from the FPGA, while table 5.20 lists the 16 bytes of data that will be received by the FPGA from the PLC. These data values will be discussed in detail in the next section.

Table 5.19 – Transmitting data from FPGA to the PLC while RDS is delevitated

Data position	**Indication**
1	Levitated indication
2	Open
3	Open
4	Open
5	Open
6	Delevitated indication
7 - 16	Open

Table 5.20 – Transmitting data to FPGA from PLC

Data position	**Indication**
1	Levitated indication
2	Open
3	GUI control indication
4	PLC control indication
5	Open
6	Delevitated indication
7 - 16	Open

The PROFIBUS network will continue to operate in the data exchange state unless an error occurs. If an error occurs at the master or receiver side, the appropriate indication will be sent to the other device. In the event of an error, the request diagnostic state will be executed to determine if the network is still operational or whether the parameterization and configuration must be executed.

The next section will discuss how the UART is designed. All the procedures displayed in table 5.10 will be used to design the universal receiver and transmitter.

UART design

A universal asynchronous receiver/transmitter (UART) is usually an integrated circuit, which plays the most important role in serial communication. The UART is mainly responsible for the conversion between serial and parallel data and transmitting and receiving the data over the serial channel. Serial communication allows for long distance communication between systems because of the low signal distortion in serial communication [42] [43].

In chapter 2, the application of VHDL programming was discussed. This language allows the programmer to develop firmware which replicates hardware to be instantiated on a FPGA. This process is followed to implement a UART receiver and transmitter. Figure 5.13 displays the block diagram of the designed UART. The functions of both the receiver and transmitter have already been discussed. In the next section, the timing and synchronization of the entire system together with the UART timing will be discussed.

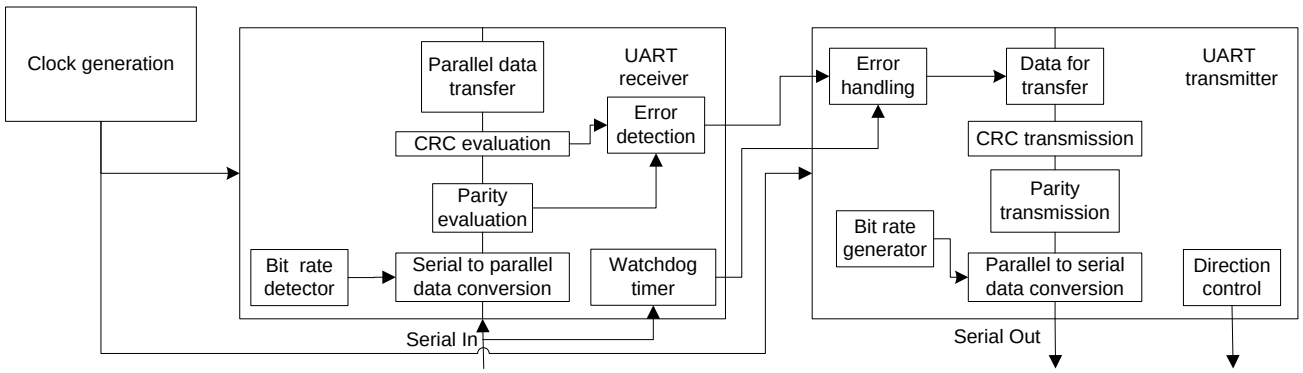


Figure 5.13 – UART design

5.2.6 Timing and synchronization

In chapter 3, the timing parameters of the PROFIBUS standard were discussed. Unlike other Fieldbus systems, which are event-driver busses, PROFIBUS was designed to guarantee a deterministic response. The determinism of a system refers to the ability of the system to precisely predict the behaviour of the system over time. This allows the calculation of a reliable system reaction time of a PROFIBUS network [16].

5.2.6.1 Timing parameter calculations

Before the synchronization of the designed PROFIBUS protocol can be discussed, it is necessary to calculate specific timing parameters used in this design. These parameters were discussed in chapter 3 and the necessary parameters for the PROFIBUS protocol design is presented here. The bit rate for communication was chosen at 1.5 Mbps, with a clock signal of 66 MHz, which is generated in synchronization with the min FPGA clock signal. All the parameters are derived by using the equations specified in the *PROFIBUS specification* [26].

Bit time $(\boldsymbol{Tbit})$

The bit time is the time needed to transmit one bit of data. Equation (5.2) presents the calculation for the bit time. Thus it can be shown that a bit time of 666.67 ns will be implemented.

$$\begin{aligned} Tbit &= \frac{1}{bit\ rate} \\ &= \frac{1}{1.5\ \text{Mbps}} \\ &= 666.67\ ns \end{aligned} \tag{5.2}$$

Synchronization time $(\boldsymbol{T_{SYN}})$

The synchronization time refers to the minimum time a station must remain in the idle state before a new request can be accepted. PROFIBUS specifies this time to be $33 \times Tbit$ which results in a synchronization time of 22 µs.

Master idle time $(\boldsymbol{T_{ID1}})$

The idle time of the master refers to the time between the transmission of the last bit of data of a telegram and the transmission of the first byte of a new telegram. The master must wait this amount of time before transmitting a new telegram. Equation (5.3) presents the calculation and a 22.67 µs result for the master idle time, where T_{SM} is a safety margin specified by the designer. For this design this value is determined to be 10 *Tbit*.

$$\begin{aligned} T_{ID1} &= T_{SYN} + T_{SM} \\ &= 33\ Tbit + 10\ Tbit \\ &= 28.67\ \mu s \end{aligned} \tag{5.3}$$

Slave response time (T_{SDR})

The slave response time simply refers to the response time of a slave to respond to a message. This value is set in the parameterization state and is designed to use the default value of 11 *Tbits*. In large PROFIBUS networks, this time will be adapted to the specific network, but is not vital in this design.

By using these parameters, the synchronization of the developed PROFIBUS network can be established. In the following sections, the UART timing, direction control and PROFIBUS network synchronization will be discussed.

5.2.6.2 Serial transmission timing design

As mentioned, the bit rate for data communication was specified as 1.5 Mbps with a clock signal of 66 MHz. Data is transmitted as non-return to zero data units in the PROFIBUS protocol. This implies that in order to transmit a binary '1' over the serial channel, the voltage level of the channel must remain high for 1 *Tbit*. A *Tbit* was specified to be 666.67 ns. The 66 MHz clock signal produces a clock signal every 15.15 ns. This results in the serial channel transmitting a binary '1' for 44 clock cycles. An example of this is shown in figure 5.14.

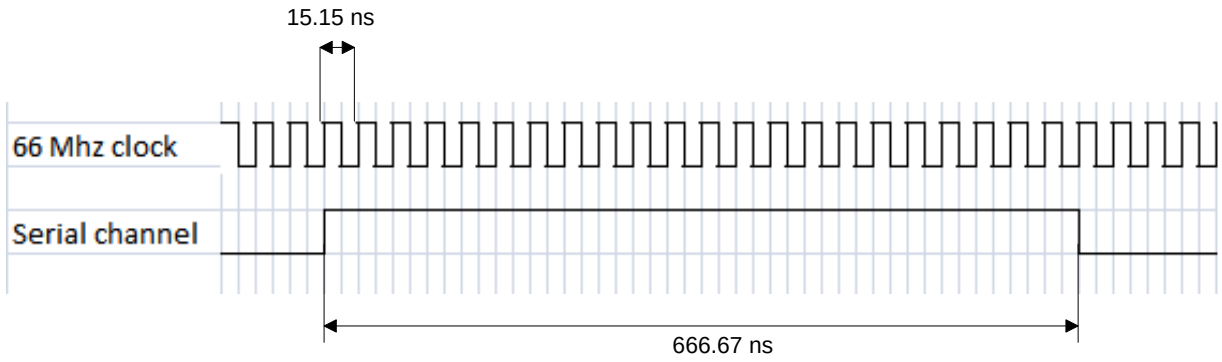


Figure 5.14 – Serial transmission timing

5.2.6.3 Serial receiving timing design

Upon reception of serial data, the receiver designed for the FPGA, will sample the incoming bits in the middle of the bit as shown in figure 5.15. This is done to ensure that the receiver will be able to distinguish between a low and high voltage level of the signal, should noise or jitter corrupt the signal. The sampling point is designed at 22 clock cycles, which results in 333.37 ns (precisely in the middle of the receiving bit).

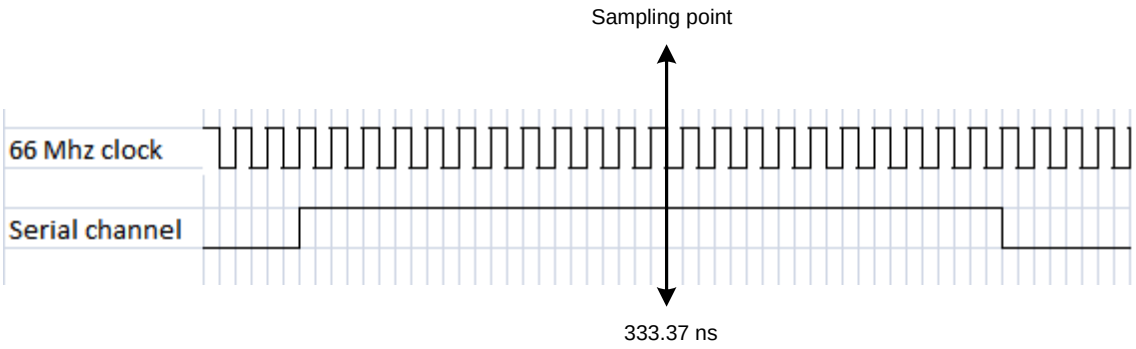


Figure 5.15 – Serial receiving timing

5.2.6.4 Channel direction control design

Previously in this chapter the RS 485 driver design was discussed. This driver uses a direction control signal to change the PROFIBUS channel between transmit and receive. In figure 5.16 this signal is illustrated together with the serial channel. A logic '1' on the direction control signal indicates transmission from the master, while a logic '0' indicates the slave device is responding over the serial channel. In this figure, the master initiator time (T_{ID1}) as well as the slave reaction time (T_{SDR}) is displayed on the serial channel.

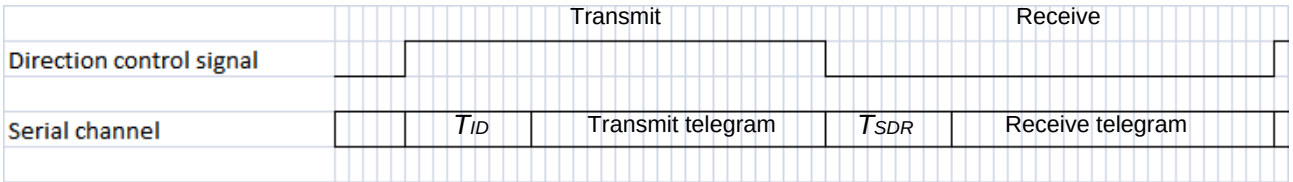


Figure 5.16 – Direction control timing

5.2.6.5 PROFIBUS protocol timing and synchronization design

By combining all the timing considerations of the developed PROFIBUS protocol, a synchronization diagram can be constructed to illustrate the internal PROFIBUS signal synchronization. Figure 5.17 is used to illustrate the main signal synchronization of the developed PROFIBUS protocol. The signals of this figure together with their corresponding function are presented in table 5.21.

Table 5.21 – PROFIBUS main internal signals

Signal	**Function**
Tx ready	Indicates to the UART to transmit data
Parallel data out	Data sent from the main controller to the UART for transmission
Parallel data received	Serial to parallel converted data received from the slave PLC
Rx ready	Indicates to the UART to receive data
Direction control	Indicates the direction of the serial channel to the RS 485 driver
Serial signal	Used to transmit and receive data to and from the slave PLC
Parity error	Indicates a parity error, on the receiving data, to the UART
CRC error	Indicates a CRC error, on the receiving data, to the UART

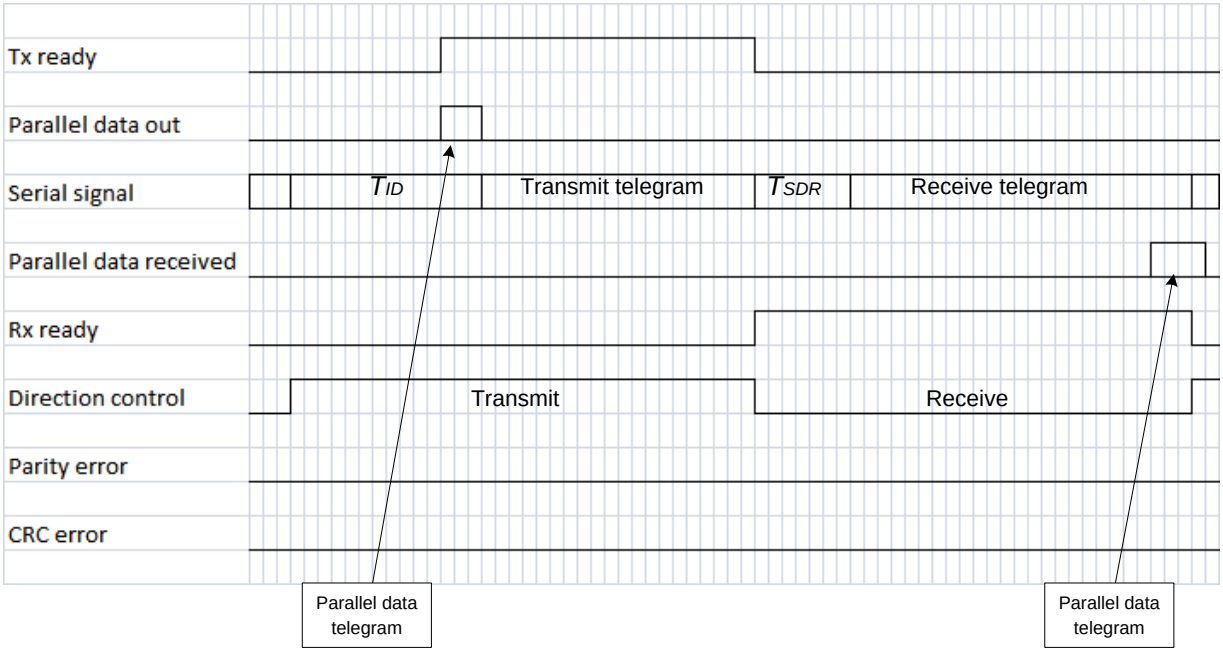


Figure 5.17 – PROFIBUS protocol timing and synchronization

5.2.7 PLC slave

The Siemens® S7-200 PLC and the PROFIBUS EM 277 module were discussed in detail in chapter 4. The PROFIBUS EM 277 module is a registered PROFIBUS slave device and fully conforms to the PROFIBUS standard. Besides the PROFIBUS communication, the PLC device is used in the ADES for various functions that include motor control, system monitoring, system temperature monitoring etc.

The PROFIBUS EM 277 module is a slave device and can only communicate to a master if the master requests data from the slave. The GSD file of the EM 277 together with the programming of the PLC is discussed in appendix C. In this section the buttons connected to the PLC and the corresponding data that will be transmitted by the PLC over the PROFIBUS network will briefly be discussed.

In figure 5.18 the PLC is seen with two buttons and a key switch connected to the device. As discussed previously, these buttons and switch are used to manually operate the rotor delevitating system (RDS). Figure 5.18 also illustrates the corresponding data units that will be transmitted over the PROFIBUS network, should the buttons be pressed or the key turned.

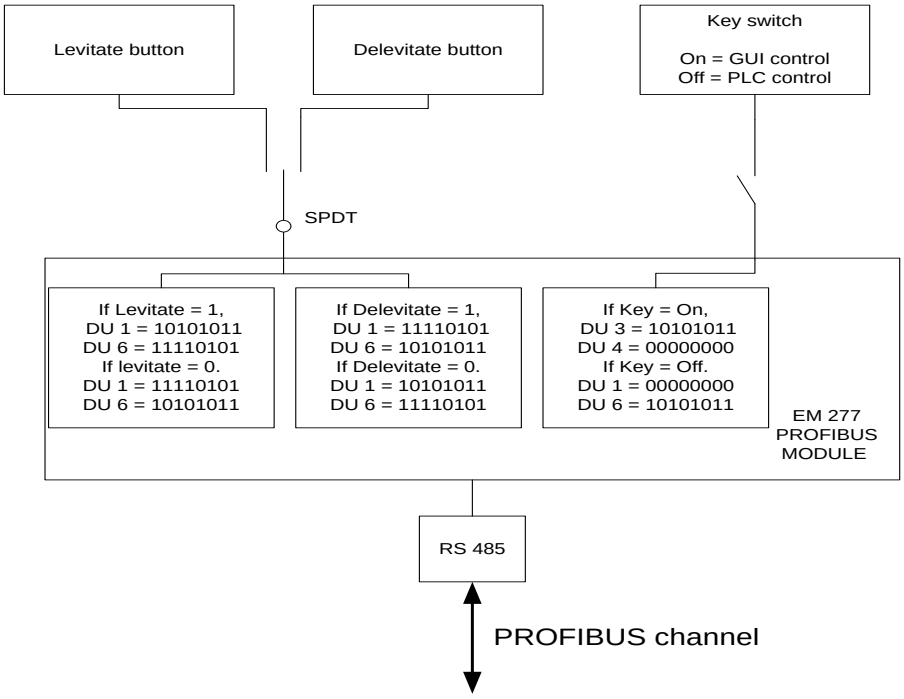


Figure 5.18 – PLC user interface

5.2.8 Final ADES implementation

All the aspects of the PROFIBUS protocol design and implementation on the ADES have been discussed in previous sections. By using this information the final implementation can be illustrated in figure 5.19. The communication flow of the developed PROFIBUS protocol with regards to the ADES implementation is seen in this figure.

In this figure, it is clearly displayed how data is transmitted from the GUI interface, or PID control unit through the PROFIBUS protocol instantiation and transmitted to the PLC over the PROFIBUS network. Data can also be sent from the PLC, through the PROFIBUS protocol instantiation to the GUI or PIC control of the rotor delevitating device (RDS).

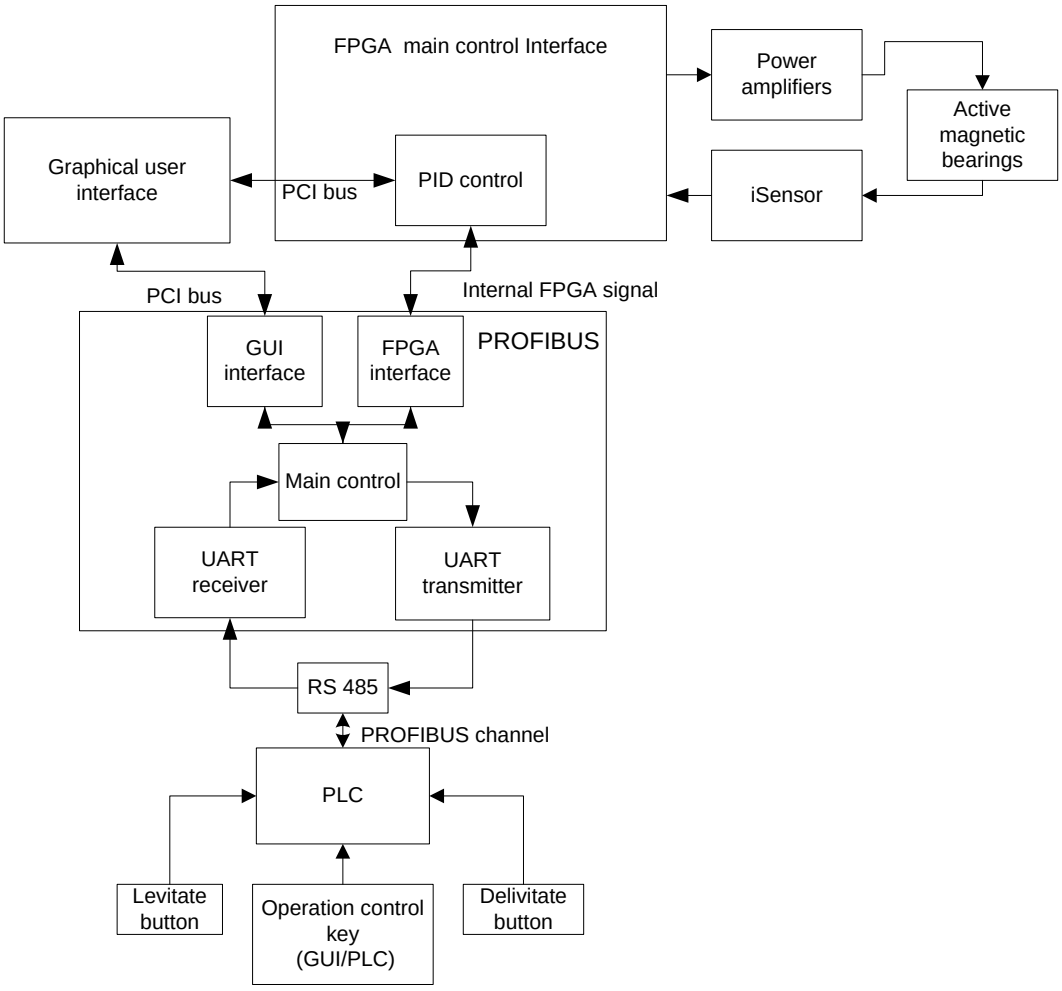


Figure 5.19 – Internal communications data flow

5.3 Conclusion

This chapter started by defining the requirements for the PROFIBUS protocol as well as the ADES implementation. Based on these requirements the PROFIBUS protocol was developed in VHDL and implemented on the Xilinx® FPGA. The design of the ADES requirements for the PROFIBUS network was also discussed together with the implementation. The VHDL code can be found in appendix B. In the next chapter, the developed PROFIBUS protocol will be verified, validated and compared to commercial PROFIBUS devices to test the feasibility of this design.

Chapter 6

Verification of the developed protocol

The previous chapter discussed the complete design and implementation of the PROFIBUS protocol. This chapter focuses on the verification of the designed protocol. Each aspect of the PROFIBUS DP protocol is evaluated in the simulation environment. The verification tests performed in this chapter is designed to ensure that the specifications of the PROFIBUS DP standard are met in the design. The validation of the implemented protocol will be done in chapter 7. The introduction to this chapter will distinguish between verification and validation as applied to this study.

6.1 Introduction

Every engineering process requires the intelligent use of verification and validation. Besides determining the accuracy of the process, verification and validation ensures the quality of the process [44]. Verification and validation is the use of proven engineering techniques which aim to stop defects in creation and detect problem areas or defects as soon as possible in product development to reduce development time and costs and at the same time ensuring that the resulting product is safe and effective [45]. A wide variety of definitions exist for verification and validation. The following definitions originate from a system engineering approach and relates closely to the developed system discussed in this document and will be used as the starting point for these procedures.

“Verification means confirmation by examination and provision of objective evidence that specified requirements related to a product or process have been met.” [45]

“Validation means establishing and documenting evidence which provides a high degree of assurance that a process will consistently produce a result or product meeting its predetermined specifications.” [45]

By using these definitions, the following can be achieved through verification and validation [44]:

- Comprehensive analysis and tests to determine if a developed process functions correctly and performs no unintended functions

- Measuring quality and reliability
- Evaluating developed processes in a systems context by evaluating them against system requirements
- Ensuring that the pre-determined specifications and requirements are met and to what extent

By applying these principles of verification and validation to the developed PROFIBUS protocol, the following questions can be answered:

- Does the developed protocol perform correctly in the simulation as well as in the application environment?
- Have the developed protocol met the predetermined requirements and end specifications?
- What is the quality and reliability of the developed PROFIBUS protocol?

These questions together with the definitions for verification will be used throughout this chapter to evaluate specific areas regarding the developed PROFIBUS protocol. The verification and validation will be done separately by evaluating each of the applicable OSI reference model layer individually.

6.2 Test and verification methods

As mentioned earlier, verification means the confirmation by examination and provision of objective evidence that specified requirements related to a product or process have been me. Examination is done in the form of developed tests to measure data and evaluate and determine the results. The physical layer and data link layer of the OSI reference model are evaluated in order to verify the design. The following tests regarding the applicable layer are performed:

Physical layer evaluation methods

- Physical layer hardware design evaluation

Data link layer evaluation methods

- Evaluation of the following aspects of the PROFIBUS DP protocol in a simulation environment:
  - Character frame
  - Telegram frame

- PROFIBUS state machine
- UART transmitter
- UART receiver
- PROFIBUS main controller
- Parity and cyclic redundancy check evaluation
- Timing and synchronization

6.3 Physical layer design verification

The physical layer is not verified through simulation. This section is only used to verify the correct design of the physical layer according to the PROFIBUS DP standard. The developed protocol is implemented on a Xilinx® FPGA. As discussed in chapter 5, the serial data as well as the channel directional control data are sent via the FPGA to three front I/O digital signals connected to the FPGA. These signals are connected to the RS 485 differential drivers that convert serial signals to differential signals. The data is then sent via the PROFIBUS Type A cable that is specified for PROFIBUS transmission. PROFIBUS Type A cable is a specifically manufactured, shielded, twisted pair cable. Figure 6.1 gives an overview of the physical layer. Termination resistors are situated on each side of the differential channels as specified by the PROFIBUS standard.

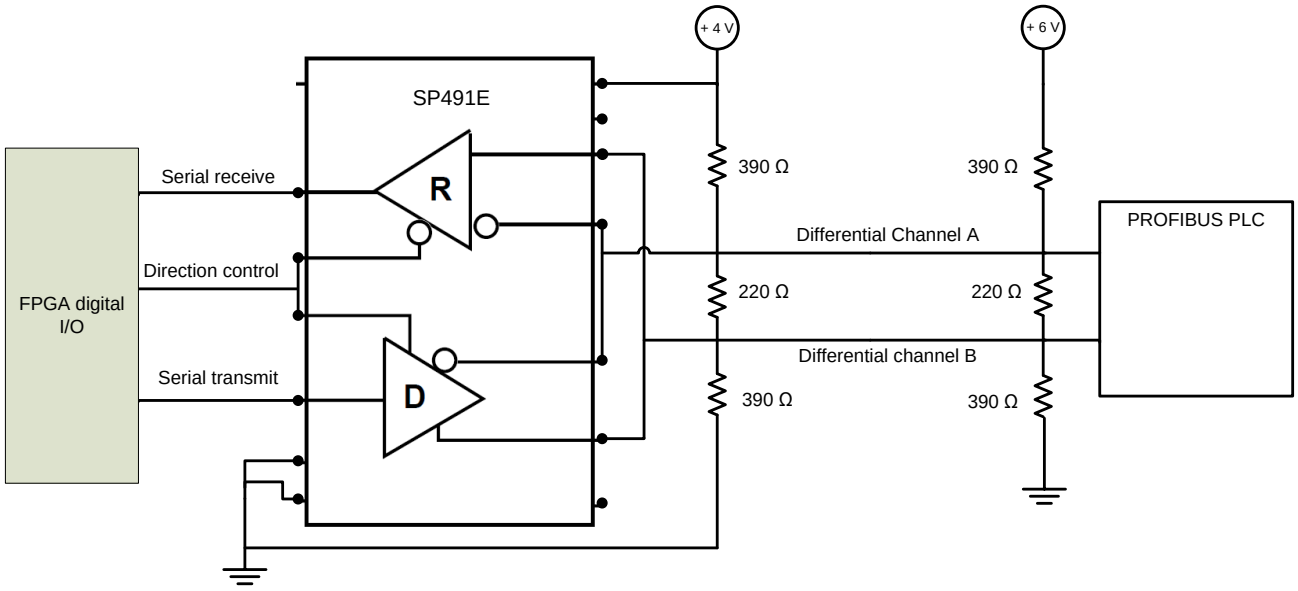


Figure 6.1 – Physical layer overview

Table 6.1 – Physical layer hardware design specifications

Physical layer implementation	**Hardware design specifications**
Termination resistors	390 Ω and 220 Ω, as specified by PROFIBUS
RS 485	Implemented with SP491E half-duplex RS 485 differential drivers
Cable	Standard PROFIBUS Type A cable

6.4 Data link layer verification

Although the performance evaluation of the physical layer has great significance, it does not ensure that the PROFIBUS protocol performs according to the required specifications. The verification of the higher data link layer addresses this. By verifying this layer, the design is tested to simulate the environment in which the design has to operate. This ensures quality design and verifying that the specifications and requirements are implemented correctly into the design.

As mentioned in chapter 2, the verification of the data link layer will be done with the simulation program Modelsim®. A test bench that simulates the real-time environment is created in Modelsim® to produce specific stimuli for the designed PROFIBUS protocol. The response of the designed PROFIBUS protocol is evaluated and measured against the requirements and specifications. Table 6.2 lists the various aspects of the PROFIBUS protocol design that will be evaluated together with the requirements that need to be verified.

Table 6.2 A – Data link layer verification

PROFIBUS subdivisions	**Requirements**
Character frame	Non-return to zero bit coding 1 Low start bit, 1 high stop bit 8 Data bits Least significant bit to most significant bit transmission 1 Even parity bit
Telegram frame	**Telegram with no data field** Start delimiter Destination address

Table 6.2 B – Data link layer verification continued

Telegram frame (continued)	Source address Function code Frame check sequence End delimiter **Telegram with variable data length** Start delimiter Length Length repeated Start delimiter repeated Destination address Source address Function code Destination service access point Source service access point Data units Frame check sequence End delimiter
PROFIBUS state machine	Request diagnostics 1 Parameterization Configuration Request diagnostics 2 Data exchange
UART transmitter	Parallel to serial data conversion and transmission
UART receiver	Serial to parallel data conversion and transmission
Parity and cyclic redundancy check evaluation	Parity detection and error flagging Cyclic redundancy checks with error correction
Timing and synchronization	Clock generation Bit rate Serial to parallel conversion PROFIBUS timing specifications Transmit and receive driver direction control

Each of the PROFIBUS specifications and requirements are analyzed by evaluating the waveforms produced through simulations in Modelsim®. In each simulation the applicable waveforms are displayed and discussed with regards to the PROFIBUS standard.

6.4.1 Character frame verification

Table 6.3 depicts the fundamental specifications of the PROFIBUS DP character frame as discussed in chapter 3. Each data byte that is transmitted over a PROFIBUS network has to conform to these specifications, including the telegram headers. It should be noted that when no data is transmitted, the idle state potential of the line is '1'. A start bit will cause the line to go to '0'. In each transmission the UART transmitter transmits the 8 bits of information from least significant bit to most significant bit and adds a parity bit [16].

Table 6.3 – PROFIBUS DP character frame specifications

Non-return to zero bit coding
1 Low start bit, 1 high stop bit
8 Data bits
Least significant bit to most significant bit transmission
1 Even parity bit

Figure 6.2 displays a single character transmitted over a serial data line to the receiver. The observations relating to the transmitted character is depicted in table 6.4.

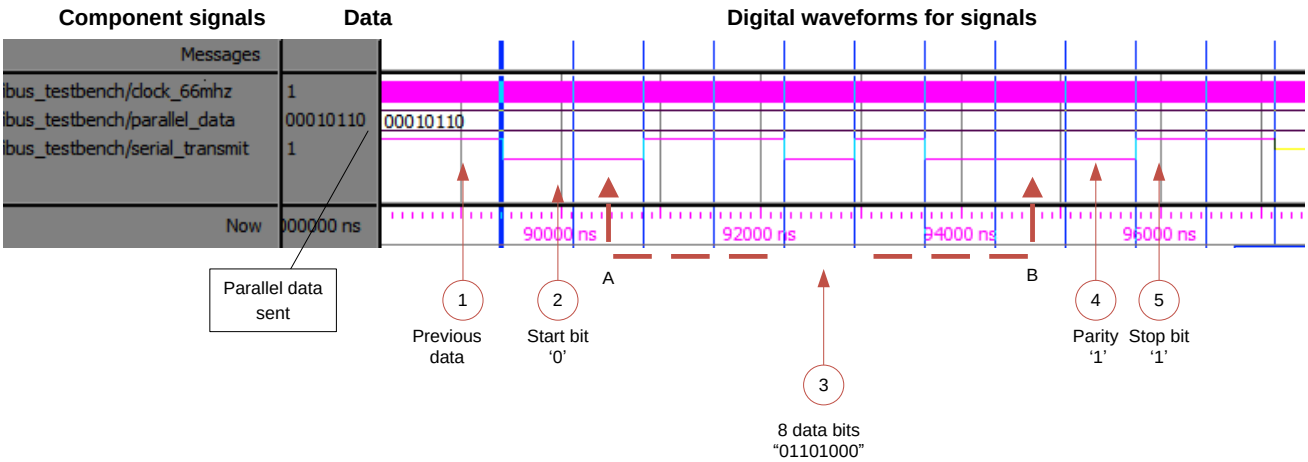


Figure 6.2 – PROFIBUS DP character simulation

Table 6.4 – Observations concerning PROFIBUS DP character

Simulation marker	**Observation**
General	The test bench simulates the signals as specified in the design. Column 1 in figure 6.2 displays the component signals. Column 2 displays the value of the signal at the highlighted cursor in the waveform. Column 3 gives a representation of the data value of the signals at a specific time. Time is in nanoseconds. The parallel data in signal 2 is transmitted as serial data in signal 3. The parallel data is not transmitted over a data line but is internally transmitted from the PROFIBUS main controller to the UART transmitter.
1	Marker 1 points to the last bit of data of the previous character that was transmitted. As shown, this data is logic ‘1’ to indicate the end of the previous character.
2	Marker 2 points to the start bit (logic ‘0’) of the character being sent to indicate a new character
3	Marker 3 points to the 8 data bits being transmitted. The value as transmitted in time is ‘01101000’. This data is transmitted form least significant bit to most significant bit, implying that this data will be received as ‘00010110’, the same data as shown in signal 2 by the parallel data being transmitted.
4	Marker 4 points to the logic ‘1’ parity bit of the character. Parity bits will be discussed in detail later in this section
5	Marker 5 points to the stop bit (logic ‘0’) of the character, indicating the end of the character.

6.4.2 Telegram frame verification

A PROFIBUS telegram can contain up to 43 bytes, with 32 bytes of data and 11 bytes of overhead per message. In the data exchange state, data transfers of up to 244 bytes can be transmitted with the use of more than one telegram message. The overhead is referred to as the telegram header. PROFIBUS specifies five types of telegrams that can be used in devices but not all are mandatory [16]. Table 6.5 highlights the requirements of the two types of telegrams used in the protocol design.

Table 6.5 – PROFIBUS DP telegram frame specifications

Telegram with no data field	**Specification**
Start delimiter	10H (Hexadecimal) value
Destination address	8 bit address of slave/master
Source address	8 bit address of master/slave
Function code	Identification of the type of telegram
Frame check sequence	Cyclic redundancy check frame
End delimiter	16H (Hexadecimal) value
Telegram with variable data length	**Specification**
Start delimiter	68H (Hexadecimal) value
Length	Nett data length
Length repeated	Nett data length
Start delimiter repeated	68H
Destination address	8 bit address of slave/master
Source address	8 bit address of master/slave
Function code	Identification of the type of telegram
Destination service access point	Service access point for receiver to determine the next service
Source service access point	Service access point for transmitter to determine the next service
Data units	1 to 244 bytes
Frame check sequence	Cyclic redundancy check frame
End delimiter	16H (Hexadecimal) value

In figure 6.3 and figure 6.4 the two telegrams specified in table 6.6 are displayed as simulated. It should be noted that each segment in the two figures displays a PROFIBUS character as displayed in the previous section. Table 6.6 and 6.7 discusses figure 6.3 and 6.4 respectively. Both figures represent a clock signal, parallel data transmitted and the resulting serial transmit signal.

Although the detail of the figure is limited by the amount of space available to display, the necessary observations can still be made.

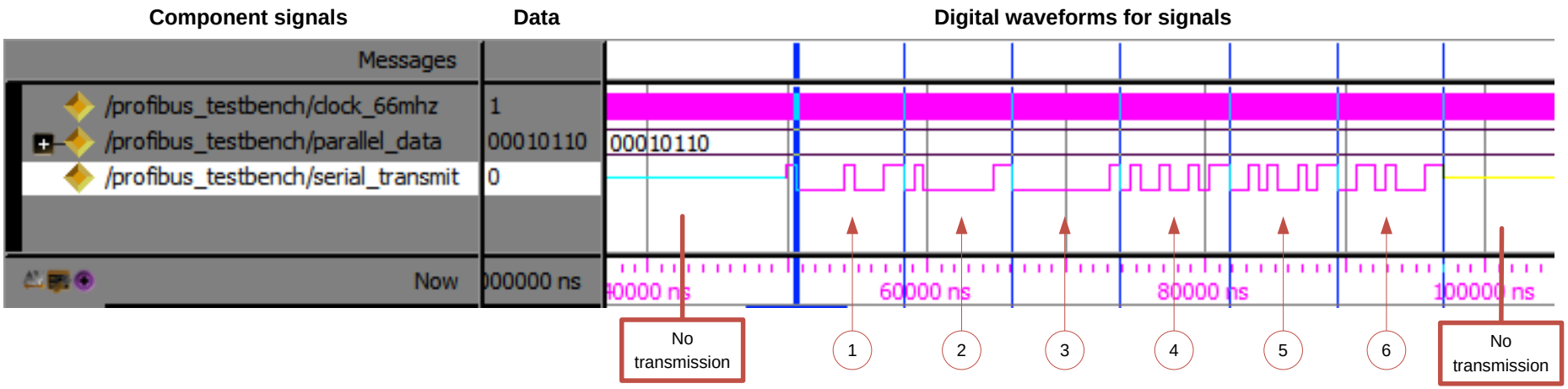


Figure 6.3 – PROFIBUS DP telegram A simulation (no data field)

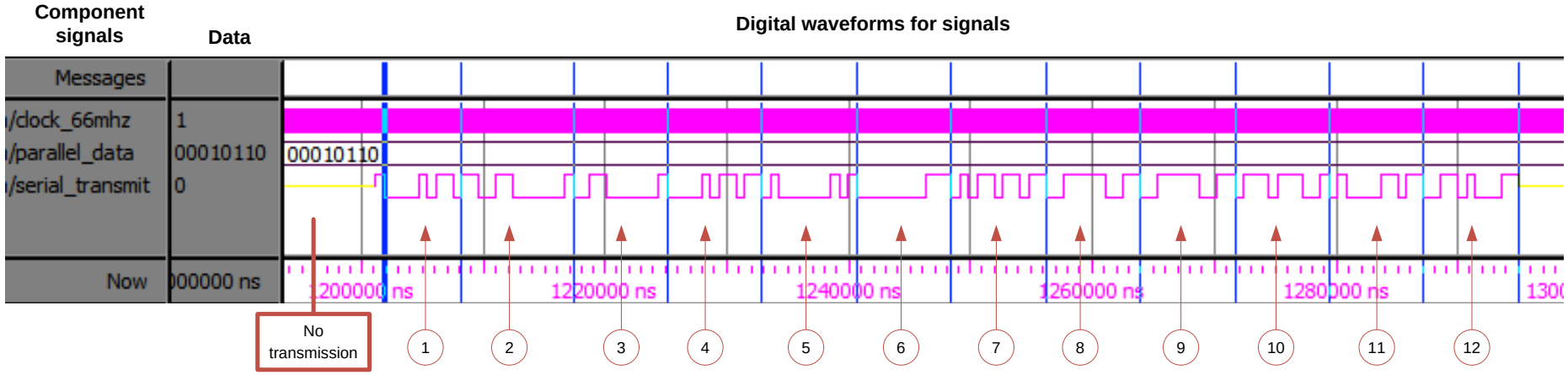


Figure 6.4 – PROFIBUS DP telegram B simulation (with data field)

Table 6.6 – Observations regarding PROFIBUS DP telegram frame 1

Simulation marker	**Observation**
General	The first “no transmission” label refers to no data being transmitted over the serial signal. The second “no transmission” label indicates a change in direction of the serial line. This is discussed in the synchronization verification
1	Marker 1 points to the character ‘0**00001000**11’, indicating the start delimiter value ‘00010000’ or ‘10H’
2	Marker 2 points to the character ‘0**10000000**11’, which can be translated as the address, ‘00000001’,of the receiver (slave)
3	Marker 3 points to the character ‘0**00000000**01’, which can be translated as the address, ‘00000000’ of the transmitter (master)
4	Marker 4 shows the function code character’ 0**10010010**11’. This specific value ‘01001001’ is a function code that specifies a request to the receiver to return the status of the receiver.
5	Marker 5 points to the frame check character ‘0**01010010**11’ with data units, ‘01001010’. This value is calculated by applying a cyclic redundancy check (CRC) value to the frame. CRCs are discussed in detail later in this section.
6	Marker 6 points to the character ‘0**01101000**11’ with data units, ‘00010110’ or ‘16H’, that indicates the end of the telegram.

Table 6.7 A – Observations regarding PROFIBUS DP telegram frame 2

Simulation marker	**Observation**
1	Marker 1 points to the character ‘0**00010110**11’, indicating the start delimiter value ‘01101000’ or ‘68H’
2	Marker 2 points to the value ‘0**11000000**01’, with data units, ‘00000110’. Converting this value to a decimal value, produces a value of 6. This is calculated by counting the amount of characters in the telegram (excluding the start delimiters, length, and end delimiter characters). The receiver uses this information to check the amount of characters in the telegram frame for error detection.
3	Marker 3 points to the same value as marker 2 which indicates the length is repeated for redundancy as specified by PROFIBUS.

Table 6.7 B – Observations regarding PROFIBUS DP telegram frame 2

4	Marker 4 again points to the character '0**00010110**11, indicating the start delimiter value '01101000' or '68H'
5	Marker 5 points to the character '0**10000001**01', which can be translated as the address, '10000001', of the receiver (slave) and indicating that a SSAP character is included in the telegram.
6	Marker 6 points to the character '0**00000001**11', which can be translated as the address, '10000000' of the transmitter (master) and indicating that a DSAP character is included in the telegram.
7	Marker 7 points to the character '0**10110110**11'. This data unit, '01101101', is a function code that requests data from the receiver.
8	Marker 8 points to the character '0**01111100**11'. This data unit, '00111110' or 62 in decimal format, indicates a destination service access point requesting the receiver to check the configuration data
9	Marker 9 points to the character '0**01111100**11'. This data unit, '00111110' or 62 in decimal format, is a source service access point indicating that the transmitter is performing a configuration check.
10	Marker 10 shows the function code character' 0**11101110**11'. This specific value '01110111' is a data unit used to specify the amount of input and output bytes that will be transmitted in the data exchange state.
11	Marker 11 points to the frame check character '0**10000110**11' with data units, '01100001'. This value is calculated by applying a cyclic redundancy check (CRC) value to the frame. CRCs are discussed in detail later in this section.
12	Marker 12 points to the character '0**01101000**11' with data units, '00010110' or '16H', that indicates the end of the telegram.

6.4.3 PROFIBUS state machine verification

When a PROFIBUS network is established, a slave device has to be assigned to a master device. This allows multiple masters and slaves to use a common bus to transmit and receive data. During this assignment, the master will establish communication with a specified slave by specifying parameters and configuring the device according to the intended network use allowed by the slave

[16]. Although only one master and slave was used in this protocol design, the full PROFIBUS standard with regards to medium access control is implemented and can easily be extended to add more slave devices. Table 6.8 lists the state machine requirements for the PROFIBUS DP standard.

Table 6.8 – PROFIBUS DP state machine requirements

PROFIBUS state	**Requirements**
Request diagnostics 1	Read slave parameterization and configuration data
Parameterization	Parameterize the slave according to the master and specifies the operating mode of the slave
Configuration	Configure the slave according to the specifications of the slave and the PROFIBUS network
Request diagnostics 2	Read slave parameterization and configuration data after configuration and parameterization
Data exchange	Cyclic transfer of data according to configuration and parameterization

Figure 6.5 shows the simulation result, indicating the execution of all the states of the PROFIBUS protocol. Table 6.9 discusses figure 6.5 and the relevant observations in detail.

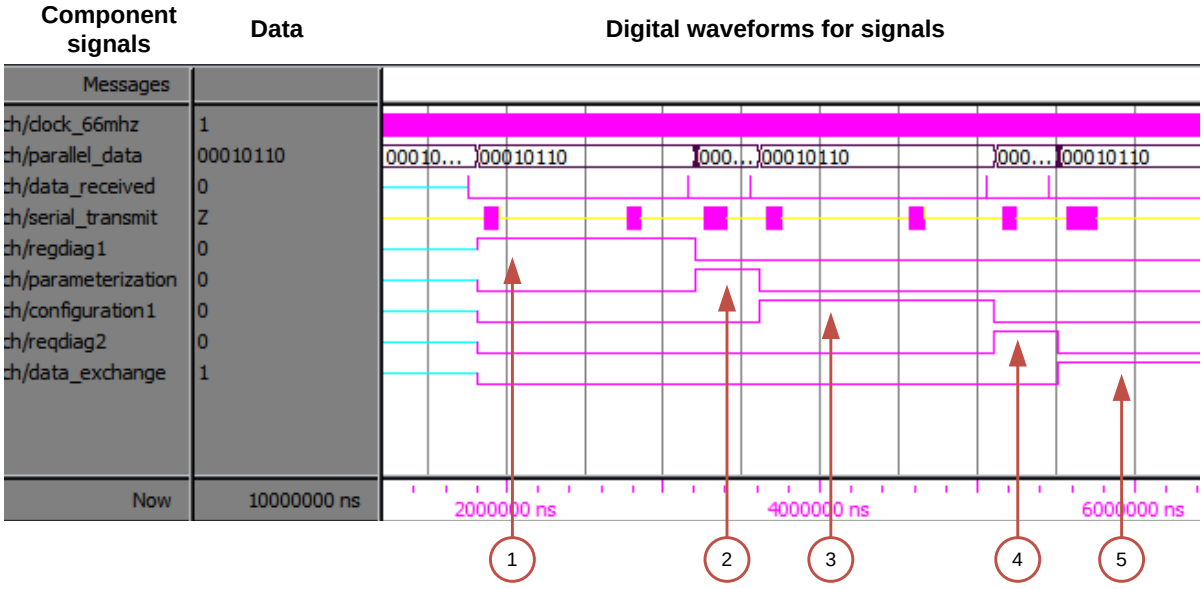


Figure 6.5 – PROFIBUS DP state diagram simulation

Table 6.9 – Observations regarding the PROFIBUS DP state machine

Simulation marker	**Observation**
General	Signal '*parallel_data*' is an internal signal with a much higher bit-rate than the serial signal. Therefore the data being transferred in the parallel signal cannot be visually evaluated in this figure, but only represents new data being sent to the UART transmitter at the beginning of each new state. A new state will only occur once correct data has been received back from the receiver. The data needs to pass parity and CRC evaluation and indicate to the transmitter (master) whether the next state can occur.
1	Marker 1 points to a simulation signal indicating the start of the first state (with logic '1'), e.g. request diagnostics 1. The corresponding serial signal transmits the applicable telegram. To display all the states in this figure, the data being transmitted over the serial signal is seen as a 'block' of data being transmitted. In the request diagnostics state shown here, the data is transmitted twice. A new state will only occur once correct data have been received after a transmission. The signal *'data_received'* indicates when data is received. In this simulation the receiver did not respond quickly enough, which triggered the data to be resent. This timing specifications of the PROFIBUS protocol is discussed more in detail later in this section
2	Marker 2 points to the indication signal, which displays (with logic '1') that the parameterization state has started. It can be noted that the *'request_diagnostics'* signal is '0', indicating that the first state ends when the new state begins. This applies to all the states of the PROFIBUS protocol.
3	Marker 3 points to the start of the configuration state and the end of the previous parameterization state.
4	Marker 4 points to the start of the second request diagnostics state and the end of the previous configuration state.
5	Marker 5 points to the start of the data exchange state and the end of the previous request diagnostics state.

6.4.4 UART transmitter verification

As discussed in chapter 5, the UART transmitter and receiver were implemented as two separate units in the VHDL design. The main functions of the UART transmitter are to receive parallel data from the PROFIBUS main control, convert the data to serial data and transmit it over the serial signal. The UART transmitter also transfers control and error signals to and from the UART receiver. Figure 6.6 indicates how the UART receiver fits into the design. The simulation results are shown in Figure 6.7 and the relating observations summarized in table 6.10.

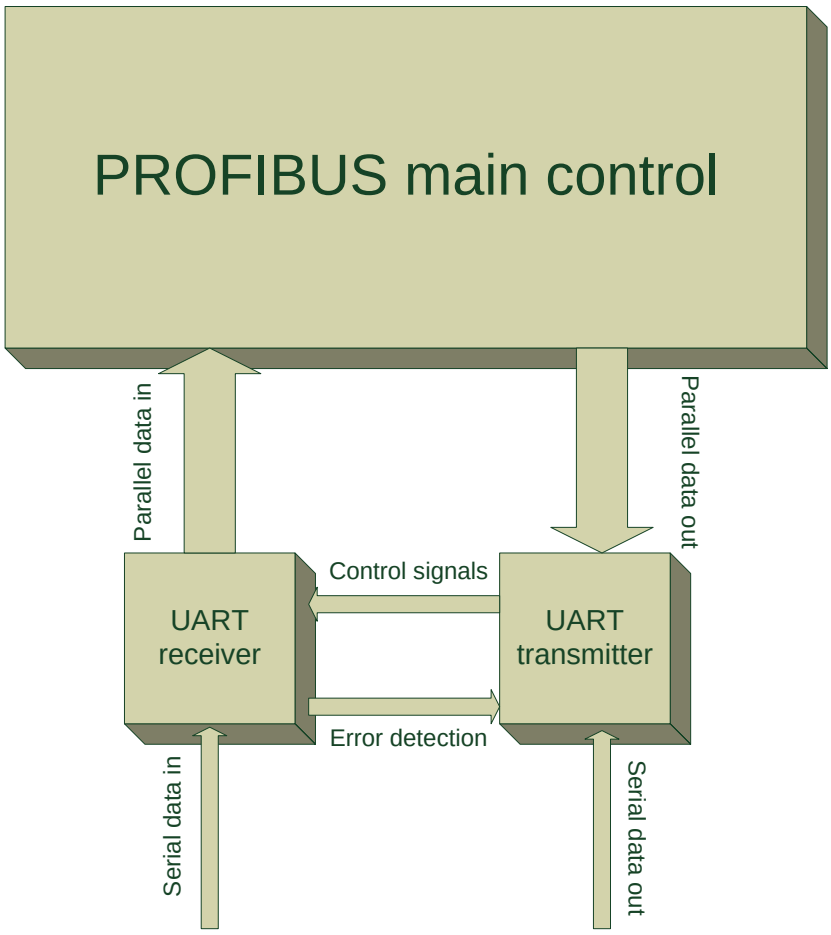


Figure 6.6 – PROFIBUS data flow design

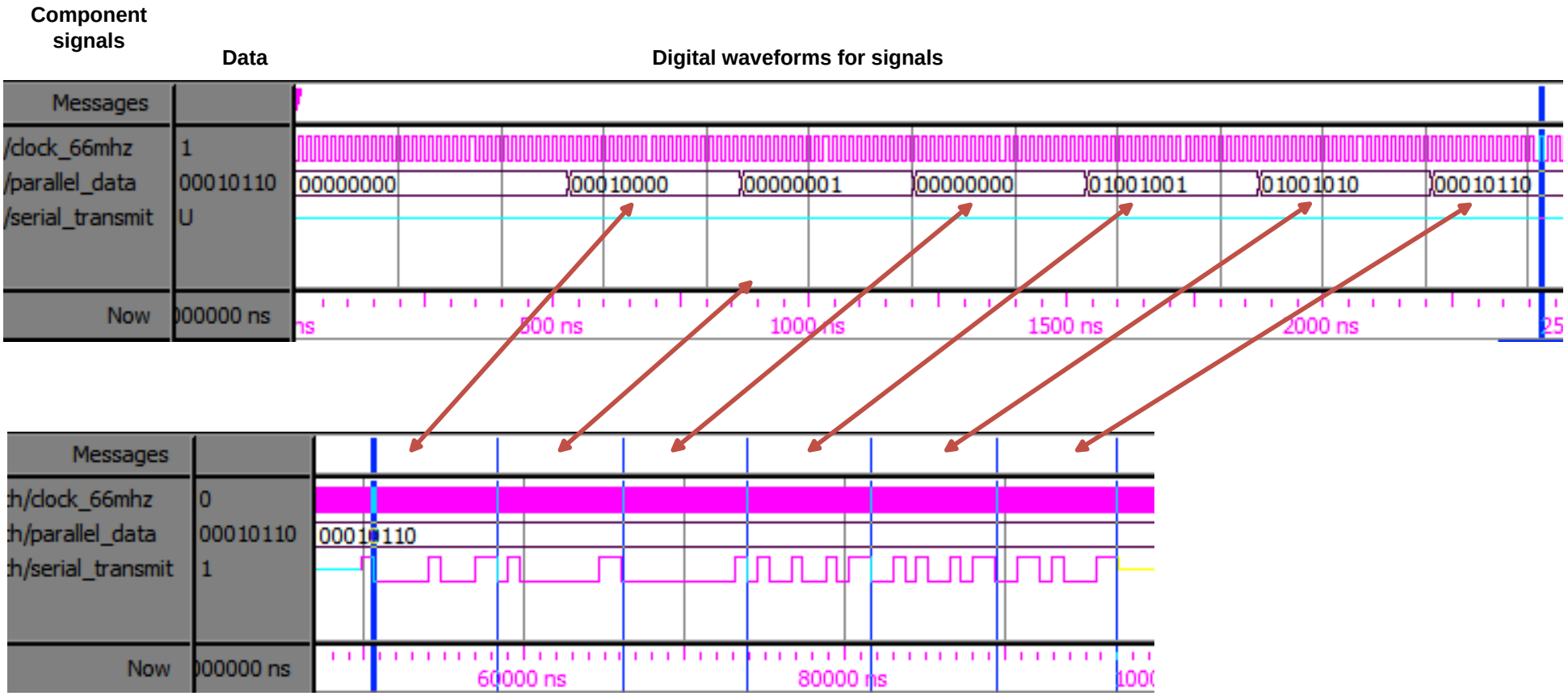


Figure 6.7 – UART transmitter simulation

Table 6.10 – Observations regarding the UART transmitter

	Observation
1	The signal *'Clock_66MHz'* is used for all the PROFIBUS operations. In the first waveform, it is clearly noted that the parallel data is sent to the UART transmitter at a much higher bit rate that the serial signal. Since the PROFIBUS main controller and the UART transmitter is connected internally (in logic), much higher data rates are achieved. The parallel data is 8 bits wide. The UART transmitter calculates the parity bit of the transferring byte and adds the start, stop and parity bit to the serial data.
2	Each arrow in the figure points to the parallel data that is converted to serial data and transmitted over the serial signal. Since the bit rate of the serial signal is much lower than that of the parallel data, the PROFIBUS clock signal has a much lower resolution and cannot be seen in detail in this figure.

6.4.5 UART receiver verification

The UART receiver replicates the UART transmitter, with the difference lying in the fact that the UART receiver converts received serial data from the slave to parallel data and transfer this data to the PROFIBUS main controller. The data flow design shown in figure 6.6 helps to illustrate the functionality of the UART receiver. Figure 6.8 shows the simulation results of the UART receiver with the accompanying observations in table 6.11. The arrow in this figure is used to indicate how the 11 bits of received serial data are converted to an eight bit parallel value.

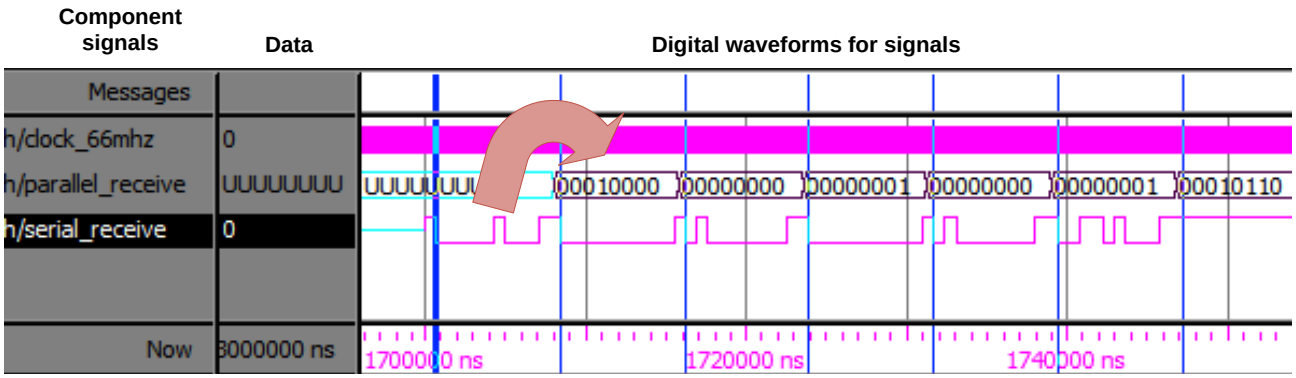


Figure 6.8 – UART receiver verification

Table 6.11 – Observations regarding the UART receiver

	Observation
1	Serial data is received as specified by the PROFIBUS character frame on the *'serial_receive'* signal and converted to parallel data indicated by the *'parallel_receive'* signal.
2	The start, stop and parity bit are removed by the UART receiver before transferring the data to the PROFIBUS main controller.

6.4.6 Parity and cyclic redundancy checks verification

PROFIBUS uses both parity and a cyclic redundancy check in the form of a frame check sequence to ensure that data is transmitted without errors and to perform error detection. The CRC value is calculated by adding specified data values of the transmitted data together in binary without adding the overflow. The requirements of the parity and the CRC are discussed in table 6.12.

Table 6.12 – PROFIBUS parity and CRC requirements

Error detection method	**Requirement**
Parity detection and error flagging	Calculate parity bit of transmitting and receiving data Signal parity error to transmitter and re-request data
Cyclic redundancy checks with error correction	Calculate parity bit of transmitting and receiving data Signal cyclic redundancy check error to transmitter and re-request data

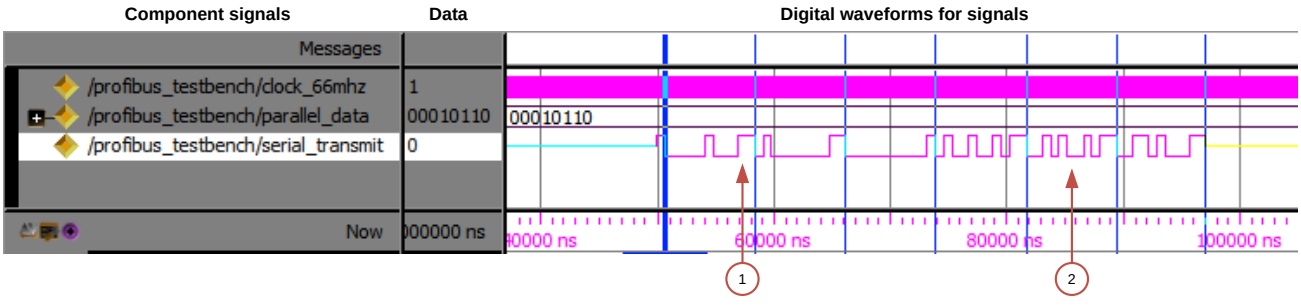


Figure 6.9 – Parity and CRC transmission

The simulation in figure 6.9 displays transmitting data with the parity and CRC added to the character and telegram frame. Table 6.13 discusses the applicable observations regarding this simulation.

Table 6.13 – Observations regarding PROFIBUS parity and CRC transmission

Simulation marker	**Observation**
General	Six characters are transmitted in this simulation although the *'parallel_data'* signal only displays the last byte that is transmitted.
1	Marker 1 points to the parity bit added to the first byte of data. The data byte being transmitted is '00010000'. Since even parity is used in PROFIBUS, the amount of '1' bits in a byte is counted. If this value is an even value in decimal format, a '0' bit will be added or a '1' bit if the value is uneven. In this simulation the amount of '1' bits are one, meaning a '1' byte is added as parity. This is seen at marker 1.
2	Marker 2 points to the frame check sequence (FCS) character of this telegram transmission. The FMS is a cyclic redundancy check byte calculated by adding character 2, 3 and 4 in this telegram without the overflow. Mathematically this value is calculated as follows: Unit1 + Unit 2 + Unit3 = 00000001 + 00000000 + 01001001 = 01001010 Transferring this character from least significant bit to most significant bit produces the character transmitted at marker 2, indicating the correct FCS value.

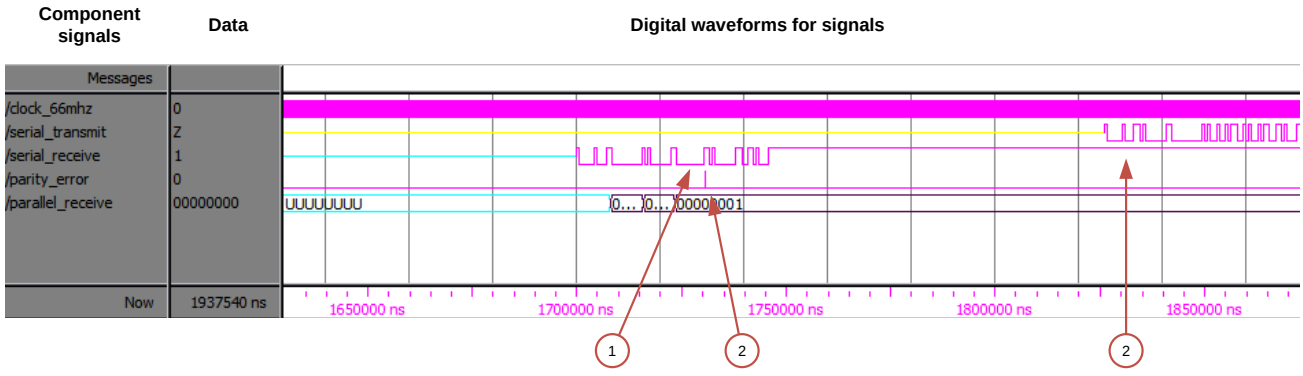


Figure 6.10 – Parity detection simulation

The simulation in figure 6.10 displays how the UART receiver detects parity errors. An error was induced in the test bench for this simulation. Table 6.14 discusses the functionality of the parity error detection of the UART receiver.

Table 6.14 – Observations regarding PROFIBUS parity detection

Simulation marker	**Observation**
1	Marker 1 points to a faulty parity bit received on the *'serial_receive'* signal. Based on the character received (00000000), the parity bit is supposed to be '0' instead of the received '1'.
2	Marker 2 shows a high pulse on the *'parity_error'* signal .When a parity error is received, the *'parity_error'* signal alerts the UART transmitter that a parity error has occurred. The UART receiver will also stop sampling the incoming serial data and not transmit the data back to the PROFIBUS main controller
3	Marker 3 points to the *'serial_transmit'* signal. On occurrence of a parity error, the UART transmitter resends the previous data to request a new response from the slave device.

The simulation in figure 6.11 displays the frame check sequence or CRC of the UART receiver. A frame check sequence error was induced in the test bench of this simulation to verify how the UART receiver will handle this situation. The observations regarding this simulation are discussed in table 6.15.

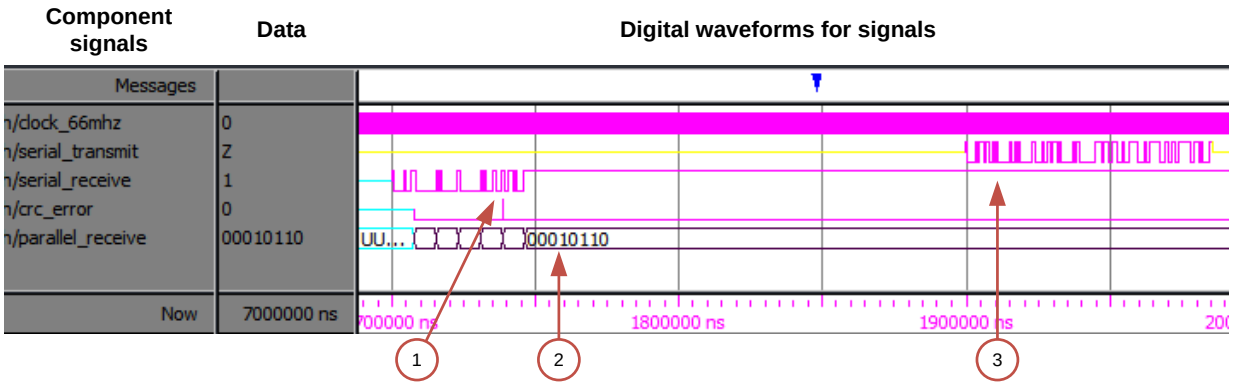


Figure 6.11 – CRC detection simulation

Table 6.15 – Observations regarding PROFIBUS CRC detection

Simulation marker	**Observation**
1	Marker 1 points to the received frame check sequence received on the *'serial_receive'* signal that is used to perform cyclic redundancy checks. The frame check sequence value received on the serial signal was deliberately induced with an error in the simulation test bench. The *'crc_error'* signal is triggered on the occurrence of the error. When a frame check sequence error is received, the *'parity_error'* signal alerts the UART transmitter that a frame check sequence error has occurred.
2	Marker 2 indicates that the data is still received from the serial signal but not transmitted to the PROFIBUS main controller.
3	Marker 3 displays the same occurrence as in the previous simulation where the UART transmitter resends the previous data on occurrence of a frame check sequence error.

6.4.7 Timing and synchronization verification

As discussed in chapter 5, the PROFIBUS network was designed to transfer data at a bit rate of 1.5 Mbps. It is therefore necessary to ensure that data is transmitted and received at the specified bit rate. A 66 MHz clock signal was used as the PROFIBUS clock signal to transfer the data. The following three simulation results illustrate various aspects of timing and synchronization regarding the PROFIBUS protocol design. Table 6.16 summarizes the requirements and specifications of the timing and synchronization.

Table 6.16 – PROFIBUS timing and synchronization requirements and specifications

Timing aspect	**Requirements and specifications**
Clock generation	Generate 66 MHz clock signal
Bit rate	1.5 Mbps
Serial to parallel conversion	Convert 1.5 Mbps serial data to parallel data
Transmit and receive driver direction control	Control direction of RS 485 half-duplex driver

In figure 6.12, a single bit of serial transmitted data together with the 2 clock signals of the design is displayed as simulated. Table 6.17 discusses the observations regarding this simulation result.

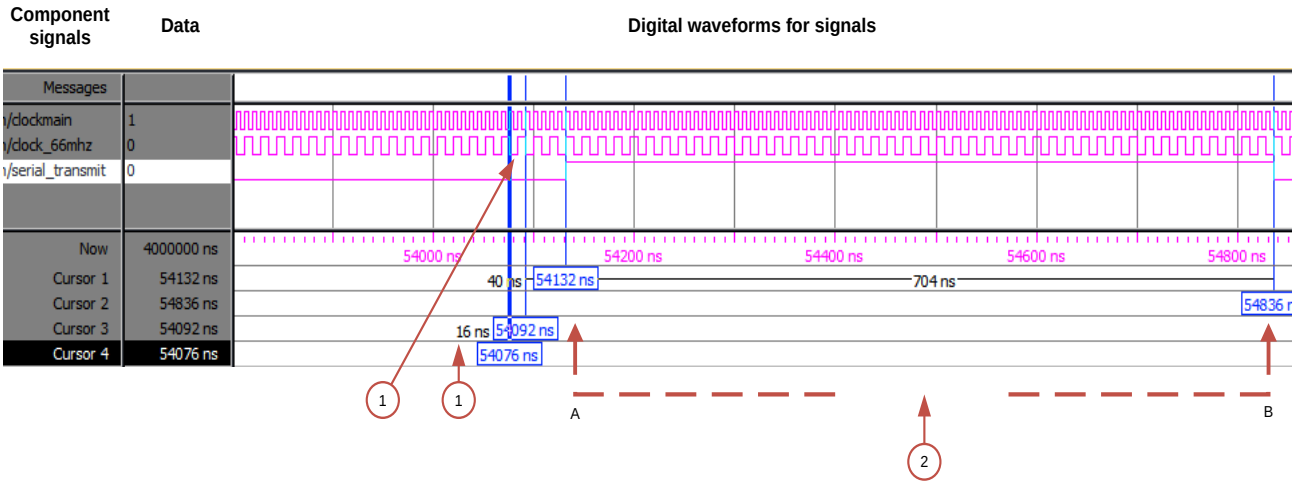


Figure 6.12 – PROFIBUS clock and bit rate simulation

Table 6.17 – Observations regarding PROFIBUS clock and bit rate

Simulation marker	Observation
1	Marker 1 both point to one 66 MHz clock cycle and the corresponding time of the clock cycle. To achieve a 66MHz clock signal, a clock cycle time needs to be 15.15 ns. Modelsim® does however not allow for floating-point values to be used in the test bench and therefore the clock cycle time used in simulation was 16 ns, producing a clock cycle signal of 62.5 MHz. The resulting clock speed of the implementation was 66.5 MHz. This value was adjusted for the implementation and will be discussed in the next chapter.
2	Marker 2 shows one high bit of serial data being transmitted. With a clock cycle time of 15.15 ns a bit rate of 1.5 Mbps is achieved by transmitting a byte every 44 clock cycles resulting in a bit time of 666 ns. With the reduced clock cycle time, 44 clock cycles at 62.5 MHz results in a bit time of 704 ns as shown by marker 2. The validation results will show the correct implementation clock cycle time.

In order to receive data, the receiving serial data has to be sampled at precise instances to capture the data from the slave device. This is illustrated in figure 6.13 and discussed in table 6.18.

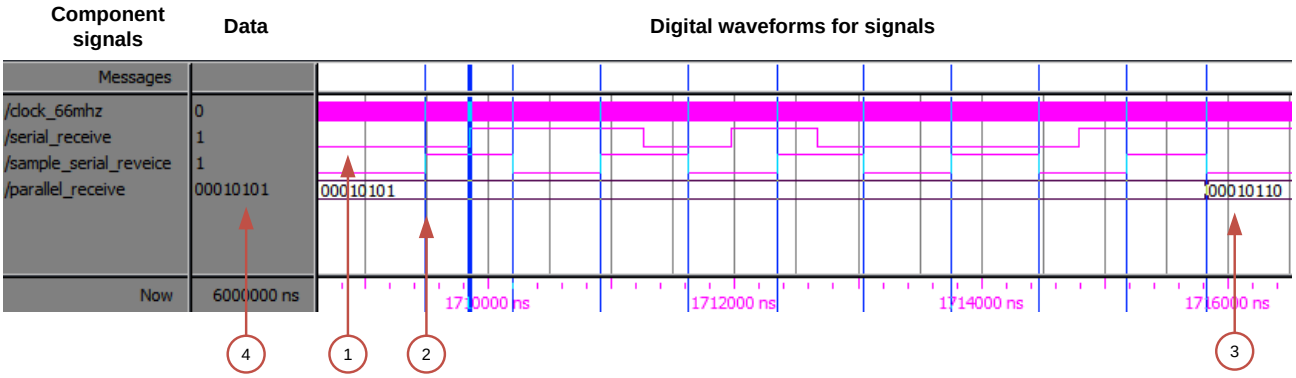


Figure 6.13 – Serial sampling simulation

Table 6.18 – Observations regarding serial sampling

Simulation marker	**Observation**
1	Marker 1 points to the start bit of the receiving character. This bit is only used to indicate a new character and not sampled.
2	Marker 2 points to the first sampling point of the data. The signal *'serial_sample_receive'* changes from '0' to '1' at each new sample indicating that every bit from the first data bit to the stop bit in the character is sampled. Sampling occurs after 22 clock cycles of a bit to ensure that no noise or jitter values are sampled. At 22 clock cycles data will be sampled in the middle of the data bit. This is the perfect position to allow the UART to distinguish between a logic high and low. In chapter 7, the eye diagrams of the data will help to illustrate this method.
3	Marker 3 points to the parallel data received at the end of the serial character.

The final verification simulation in figure 6.14 shows various signals used in the PROFIBUS protocol design. Table 6.19 discusses the observations regarding the synchronization of the various signals.

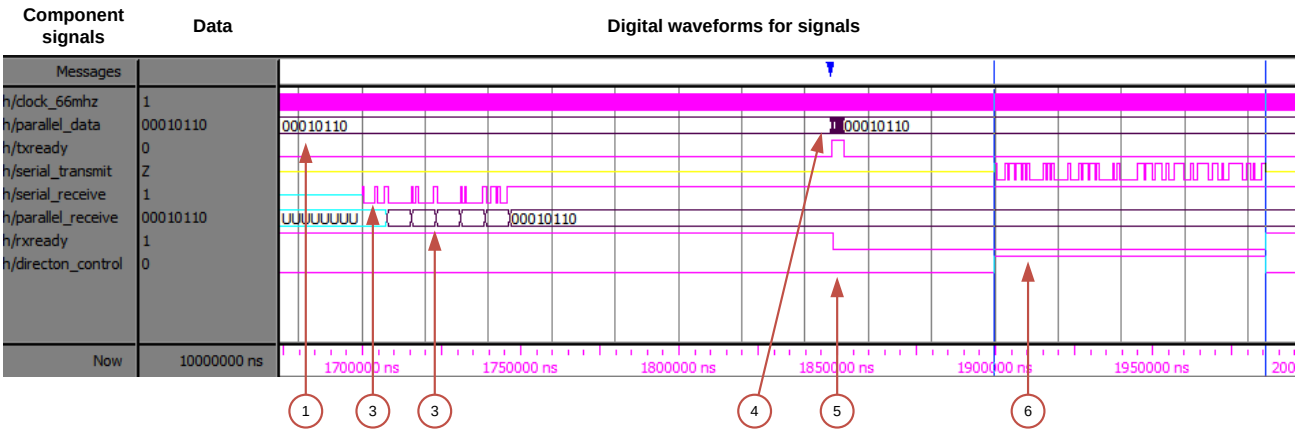


Figure 6.14 – Synchronization overview

Table 6.19 – Observations regarding synchronization

Simulation marker	**Observation**
1	Marker 1 points to the last character of the serial data that was transmitted
2	Marker 2 shows data received on the *'serial_receive'* signal. This data is converted to parallel data and transferred to the PROFIBUS main controller.
3	Marker 3 displays the parallel data sent to the PROFIBUS main controller. Upon reception of this data, the PROFIBUS main controller will determine if the previous state has completed successfully and continue to the next determined state.
4	Marker 4 shows new parallel data sent to the UART transmitter for transmission over the serial signal.
5	Marker 5 points to the signal *'TX_ready'* and *'RX_ready'*. The signal *'TX_ready'* is high for the duration of the parallel data transmission to the UART transmitter. The corresponding *'RX_ready'* signal will remain high after serial transmission to sample any response data.
6	Marker 6 points to the *'direction_control'* signal that controls the direction of the half-duplex serial signal. A logic '1' indicates the UART transmitter is transmitting data, and a logic '0' indicates that the UART receiver is receiving data over the *'serial_receive'* signal.

6.4.8 Verification conclusion

Table 6.20 summarizes the specific aspects of the PROFIBUS protocol design that was tested during verification.

Table 6.20 – Verification results

Physical layer component	**Verified**
Physical layer hardware design evaluation	√
Data link layer component	**Verified**
Character frame	√
Telegram frame	√
PROFIBUS state machine	√
UART transmitter	√
UART receiver	√
Parity and cyclic redundancy check evaluation	√
Timing and synchronization	√

This aim of this chapter was to verify the design of the PROFIBUS DP protocol. Each aspect of the designed protocol was verified according to the PROFIBUS DP standard. The results are summarized in table 6.20, which suggest that the requirements of the PROFIBUS DP standard have been met in the design. Chapter 7 will commence by validating the implementation of the protocol on the ADES in the same manner as presented in this chapter. Real-time data of the implemented protocol will be captured with a digital oscilloscope and evaluated. The physical layer is also valuated in detail with the use of eye diagram.

Chapter 7

Validation of the implemented protocol

The previous chapter discussed the verification of the designed protocol. Chapter 7 follows the same approach in evaluating every aspect of the implemented protocol on the ADES. Real-time data is captured and analyzed. The physical layer is also evaluated in detail in this chapter. Furthermore, the implemented PROFIBUS DP communication network is validated to determine the functionality of the implemented protocol. Lastly, the developed PROFIBUS DP protocol will be compared with a commercial off-the-shelf counterpart, to determine the feasibility of this project.

7.1 Introduction

In chapter 6, the difference between verification and validation was established with regards to this project. Validation means establishing and documenting evidence which provides a high degree of assurance that a process will consistently produce a result or product meeting its predetermined specifications. The validation process will therefore determine if the implementation has met its specifications that were discussed in chapter 5.

7.2 Test and validation methods

In chapter 4, the ADES was specified and how the developed protocol will be applied in this system. In order to validate the developed PROFIBUS DP protocol, various tests are performed on the application of the developed protocol on the ADES. Although the verification has shown the correct design of the PROFIBUS protocol, validation will ensure that the implemented PROFIBUS DP protocol functions correctly in the real-time environment while external and internal noise is presnt. All validation tests were performed while the ADES and RDS was fully functional. This implies that the maximum amount of external noise were present while the tests were performed. The physical layer and data link layer of the OSI reference model is again evaluated in order to validate the design and implementation. The following tests regarding the applicable layer are performed:

Physical layer

- Waveform analysis
- RS 485 driver analysis
- Detail eye diagram and analysis of transmitting and receiving data while the rotor of the RDS is levitated
- Jitter and noise evaluation of transmitting and receiving data while the rotor of the RDS is levitated

Data link layer

As with verification, the following aspects of the PROFIBUS DP protocol will be validated during real-time operation by comparing the results with the PROFIBUS DP standard and the ADES requirements:

- Character frame
- Telegram frame
- PROFIBUS state machine
- UART transmitter
- UART receiver
- PROFIBUS main control
- Parity and cyclic redundancy check evaluation
- Timing and synchronization

Finally the developed PROFIBUS DP protocol will be compared with the commercial off-the-shelf PROFIBUS PMC module. This comparison will be done based on methods used to compare various Fieldbus systems. Throughput, data coding efficiency, medium access efficiency, cyclic times, responsiveness, bit-error-ratio and implementation cost will be compared. This will help to determine the comparative quality of the VHDL-based PROFIBUS DP protocol.

7.3 Physical layer validation

In order to validate the physical layer of the PROFIBUS DP protocol design, it must be executed in a realistic environment. This implies that the PROFIBUS DP protocol design must be implemented on the active magnetic drive electronic system (ADES) and validated according to the requirements of the ADES and the PROFIBUS DP standard.

Validation will therefore commence on the ADES environment. Figure 7.1 illustrates how the PROFIBUS DP protocol is implemented on the ADES system.

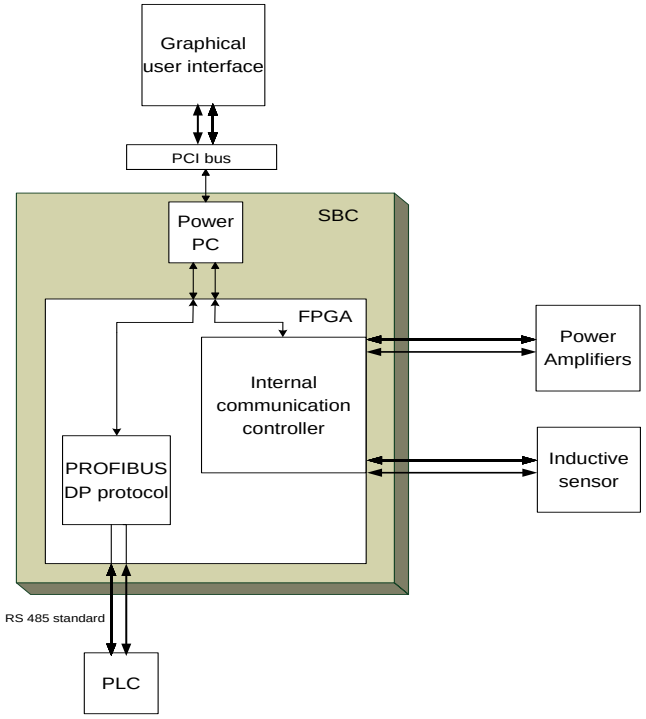


Figure 7.1 – ADES basic view with implemented PROFIBUS DP protocol

To do real-time evaluation of the developed protocol, the physical layer is evaluated with the use of a real-time LeCroy® oscilloscope and MATLAB® software. Real-time data is captured on the oscilloscope and either physically evaluated or analysed in MATLAB®. Real-time oscilloscopes use electrical probes to acquire a signal from a conductor and display a waveform from a single observation of a signal that may never occur again. Their versatility and ease of use makes them the preferred tool of choice for jitter and timing measurement [29]. Table 7.1 lists the various aspects of the physical layer that will be validated.

Table 7.1 – Physical layer validation

• Noise estimation
• Jitter calculation
• Value estimation
• Eye diagram construction and analysis

7.3.1 Waveform

In figure 7.2 and 7.3, 1 bit of data is transmitted and received over the PROFIBUS network. The differential RS 485 transmission cable is terminated at both ends as described in chapter 5. The data was captured with the real-time oscilloscope and displayed using MATLAB®. Using these waveforms, the rise time t_r can be calculated by measuring the time from 20% to 80% of the amplitude of the data bit.

The peak to peak amplitude of the transmitting bit in figure 7.2 is 4 V, from -2 V to +2 V. Figure 7.2 displays two data points indicating the 20% and 80% amplitude levels measured at -1.2 V and 2 V respectively. The rise time t_r is calculated by measuring the time between the two values. The same is done for figure 7.3 where the 20% and 80% voltage levels are at -2.4 V and 2.4 V respectively. The results are tabulated in table 7.2.

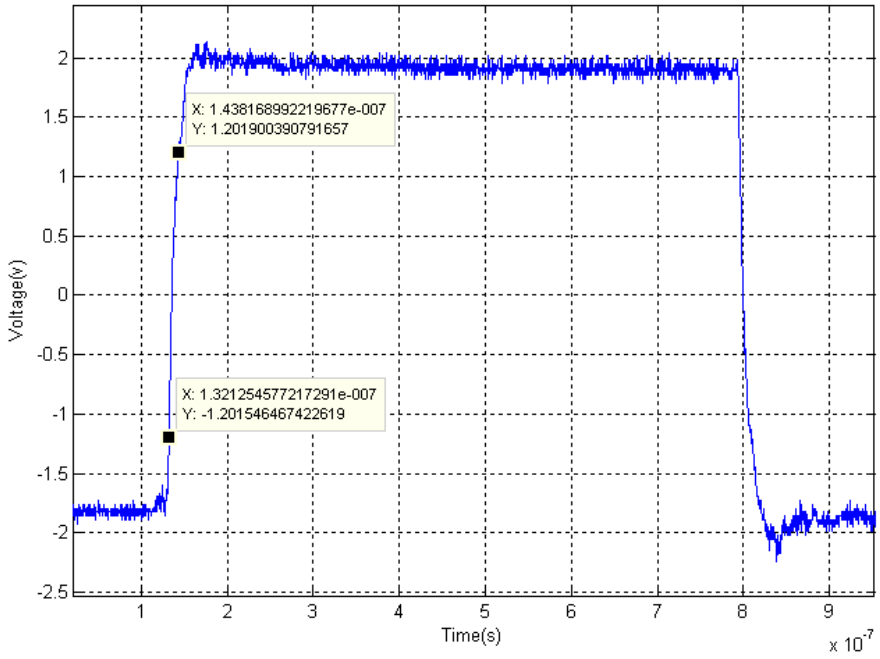


Figure 7.2 – UART transmitted bit

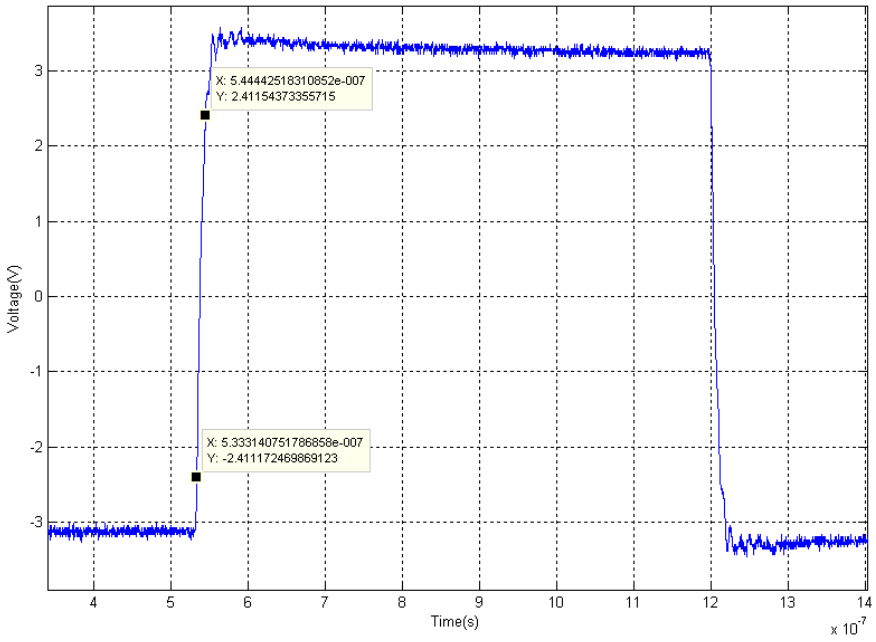


Figure 7.3 – UART received bit

Table 7.2 Rise time analysis

	Rise time t_r	**Unit interval (UI)**	$t_r = \%t_{UI}$
Transmitting bit	11.7 ns	660 ns	1.77 %
Receiving bit	11.1 ns	660 ns	1.68%

7.3.1.1 RS 485 differential eye diagrams

PROFIBUS uses the RS 485 standard for the physical layer implementation. As discussed in chapter 2, RS 485 uses two differential signals to transfer data in a half-duplex communication network. The differential signals form a mirror data transmission with an absolute difference of more than 0.2 V. Figure 7.4 displays a real-time captured waveform that displays the two differential signals as transmitted over the PROFIBUS network as well as the resulting data signal retrieved from the RS 485 signal. The transmitted and received telegram is shown in figure 7.4. This figure helps to illustrate how RS 485 signals, for non-return to zero line coding, are constructed as discussed in chapter 5.

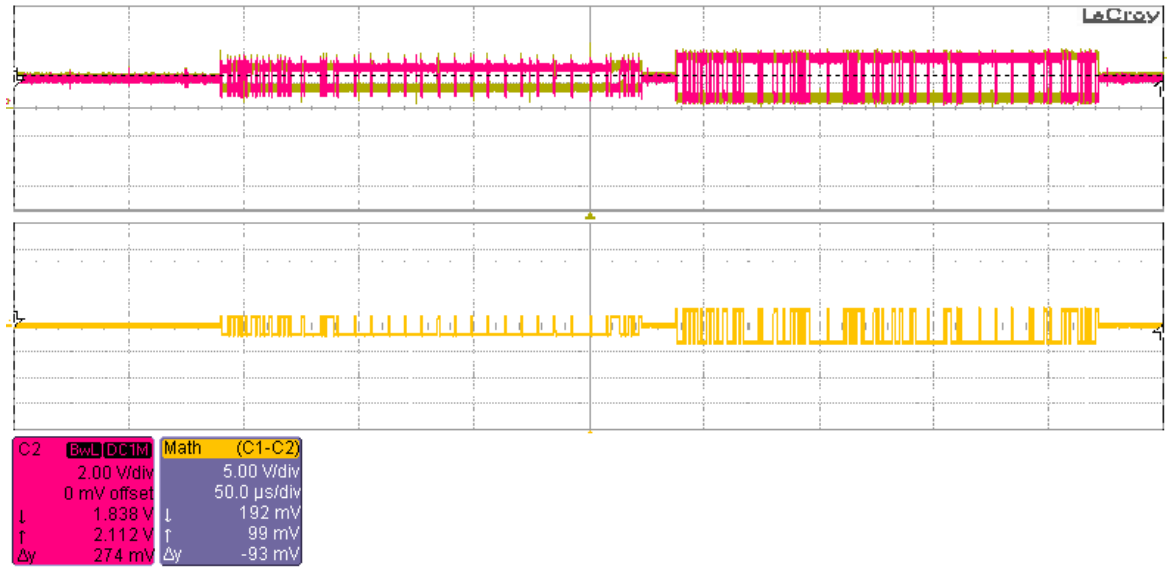


Figure 7.4 – PROFIBUS RS 485 differential signal and resulting data.

7.3.2 Eye diagram construction and analysis

An eye diagram is a time-folded representation of an electrical or optical signal that carries digital information. Time folding is prepared with the time interval used to transmit a single bit of information. The implemented unit interval specified for the 1.5 Mbps data rate on the ADES is 640 ns. Eye diagrams represent significant events, for example transitions or edges that occur at the same periodic intervals. In an eye diagram a signal is represented in the context of the receiver and reflects its ability to make decisions about the digital content [29].

A typical eye diagram for a non-return-to-zero (NRZ) signal is shown in figure 7.5. The eye diagram is a plot of voltage versus relative bit time. NRZ signalling requires the bit to remain at ‘1’ or ‘0’ for the duration of the bit. By overlaying multiple bits of data in unit interval periods, the eye is formed. The centre of the eye represents the ideal sampling time of the bit. The main function of the eye diagram is to demonstrate sufficient eye opening near the ideal sampling instance. A large eye opening ensures that the receiver has significant time to determine whether the bit is logic high or low. Various other signal characteristics can be determined with an eye diagram including jitter and noise as shown in figure 7.5. Using visual inspection and mathematical analysis, the eye diagrams of the PROFIBUS protocol is evaluated.

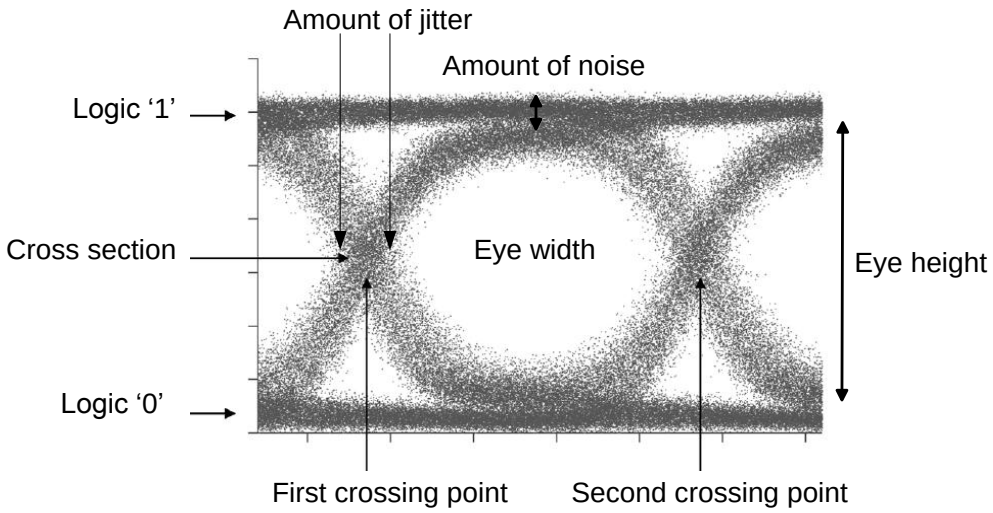


Figure 7.5 – Properties of eye diagram

7.3.2.1 Visual inspection

In order for a receiving device to distinguish between a ‘0’ and a ‘1’ on a data signal, a good amplitude difference between a logic ‘0’ and ‘1’ should be present. As discussed in chapter 5, the sampling point occurs in the middle of the unit interval. A good eye opening will ensure correct sampling of the data. A low jitter value of between 5% and 10% is required for correct data acquisition. These characteristics can be evaluated with visual inspection before detail mathematical analysis is done.

Figure 7.6 displays an eye diagram at the transmitter, constructed by plotting data in MATLAB® that was captured with the real-time LeCroy® oscilloscope. Data is overlaid at unit interval times of 640 ns to construct the eye diagram. Data is transmitted at a bit rate of 1.5 Mbps at a voltage level from -2 V to 2 V. By inspecting the eye diagram visually, the following can be observed.

- A good eye opening is present which ensures correct data sampling
- Amplitude separation between the logic low and logic high is present
- Jitter is less than 10% of the unit interval

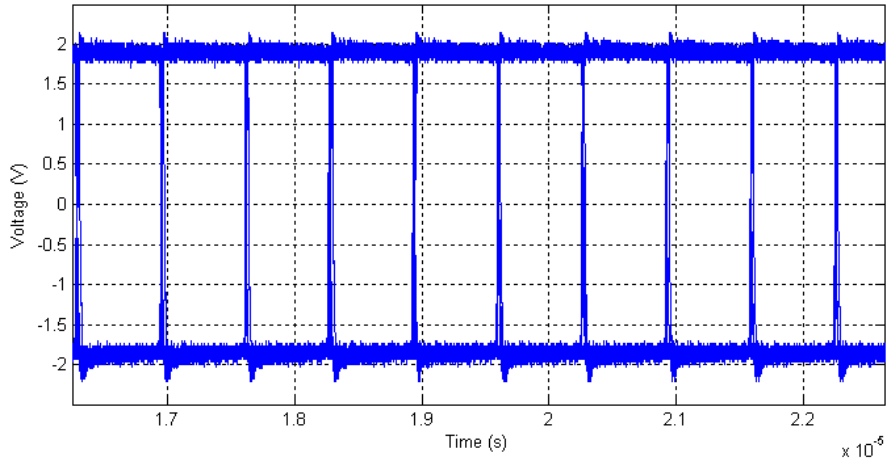


Figure 7.6 – Eye diagram of transmitter constructed with oscilloscope

Figure 7.7 displays an eye diagram at the receiver, constructed by plotting data in MATLAB® that was captured with the real-time oscilloscope. Data is overlaid at unit interval times of 640 ns to construct the eye diagram. Data is transmitted at a bit rate of 1.5 Mbps at a voltage level between -3 V and 3 V. By inspecting the eye diagram visually, the following can be observed.

- A good eye opening is present which ensures correct data sampling
- Amplitude separation between the logic low and logic high is present and better than the transmitting data
- Jitter is less than 10% of the unit interval

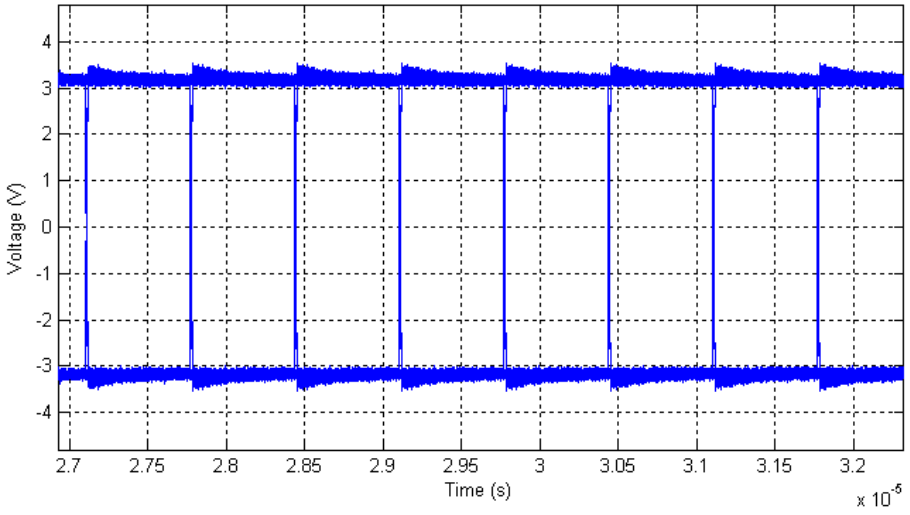


Figure 7.7 – Eye diagram of receiver constructed with oscilloscope

7.3.3 Detail eye diagram analysis

In order to objectively determine the quality of eye diagrams, compare them and test for compliance, various features must be measured. Each eye measurement characterizes a specific feature of the displayed record. Parametric or histogram measurements can be used for eye diagram measurements. Using these measurements, the following characteristics of the data signal can be obtained [29]:

- Eye height
- Eye width
- Extinction ratio
- Crossing percent (eye crossing point expressed as a percentage of the eye height)
- Jitter peak-to-peak
- Jitter RMS
- Noise peak-to-peak
- Noise RMS
- Signal-to-noise ratio

The parametric method is used to measure the various characteristics of the PROFIBUS protocol. Data is captured with the real-time LeCroy® oscilloscope and eye diagrams are constructed in MATLAB®. This allows much more data to be overlaid in order to do a more realistic analysis of the eye diagram. For the construction of the eye diagrams, 20 data signals were overlaid at unit interval periods of 640 ns to produce the eye diagrams in the following sections. All the equations used for calculations are obtained from [29].

7.3.3.1 Detail transmitter eye diagram analysis

Figure 7.8 displays the eye diagram constructed at the transmitter end. Data is transmitted at a bit rate of 1.5 Mbps with a unit interval of 640 ns. In this figure, data captured from the oscilloscope is overlaid to produce an eye diagram that represents all the characteristics of the transmitting signal. Parametric measurements are made on the eye diagram to measure the characteristics of the data signal listed in the previous section.

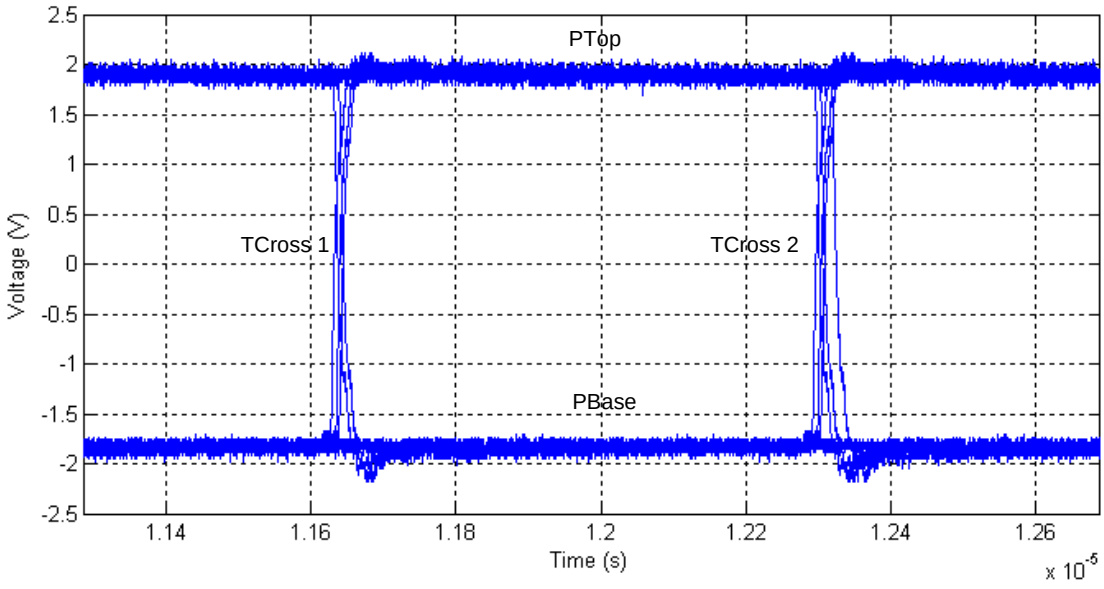


Figure 7.8 – Detail eye diagram of the transmitting signal

Figure 7.8 displays one eye of the transmitting data. By using the mean and standard deviation at *TCross 1, TCross 2, PTop and PBase,* the characteristics of the signal can be calculated. Table 7.3 lists the values associated with the mean and standard deviation at each point of interest.

Table 7.3 – Transmitting eye diagram properties

Property	**Symbol**	**Position**	**Value**
Standard deviation	σ	TCross1	4.29 ns
Mean	μ	TCross1	11.64 us
Standard Deviation	σ	TCross2	5.69 ns
Mean	μ	TCross2	12.31 us
Standard Deviation	σ	PTop	46.3 mV
Mean	μ	PTop	1.92 V
Standard Deviation	σ	PBase	38.5 mV
Mean	μ	PBase	-1.81 V

Eye height

Eye height can be determined using (7.1):

$$Eye\ Height = \left(\mu_{PTop} - 3\sigma_{PTop}\right) - \left(\mu_{PBase} + 3\sigma_{PBase)}\right) \quad (7.1)$$

$$Eye\ Height = \left(1.92 - 3(0.0463)\right) - \left(-1.81 + 3(0.0385)\right)$$

$$= 3.475\ \mathrm{V}$$

σ_{PTop} and σ_{PBase} is the standard deviation from the mean values μ_{PTop} and μ_{PBase} respectively.

Eye width

Eye width is the measurement of the eye width in nanoseconds, shown in (7.2)

$$\mathrm{Eye\ Width} = \left(\mu_{TCross2} - 3\sigma_{TCross2}\right) - \left(\mu_{TCross1} + 3\sigma_{TCross1}\right) \quad (7.2)$$

$$\mathrm{Eye\ Width} = \left(12310 - 3(5.69)\right) - \left(11641 + 3(4.29)\right)$$

$$= 639.06\ ns$$

The eye width helps to identify the correct position to sample data. The centre of the eye is considered as the best position to sample data and ensures that the receiver can distinguish between the logic ‘0’ and logic ‘1’ values.

Jitter RMS

$$\mathrm{Jitter\ RMS} = \sigma_{TCross1} \quad (6.3)$$

$$\mathrm{Percentage\ Crossing\ Skew} = 6\ \mathrm{sigma\ jitter} \quad (6.4)$$

The root mean square jitter is the RMS jitter value for the edge jitter for the horizontal units, while 6 sigma jitter accounts for 99.999% of all jitter in the signal [29].

$$\mathrm{Jitter\ RMS} = 4.29\ \mathrm{ns}$$

$$6 \times \mathrm{sigma\ jitter} = 6 \times 4.29$$

$$= 25.74\ \mathrm{ns}$$

Percentage jitter

$$Jitter\ = \frac{Percentage\ Crossing\ Skew}{Unit\ Interval} \times 100\% \tag{6.5}$$

The percentage jitter is representative of the amount of jitter throughout the entire data signal.

$$Jitter\ = \frac{25.74}{640} \times 100 = 4.02\%$$

Noise peak-to-peak

$$Noise\ pp = \max(PTop) - \min(PTop) \tag{6.6}$$

Peak-to-peak noise is the value of the noise at the top or base of the signal, where *max(PTop)* and *min(PTop)* is the maximum and minimum values at *PTop* respectively.

$$Noise\ pp = 2.01399 - 1.76201$$
$$= 0.25198\ \mathrm{V}$$

Noise RMS

$$Noise\ RMS = \sigma_{PTop}\ \ or\ \ \sigma_{PBase} \tag{6.7}$$

Noise RMS is the root mean square value of the noise at either the top or base of the signal.

$$Noise\ RMS = 46.3\ \mathrm{mV}\ \ or\ \ 38.5\ \mathrm{mV}$$

Signal-to-noise ratio

$$Signal\ to\ noise\ Ratio = \frac{(\mu_{PTop} - \mu_{PBase})}{(\sigma_{PTop} + \sigma_{PBase})} \tag{6.8}$$

Signal-to-noise ratio is the ratio of the amplitude to the noise of the signal at either the top or the base.

$$Signal\ to\ noise\ Ratio = \frac{(1.92 - (-1.81)}{(0.0463 + \ 0.0385)}$$
$$= 43.99\ \text{V}$$

To calculate the dB value of the signal-to-noise ratio, (6.9) is used.

$$20 \log_{10} SNR \tag{6.9}$$
$$20\log_{10} 43.99 = 32.87\ \text{dB}$$

7.3.3.2 Detail receiver eye diagram analysis

Figure 7.9 displays the eye diagram constructed of the receiving data. In this figure, data captured from the oscilloscope is overlaid to produce an eye diagram that represents all the characteristics of the receiving signal. Parametric measurements are made on the eye diagram to measure the characteristics of the data signal listed in the previous section.

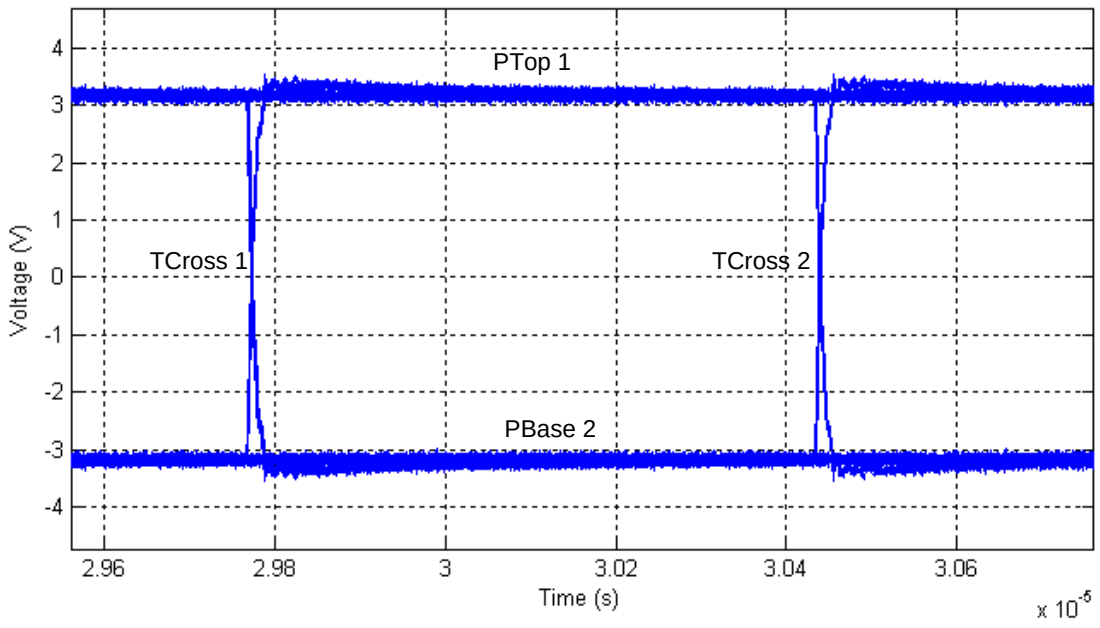


Figure 7.9 – Detail eye diagram of the receiving signal

Table 7.4 – Transmitting eye diagram properties

Property	**Symbol**	**Position**	**Value**
Standard deviation	σ	TCross1	1.26 ns
Mean	μ	TCross1	29.77 us
Standard Deviation	σ	TCross2	0.778 ns
Mean	μ	TCross2	30.44 us
Standard Deviation	σ	PTop	70.9 mV
Mean	μ	PTop	3.2 V
Standard Deviation	σ	PBase	61.5 mV
Mean	μ	PBase	-3.167 V

The same measurements made on the transmitting signal are repeated for the receiving signal. Table 7.4 lists the measured parameters of the receiving signal. The receiving data is received from a commercial PLC and will not be interpreted but forms a base point for good comparison against the developed protocol. The equations are just presented with the calculated values and not discussed.

Eye height

$$Eye\ Height = \left(\mu_{PTop} - 3\sigma_{PTop}\right) - \left(\mu_{PBase} + 3\sigma_{PBase)}\right. \tag{6.1}$$

$$Eye\ Height = \left(3.3 - 3(0.0709)\right) - \left(-3.167 + 3(0.0615)\right)$$
$$= 6.0698\ \text{V}$$

Eye width

$$Eye\ Width = \left(\mu_{TCross2} - 3\sigma_{TCross2}\right) - \left(\mu_{TCross1} + 3\sigma_{TCross1}\right) \tag{6.2}$$

$$Eye\ Width = \left(30440 - 3(0.778)\right) - \left(29770 + 3(1.26)\right)$$
$$= 663.886\ \text{ns}$$

Jitter RMS

$$Jitter\ RMS = \sigma_{TCross1} \tag{6.3}$$

$$Percentage\ Crossing\ Skew = 6\ sigma\ jitter \tag{6.4}$$

$$Jitter\ RMS = 1.26\ ns$$

$$6\ sigma\ jitter = 6 \times 1.26 = 7.56\ \text{ns}$$

Percentage jitter

$$Jitter = \frac{Percentage\ Crossing\ Skew}{Unit\ Interval} \times 100\% \tag{6.5}$$

$$Jitter = \frac{7.56}{663} \times 100 = 1.14\%$$

Noise peak-to-peak

$$Noise\ pp = \max(PTop) - \min(PTop) \tag{6.6}$$

$$Noise\ pp = 3.39205 - 2.81613$$

$$= 0.578\ \text{V}$$

Noise RMS

$$Noise\ RMS = \sigma_{PTop}\ \ or\ \ \sigma_{PBase} \tag{6.7}$$

$$Noise\ RMS = 70.9\ \text{mV}\ \ or\ \ 61.5\ \text{mV}$$

Signal-to-noise ratio

$$Signal\ to\ noise\ Ratio = \frac{(\mu_{PTop} - \mu_{PBase})}{(\sigma_{PTop} + \sigma_{PBase})} \tag{6.8}$$

$$Signal\ to\ noise\ Ratio = \frac{(3.2 - (-3.167)}{(0.0709 + 0.0615)}$$

$$= 48.09\ \text{V}$$

$$20 \log_{10} SNR \tag{6.9}$$

$$20\log_{10} 48.09 = 33.64\ \text{dB}$$

7.3.4 Eye measurement conclusion

Although the PLC slave uses the same physical layer as the developed PROFIBUS protocol, the focus of the conclusions will rest mainly on the transmitting data. By examining the eye diagram measurements of the transmitting signal, the following conclusions can be drawn:

- The unit interval was implemented to be 640 ns which results in a bit rate of 1.5 Mbps. The eye of the transmitting data has a width of 639 ns and can be interpreted as representing a 1.51 Mbps unit interval that can be used for data sampling.
- The height of the eye is 3.475 V. This, together with the eye width, produces a very 'open' eye and represents an excellent signal for data transmission. The perfect eye would have a 4 V height and 640 ns width which suggests the results of the eye openness is sufficient.
- Very low jitter was calculated, where low jitter is considered to be less than 5% of the signal. This ensures that no bit errors will occur as a result of jitter.
- The transmitter has a signal-to-noise ratio of 32.87 dB, which will not have any noticeable effect on the channel. A SNR of 10 db would imply that the eye of the channel would almost be completely closed. SNR can be directly related to bit error ratio, where fewer bits occur with higher SNR. The PROFIBUS cable used is only 2 m long, but it is clear from the SNR and quality factor that the cable may easily be extended to the designed length of 100 m without added repeaters.

The physical layer conforms to all the physical requirements of the PROFIBUS standard and the results in this section have shown the quality of the layer. The next step in validation is to ensure that the data link layer meets the PROFIBUS specifications.

7.4 Data link layer validation

Although eye diagrams can predict physical layer performance, it does not ensure that higher layers of communications or data links work correctly. Even if the transmitted signal meets the physical properties, it does not guarantee device interoperability with an arbitrary receiver [29]. It is therefore necessary to validate the higher data link layer of communication to validate the data transmission according to the PROFIBUS DP standard. Table 7.5 lists the PROFIBUS requirements and specifications that will be validated. Validation will commence with the real-time LeCroy® oscilloscope and MATLAB® software. The waveforms used for validation are captured on the oscilloscope while the ADES is fully operational by connecting the digital probes of the LeCroy® oscilloscope to the differential channels of the PROFIBUS network.

Table 7.5 – Data link layer validation

PROFIBUS subdivisions	Requirements
Character frame	Non-return to zero bit coding 1 Low start bit, 1 high stop bit 8 Data bits Least significant bit to most significant bit transmission 1 Even parity bit
Telegram frame	**Telegram with no data field** Start delimiter Destination address Source address Function code Frame check sequence End delimiter **Telegram with variable data length** Start delimiter Length Length repeated Start delimiter repeated Destination address Source address Function code Destination service access point Source service access point Data units Frame check sequence End delimiter
PROFIBUS state machine	Request diagnostics 1 Parameterization Configuration Request diagnostics 2 Data exchange
UART transmitter	Parallel to serial data conversion and transmission
UART receiver	Serial to parallel data conversion and transmission
Parity and cyclic redundancy check evaluation	Parity detection and error flagging Cyclic redundancy checks with error correction
Timing and synchronization	Clock generation Bit rate Parallel to serial conversion Serial to parallel conversion Transmit and receive driver direction control

7.4.1 Character frame validation

As discussed in chapter 5, PROFIBUS specifies a specific character frame used for data transmission. These specifications are listed in table 7.6. One character frame is captured with the oscilloscope and presented by using MATLAB® software in figure 7.10. The corresponding observations and results are listed in table 7.7.

Table 7.6 – PROFIBUS DP character frame specifications

• Non-return to zero bit coding
• 1 Low start bit, 1 high stop bit
• 8 Data bits
• Least significant bit to most significant bit transmission
• 1 Even parity bit

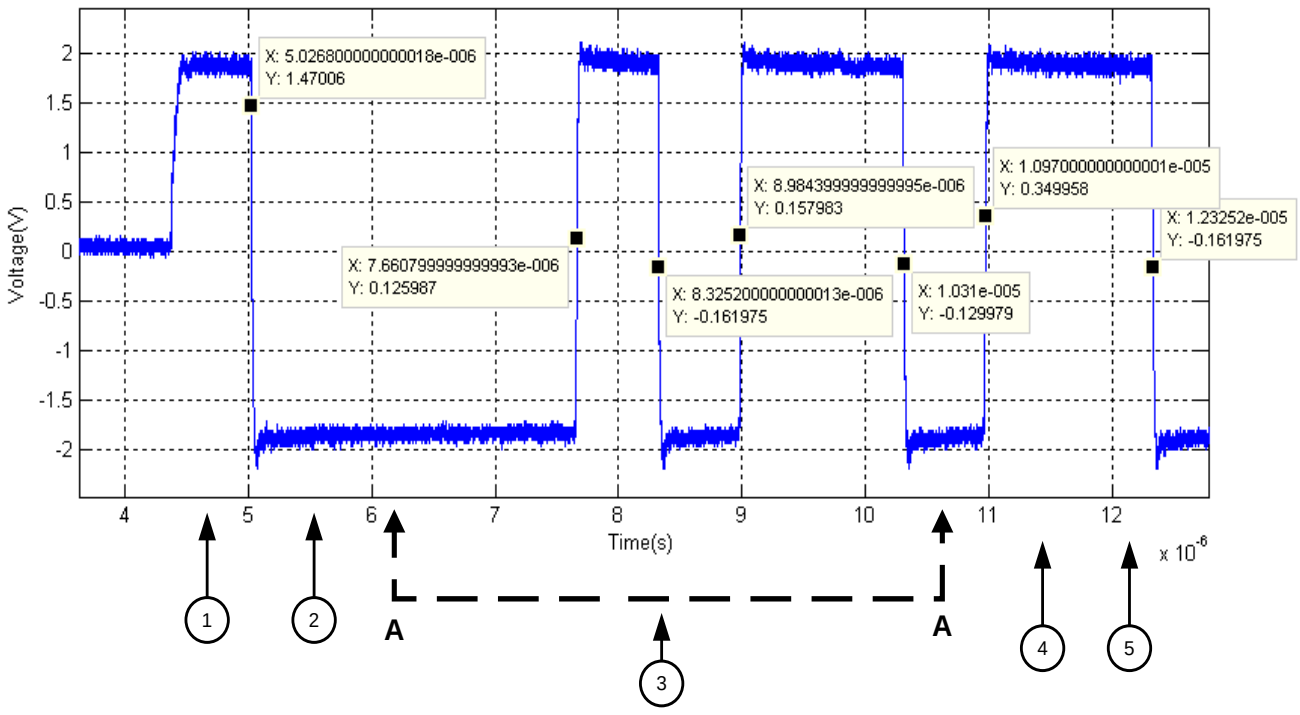


Figure 7.10 – PROFIBUS DP character frame validation

Table 7.7 – Observations concerning PROFIBUS DP character

Simulation marker	**Observation**
General	Figure 7.10 displays 11 bits of data that forms 1 character. The signal represents data being transmitted from the FPGA to the PLC device. The first bit of data occurs at $5.0268e^{-6}$ and the end of the last bit of data occurs at time $1.23252e^{-5}$. This produces a character time of 7.298 ns. Dividing this value by 11 bits of data results in a bit time of 663.4 ns. This results in a bit rate of 1.507 Mbps. This validates the designed bit time of 1.5 Mbps
1	Marker 1 points to the first logic high transition of the signal. This section of data is not used for data transmission but is required by the PROFIBUS standard. The receiving device detects data on the first falling edge of the signal, therefore the first part of the transmission will be logic high, to allow the receiver to detect the first falling edge
2	Marker 2 points to the start bit (logic '0') of the character being sent to indicate a new character
3	Marker 3 points to the 8 data bits being transmitted. The value as transmitted in time is '00010110'. This data is transmitted from least significant bit to most significant bit, implying that this data will be received as '01101000'.
4	Marker 4 points to the logic '1' parity bit of the character. Parity bits will be discussed in detail later in this section.
5	Marker 5 points to the stop bit (logic '0') of the character, indicating the end of the character.

7.4.2 Telegram frame validation

As mentioned earlier, PROFIBUS telegrams can contain up to 256 bytes, with 244 bytes of data and 11 bytes of overhead per message. The overhead is referred to as the telegram header. PROFIBUS specifies 5 types of telegrams that can be used in devices but not all are mandatory [16]. Table 7.8 lists the specifications of the two main telegrams used in this PROFIBUS design.

Table 7.8 – PROFIBUS DP telegram frame specifications

Telegram with no data field	**Specification**
Start delimiter	10H (Hexadecimal) value
Destination address	8 bit address of slave/master
Source address	8 bit address of master/slave
Function code	Identification of the type of telegram
Frame check sequence	Cyclic redundancy check frame
End delimiter	16H (Hexadecimal) value
Telegram with variable data length	**Specification**
Start delimiter	68H (Hexadecimal) value
Length	Nett data length
Length repeated	Nett data length
Start delimiter repeated	68H
Destination address	8 bit address of slave/master
Source address	8 bit address of master/slave
Function code	Identification of the type of telegram
Destination service access point	Service access point for receiver to determine the next service (Optional)
Source service access point	Service access point for transmitter to determine the next service (Optional)
Data units	1 to 244 bytes
Frame check sequence	Cyclic redundancy check frame
End delimiter	16H

Figure 7.10 and 711 show examples of the different telegram frames that was captured with the real-time oscilloscope and plotted in MATLAB® for analysis. Table 7.9 and 7.10 discuss the figures with regards to the PROFIBUS telegram specifications.

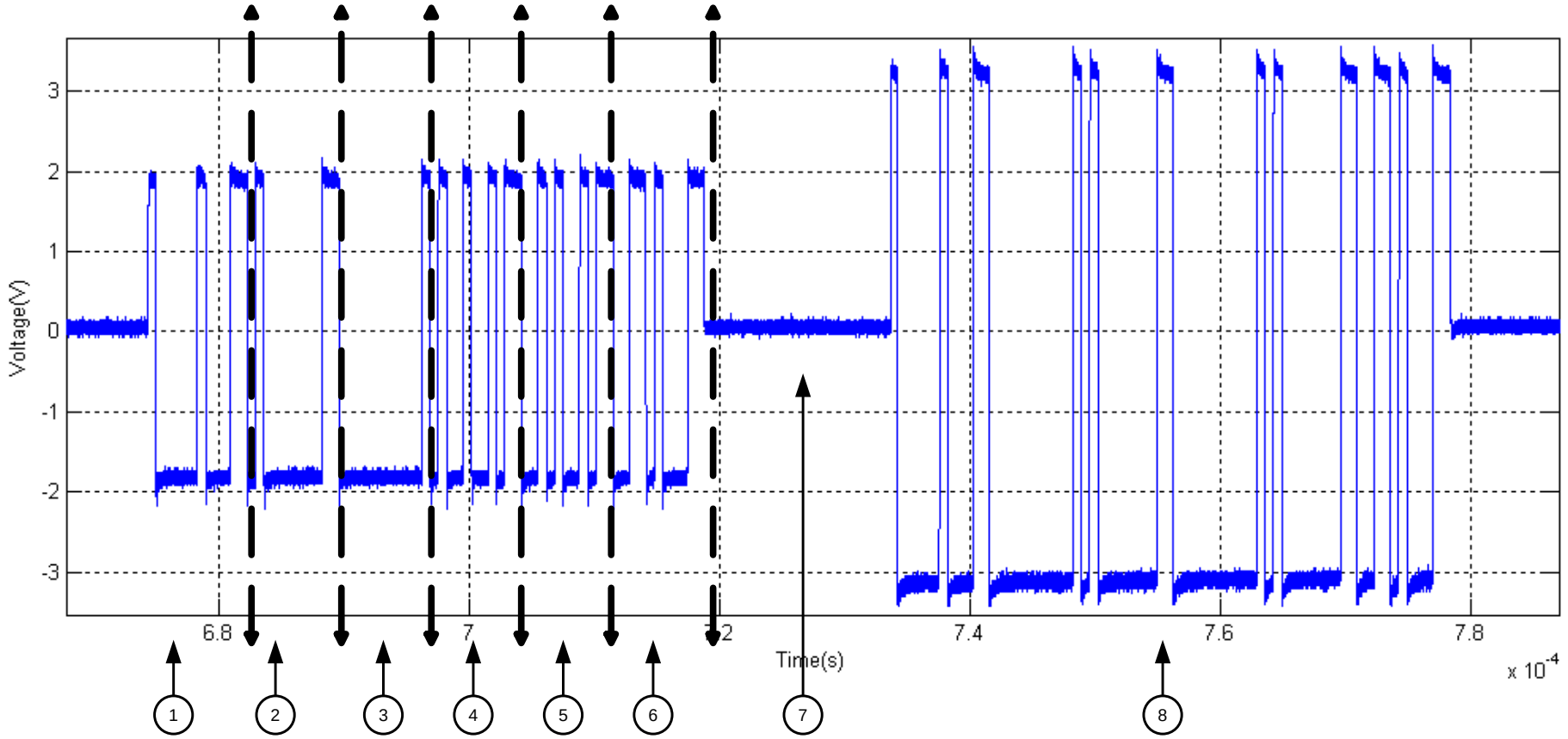


Figure 7.10 – PFOBIUS DP telegram frame 1

Table 7.9 – Observations regarding PROFIBUS DP telegram frame 1

Simulation marker	**Observation**
General	Two telegrams are pictured in figure 7.10. The first telegram is transmitted from the FPGA to the PLC. The second telegram is the returning response from the PLC. The voltage differences of the telegrams are processed by the RS 485 drivers discussed in chapter 5 and do not influence the data.
1	Marker 1 points to the character '0**00001000**11', indicating the start delimiter value '00010000' or '10H'
2	Marker 2 points to the character '0**10000000**11', which can be translated as the address, '00000001',of the receiver (slave)
3	Marker 3 points to the character '0**00000000**01', which can be translated as the address, '00000000' of the transmitter (master)
4	Marker 4 shows the function code character' 0**10010010**11'. This specific value '01001001' is a function code that specifies a request to the receiver to return the status of the receiver.
5	Marker 5 points to the frame check character '0**01010010**11' with data units, '01001010'. This value is calculated by applying a cyclic redundancy check (CRC) value to the frame. CRCs are discussed in detail later in this section.
6	Marker 6 points to the character '0**01101000**11' with data units, '00010110' or '16H', that indicates the end of the telegram.

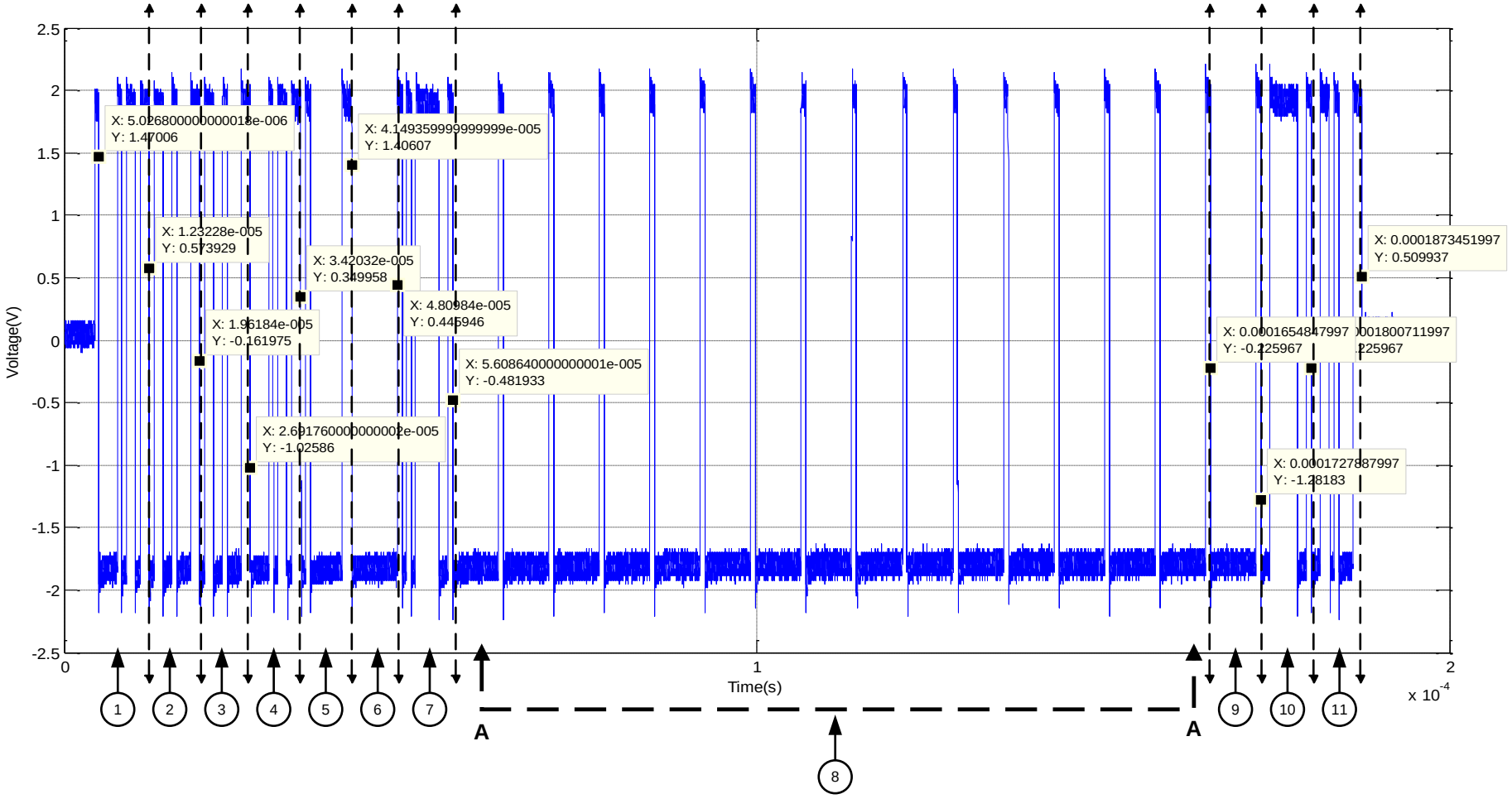


Figure 7.11 – PROFIBUS DP telegram frame 2

Table 7.10 – Observations regarding PROFIBUS DP telegram frame 2

Marker	Observation
General	Figure 7.11 displays the telegram frame used for data transmission. Each marker represents a new character frame with the exception of the 16 data units which is not labelled. The start and end time markers of the first character are labelled as $5.0268e^{-6}$ and $1.23228e^{-5}$ respectively. The time interval for this character is 7.296 ns, resulting in a unit interval for 1 bit of 663 ns.
1	Marker 1 points to the character '0**00010110**11', indicating the start delimiter value '01101000' or '68H'
2	Marker 2 points to the value '0**11001000**11', with data units, '00010011'. Converting this value to decimal produces a value of 19. This is calculated by counting the amount of characters in the telegram (excluding the start delimiters, length, and end delimiter characters). The receiver uses this information to check the amount of characters in the telegram frame for error detection.
3	Marker 3 points to the same value as marker 2 which indicates the length is repeated for redundancy as specified by PROFIBUS.
4	Marker 4 again points to the character '0**00010110**11, indicating the start delimiter value '01101000' or '68H'
5	Marker 5 points to the character '0**10000000**11', which can be translated as the address, '00000001', of the receiver (slave). This value indicates that no SSAP character is included in the telegram.
6	Marker 6 points to the character '0**00000000**01', which can be translated as the address, '00000000' of the transmitter (master). This value indicates that no DSAP character is included in the telegram.
7	Marker 7 points to the character '0**10111110**01'. This data unit, '01111101', is a function code that requests data from the receiver.
8	Marker 8 displays the first 15 characters of data units transferred. In this telegram, all the data units have a value of '00000000'
9	Marker 9 points to the 16th data character of the frame that is transmitted to illustrate the time frame of the data unit.
10	Marker 10 points to the frame check character '0**01111110**01' with data units, '01111110'. This value is calculated by applying a cyclic redundancy check (CRC) value to the frame. CRCs are discussed in detail later in this section.
11	Marker 11 points to the character '0**01101000**11' with data units, '00010110' or '16H', that indicates the end of the telegram.

7.4.3 PROFIBUS state machine validation

When a PROFIBUS network is established, a slave device has to be assigned to a master device. This allows multiple masters and slaves to use a common bus to transmit and receive data. During this assignment, the master will establish communication with a specified slave by specifying parameters and configuring the device according to the intended network use, which is allowed by the slave [16].

During verification the transmitted states were verified. By capturing data on the real-time oscilloscope, the transmitting data as well as response telegrams from the PLC slave device can be analyzed. By evaluating the response telegrams, the next transmitting state of the PROFIBUS network is determined. Table 7.11 lists the PROFIBUS state machine requirements. The data captured in figure 7.12 A and B represents the 5 states of the PROFIBUS standard with their responses from the PLC slave device. Table 7.12 discusses the observations regarding the waveforms of figure 7.12 A and B.

Table 7.11 – PRODIBUS DP state machine requirements

PROFIBUS state	**Requirements**
Request diagnostics 1	Read slave parameterization and configuration data
Parameterization	Parameterize the slave according to the master and specifies the operating mode of the slave
Configuration	Configure the slave according to the specifications of the slave and the PROFIBUS network
Request diagnostics 2	Read slave parameterization and configuration data after configuration and parameterization
Data exchange	Cyclic transfer of data according to configuration and parameterization

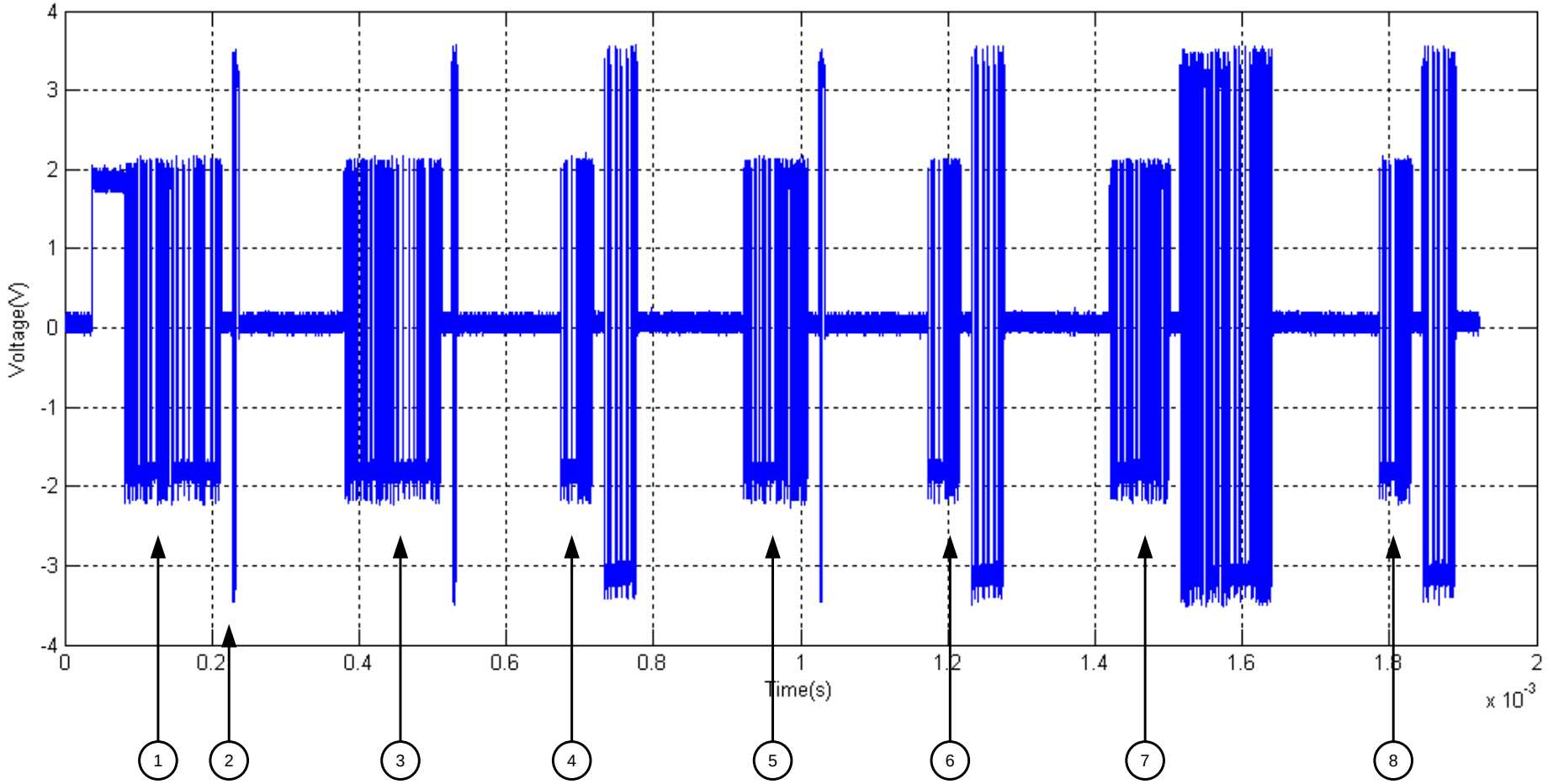


Figure 7.12 A – PROFIBUS states A

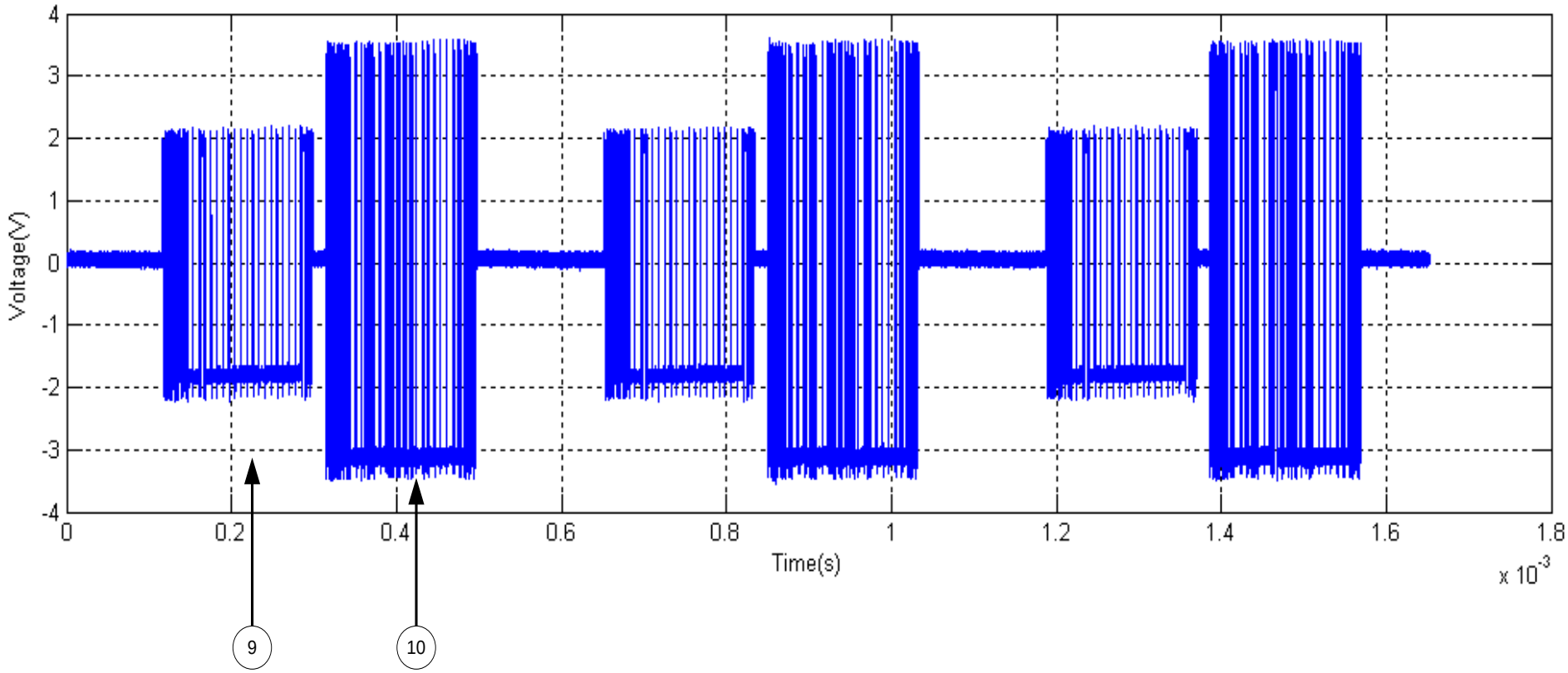


Figure 7.12 B – PROFIBUS states B

Table 7.12 – Observations regarding the PRODIBUS DP state machine

Marker	**Observation**
General	In figure 6.29 A and B, the lower voltage levels in the waveform represent the transmitting data, while the higher voltage levels represents the response telegram from the PLC slave. A new state will only occur once correct data has been received back from the receiver. The data needs to pass parity and CRC evaluation and will indicate to the transmitter (master) whether the next state can occur. In order to display all the data of the PROFIBUS network, the individual data bits cannot be distinguished. After a state transmission, the serial signal returns to 0 V. At the beginning of a new state, the serial signal goes high for an undefined time and the receiver detects the first falling edge and capture data.
1	Marker 1 points to the parameterization transmitting telegram. Since the state request diagnostics occur twice in the PROFIBUS state machine, the parameterization state is shown first.
2	Marker 2 points to the response telegram from the slave device. The higher voltage level of the response telegram is interpreted by the RS 485 drivers at each side of the transmission line.
3	Marker 3 points to a repeat of the parameterization telegram sequence. This occurs when the responder has returned data that indicates that parameterization has not completed. Only once the parameterization is complete the next state can occur.
4	Marker 4 points to a state sequence that is not mandatory in the PROFIBUS standard. This check_state sends a telegram to the slave between states, from the parameterization state until data exchange state occurs. The response telegram is once again seen by the higher voltage level.
5	Marker 5 points to the configuration sate. This state configures the slave for the amount of inputs and outputs. The slave responds with a short message indicating correct configuration or an error.
6	Marker 6 points to a repeat of the check_state.
7	Marker 7 points to the second request diagnostics state. In this state, the master checks the parameterization and configuration of the slave. If the response indicates success. The data exchange state will occur.
8	Marker 8 points to the last check_state sequence before data exchange.
9,10	Marker 9 and 10 points to the data exchange state and the response from the slave device. During this state, the amount of data specified in configuration is transmitted to and from the receiver. The network will remain in this state until an error in data transmission occurs. In the event of an error, the parameterization and configuration state will occur before data exchange can occur.

7.4.4 UART validation

In the previous section the UART transmitter and receiver was verified in the Modelsim® environment. It is however not possible to measure the internal signals of the FPGA on which the PROFIBUS protocol is implemented. As mentioned, the main functions of the UART transmitter are to receive parallel data from the PROFIBUS main control, convert the data to serial data and transmit it over the serial signal. The UART receiver replicates the UART transmitter, with the difference lying in the fact that the UART receiver converts received serial data from the slave to parallel data and transfers this data to the PROFIBUS main controller.

Figures 7.10 and 7.11 in the telegram validation section produce the necessary evidence that the UART transmitter does convert parallel data to serial data and transmit the data over the serial transmission line.

With regards to the UART receiver, figure 7.13 displays exactly when receiving data is sampled. The receiving data signal displays one character data that is received from the PLC slave device. The sample signal indicates the position at which data is sampled. During each new bit, the sample signal undergoes a logic transition to indicate the sampling point. As shown in the verification section, the start bit of the character is not sampled. As discussed in the eye diagram evaluation section, data is sampled in the middle of the unit interval of the bit. This ensures that a difference between a logic ‘0’ and ‘1’ can easily be distinguished.

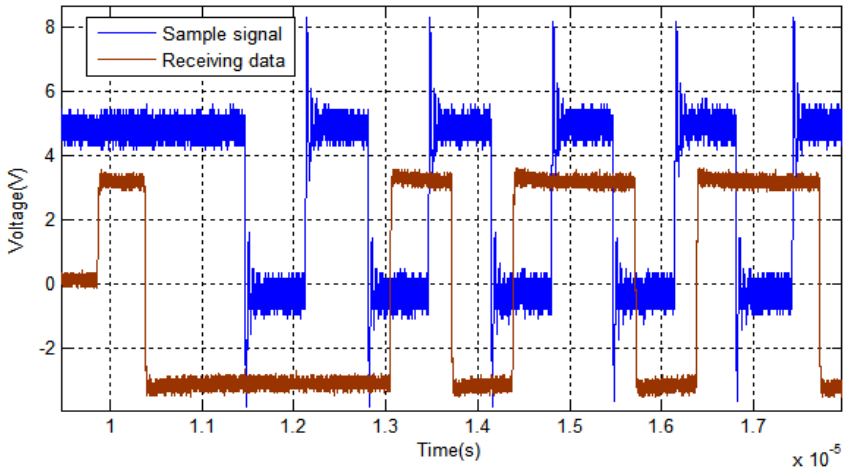


Figure 7.13 – Sampling of receiving data

7.4.5 Parity and cyclic redundancy check validation

PROFIBUS uses both parity and a cyclic redundancy check in the form of a frame check sequence to ensure that data is transmitted without errors and to do error detection. The CRC value is calculated by adding specified data values of the transmitted data together in binary without adding the overflow.

These error detection methods are not only necessary for the PROFIBUS standard, but of utmost importance for the ADES. As mentioned, the rotor delevitating system (RDS) can operate to a speed of 19,000 r/min. When the PLC is used to operate the system, an error in data transmission between the PLC and the FPGA can cause catastrophic failure of the RDS. PROFIBUS was specifically chosen as Fieldbus standard because of its robust protocol. The parity and CRC is therefore validated to ensure the accurateness of the data transmission. Table 7.13 lists the requirements of the parity and frame check sequence, or CRC.

Table 7.13 – PROFIBUS parity and CRC requirements

Error detection method	**Requirement**
Parity detection and error flagging	Calculate parity bit of transmitting and receiving data Signal parity error to transmitter and re-request data
Cyclic redundancy checks with error correction	Calculate parity bit of transmitting and receiving data Signal cyclic redundancy check error to transmitter and re-request data

To validate the parity construction of the data transmission, consider figure 7.14 which displays one data character of transmitted data. The markers A represent the 8 bytes of data in the character frame. In this character the data transmitted is ‘00010110’. The amount of logic ‘1’ bits in this character is 3. Since even parity is used in PROFIBUS, this value results in a logic ‘1’ added as parity. This is seen at marker 1 which represents the parity bit. A parity bit is added to each character that is transmitted and received over the PROFIBUS network.

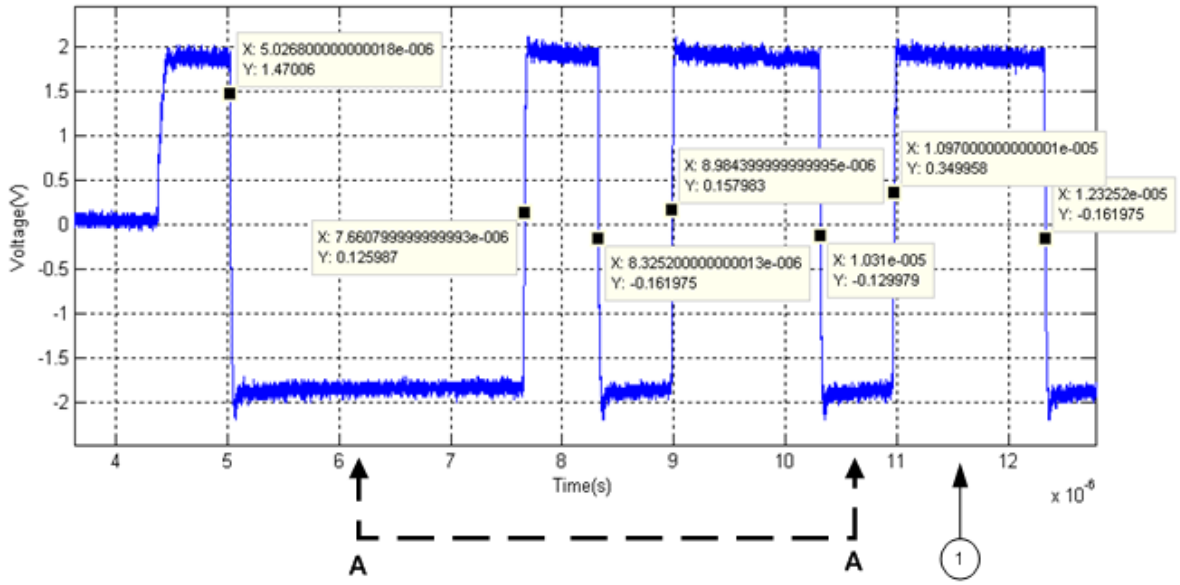


Figure 7.14 – Parity transmission

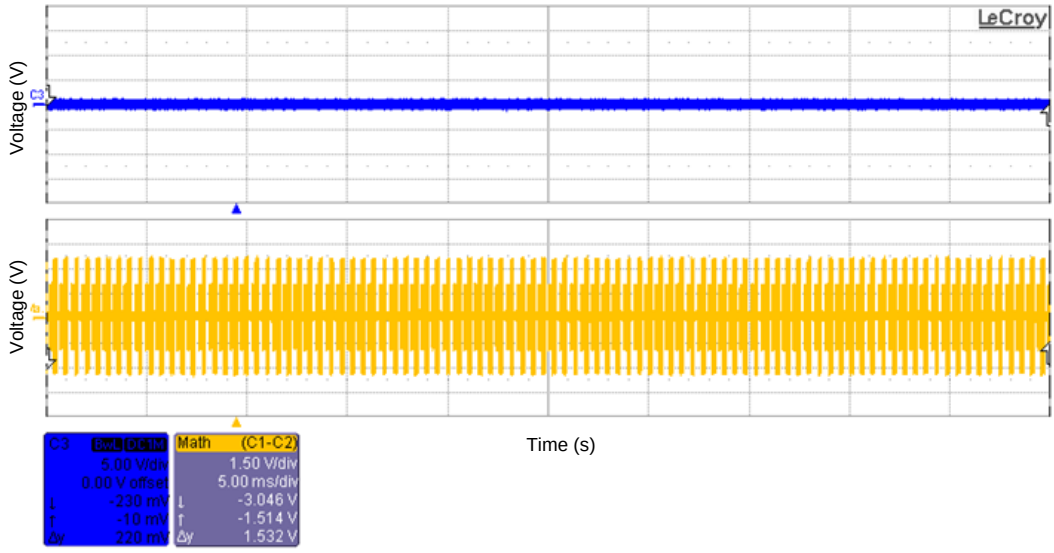


Figure 7.15 – Parity detection

Figure 7.15 shows the result of a parity error test performed on the PROFIBUS network. The top signal is connected to the UART receiver to indicate when a parity error is received with a logic high pulse. A time division of 5 ms/div is displayed as captured with the oscilloscope in real-time. In these 50 ms of data transmission not 1 parity error is detected. Roughly about 100 data exchange sequences occur in the 50 ms, resulting in 52800 bits of data that are transmitted without an error. Since no parity error could be captured, alternative methods are used at the end of this chapter to evaluate the error-rate of the VHDL-based protocol.

To validate the CRC of the PROFIBUS network, figure 7.16 represents a telegram frame transmitted over the network to the PLC device. Six characters are transmitted in this telegram. The values of the telegram are as follows:

- Start delimiter – '00010000'
- Destination address – '00000001'
- Source address – '00000000'
- Function code – '01001001'
- **Frame check sequence – '01001010'**
- End delimiter – '00010110'

The frame check sequence in this telegram is calculated by adding the values of the destination address, source address and function code. This results in a binary value of '01001010'. By transferring this data from least significant bit to most significant bit and adding a start bit (0), stop bit (1) and parity bit ('1' for even parity), the transmitted character should result in '0**01010010**11'. The frame check sequence character at marker 1 displays this value as transferred in the PROFIBUS telegram which validates the cyclic redundancy check of the designed PROFIBUS protocol.

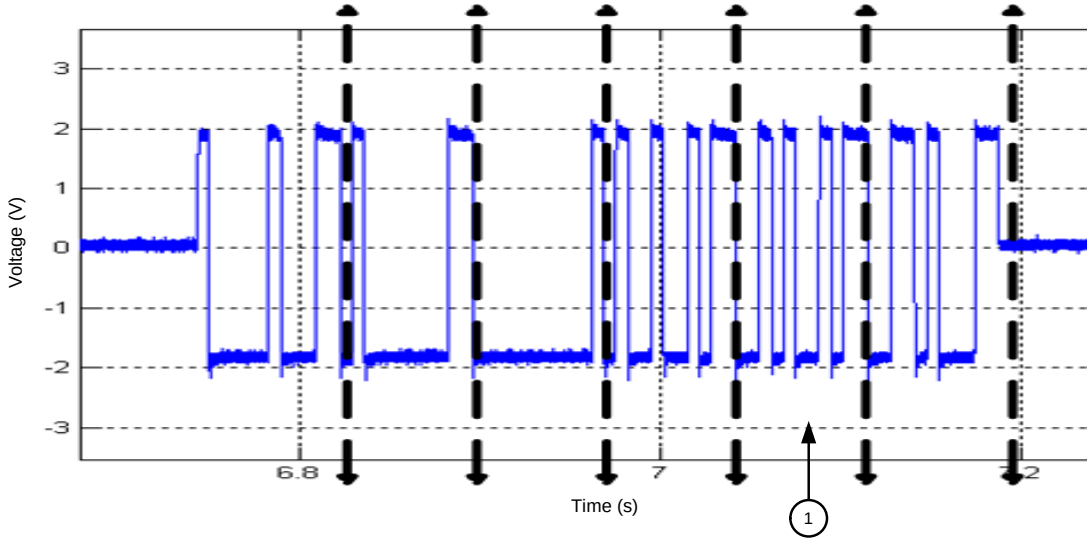


Figure 7.16 – PROFIBUS CRC calculation

7.4.6 Timing and synchronization validation

PROFIBUS uses the RS 485 standard as transmission. Half-duplex communication applies to the RS 485 standard which implies that the direction of the transmission signal must be controlled. Together with this, a 66 MHz clock signal is generated to convert parallel data to serial data and transmit the data at a bit rate of 1.5 Mbps over the serial transmission line. The timing and synchronization requirements and specifications are summarized in table 7.14.

Table 7.14 – PROFIBUS timing and synchronization requirements and specifications

Timing aspect	**Requirements and specifications**
Clock generation	Generate 66 MHz clock signal
Bit rate	1.5 Mbps
Parallel to serial conversion	Convert parallel data to 1.5 Mbps serial data
Serial to parallel conversion	Convert 1.5 Mbps serial data to parallel data
Transmit and receive driver direction control	Control direction of RS 485 half-duplex driver

Figure 7.17 shows the clock signal that is produced and used for the PROFIBUS protocol. By dividing the two time markers of figure 7.17 it is seen that one clock pulse is generated every 15.02 ns. This relates to a frequency of 66.57 MHz. To produce a bit rate of 1.5 Mbps, a single bit is transmitted for 44 clock cycles.

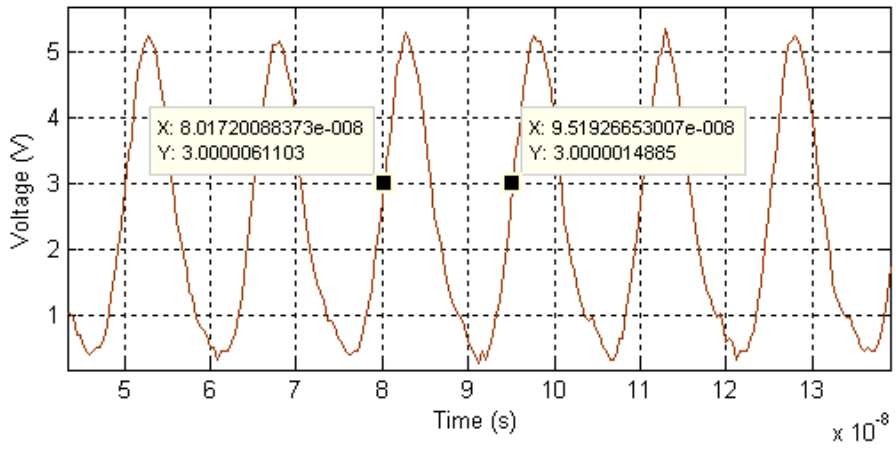


Figure 7.17 – PROFIBUS clock generation

Figure 7.18 displays the same clock signal together with 1 high bit of transmitted data. In this figure the logic ‘1’ value is transmitted approximately 44 clock cycles to produce a bit rate that vary from 1.494 Mbps to 1.5104 Mbps. Data is sampled in the middle of the transmitting data (22 clock cycles) and this small variation in bit rate will not affect the sampling of the data.

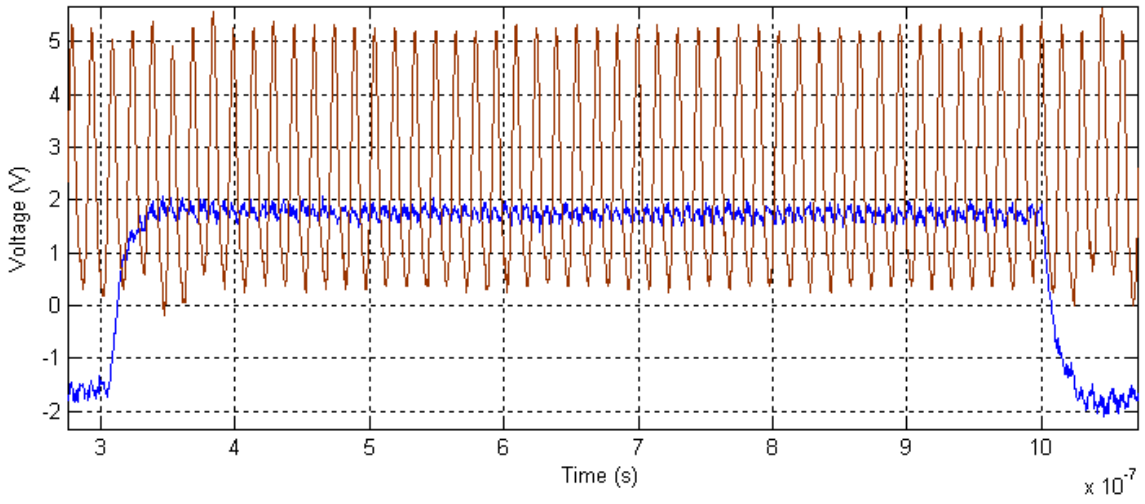


Figure 7.18 – PROFIBUS bit rate

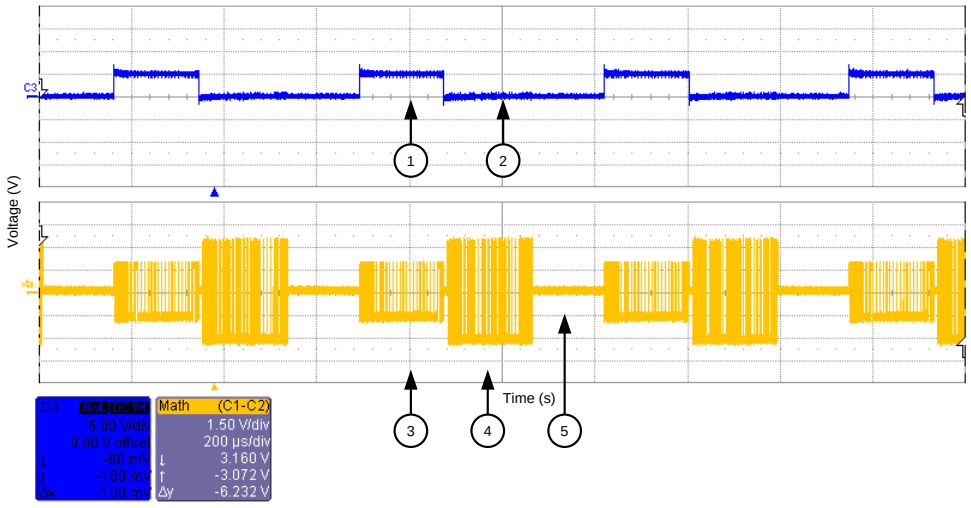


Figure 7.19 – UART direction control

The RS 485 drivers discussed in chapter 5 can either receive of transmit data over the differential signals. Figure 5.7 displayed the physical layer design of the RS 485 differential drivers. As shown, the transmitting direction signal is attached to the SP491E driver. A logic ‘1’ transmitted to the SP491E driver will control the driver to transmit data while a logic ‘0’ will control the driver to receive

data. In figure 7.19, the first signal show the direction control signal used to control the RS 485, while the second signal illustrate transmitting data and receiving data on the PROFIBUS network.

Marker 1 points to a logic ‘1’ indicating transmission while marker 2 points to a logic ‘0’ which indicates the driver is receiving data. Marker 3 illustrates the data transmitted in the transmission state and marker 4 points to the response data received during the receiving state of the RS 485 driver. Marker 5 points to ‘dead’ time of the transmission line during which the PROFIBUS main controller analyses the received data to determine the next state.

7.4.7 Validation conclusion

The specific characteristics and specifications regarding the physical layer and data link layer of the developed PROFIBUS protocol that were validated are listed in table 7.15.

Table 7.15 – Validation results

Physical layer component	**Verified**
Physical layer hardware design evaluation	√
Low noise	√
Low signal jitter	√
Good value estimation	√
Sufficient eye diagram results for data transfer during visual analysis	√
Sufficient eye diagram results for data transfer during mathematical analysis	√
Data link layer component	**Verified**
Character frame	√
Telegram frame	√
PROFIBUS state machine	√
UART transmitter	√
UART receiver	√
Parity and cyclic redundancy check evaluation	√
Timing and synchronization	√

7.5 ADES system validation

In chapter 4 and 5 the ADES system was discussed in detail. It was shown that the rotor delevitating system (RDS) could be operated either by means of a graphical user interface connected to the FPGA through the PCI bus, or by means of the PLC device. The goal of the PROFIBUS network was to establish communication between the PLC and the FPGA which controls the RDS. Once the communication was established, the following was communicated between the PLC and FPGA:

- Local/Remote operation
- Levitate/Delevitate the RDS

The PLC has a physical key that can switch operation of the RDS between the PLC or the graphical user interface (GUI). The PLC is programmed to transmit different data values to indicate the state of the key as shown in table 7.15. The two main buttons of the PLC – levitated and delevitated – also has specific data values assigned to each of them shown in table 7.16. The PLC transmits the specific data units to the FPGA, based on which button on the PLC is activated. If the key connected to the PLC indicates that the GUI is operating the system, specific data regarding the state of the RDS is transmitted from the FPGA to the PLC. All these data values are summarized in table 7.16 and will be validated to illustrate the operation of the ADES.

Table 7.16 – PROFIBUS data units

PLC transmitting	**Corresponding data value 1**	**Corresponding data value 2**	**Data position in transmission**
Key device (GUI control)	‘10101011’ – GUI control	‘00000000’ – PLC control	3
Key device (PLC control)	‘10101011’ – PLC control	‘00000000’ – GUI control	4
Levitated	‘10101011’ - levitated	‘11110101’ - delevitated	1
Delevitated	‘10101011’ - delevitated	‘11110101’ - levitated	6
FPGA transmitting	**Corresponding data value 1**	**Corresponding data value 2**	**Data position in transmission**
Levitated	‘10101011’ - levitated	‘11110101’ - delevitated	1
Delevitated	‘10101011’ - delevitated	‘11110101’ - levitated	6

With regards to table 7.16, the accuracy of the data exchange between the PLC and the FPGA can be evaluated by examining the exact data that is transmitted and received. Figure 7.20 displays the transmitting data telegram to the PLC. This data is captured in real-time with the LeCroy® telescope. Markers 1 to 6 display the data transmitted from the FPGA to the PLC to indicate that the rotor delevitation system (RDS) is levitated. The 6 data values are transmitted as shown in table 7.17.

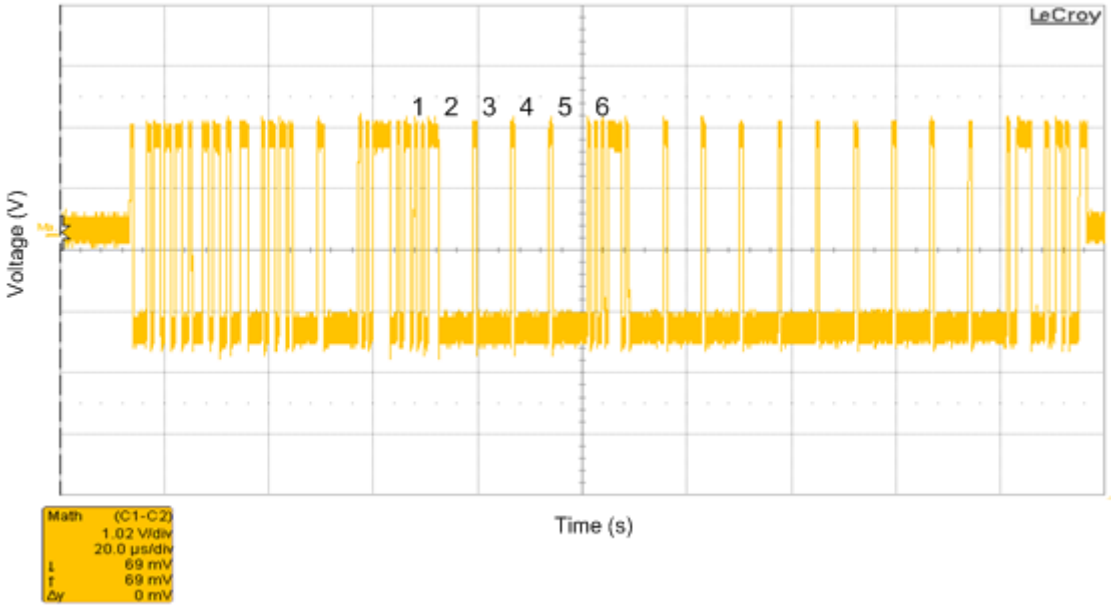


Figure 7.20 – Transmitting data while RDS is levitated

Table 7.17 – Transmitting data to PLC while RDS is levitated

Data position	**Binary value**	**Indication**
1	10101011	Levitated = true
2	00000000	Open
3	00000000	Open
4	00000000	Open
5	00000000	Open
6	11110101	Delevitated = false

Figure 7.21 is representative of the same data as in figure 7.20 with the exception that the RDS is delevitated. Table 7.18 lists the six data values that are transmitted. Note the change in data values of the levitated and delevitated bytes.

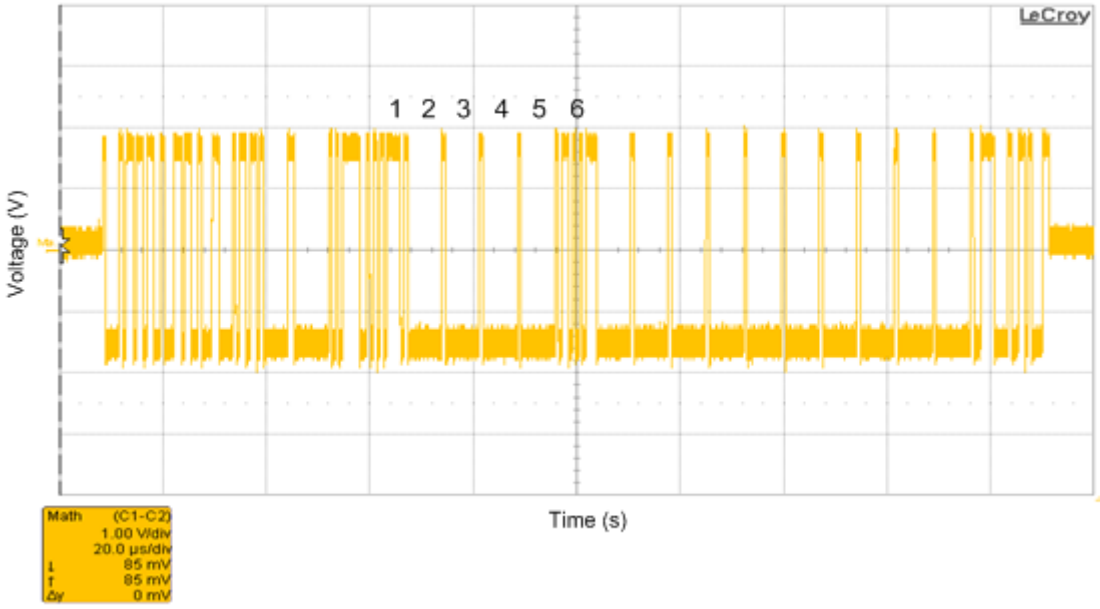


Figure 7.21 – Transmitting data while RDS is delevitated

Table 7.18 – Transmitting data to PLC while RDS is delevitated

Data position	**Binary value**	**Indication**
1	11110101	Levitated = false
2	00000000	Open
3	00000000	Open
4	00000000	Open
5	00000000	Open
6	10101011	Delevitated = true

Figures 7.22 and 7.23 display the data received from the PLC. Note that although the six data values have similarities, they are not the same data. The six data values received from the PLC are used by the FPGA to determine if the PLC or the graphical user interface is operating the RDS. If the PLC has operational control, the data from the PLC is used to levitate or delevitate the RDS. A key switch device is used to indicate this to the FPGA. Together with this, the PLC will indicate to the FPGA when the delevitated or levitated buttons on the PLC are activated. The data values in figures 7.21 and 7.22 that are received from the PLC are listed in tables 7.19 and 7.20 respectively.

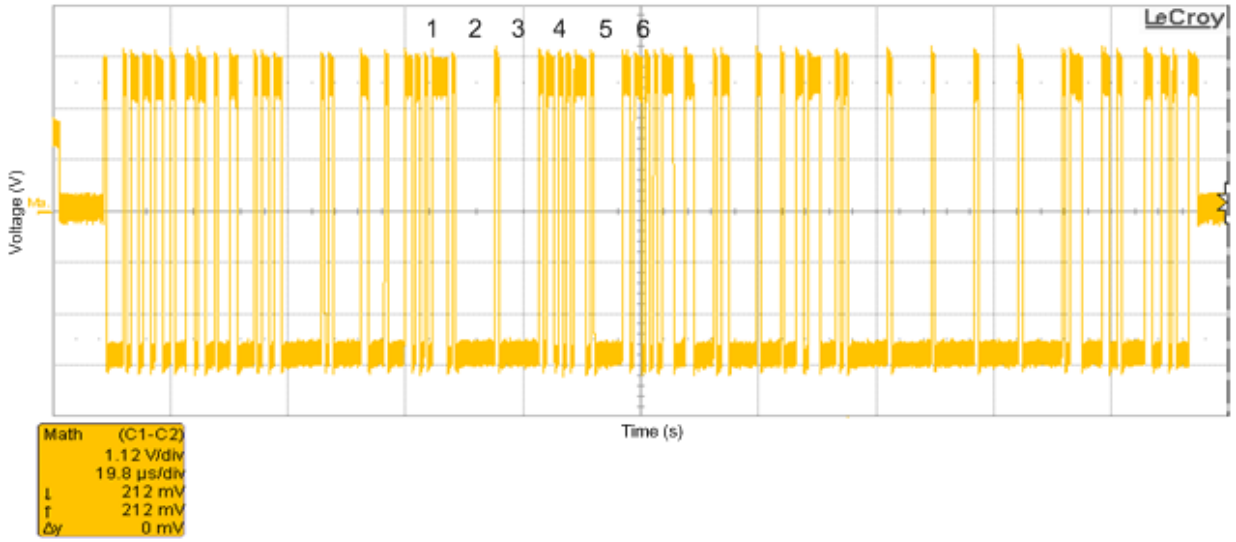


Figure 7.22 – Receiving data from PLC to delevitate the RDS

Table 7.19 – Transmitting data to FPGA while PLC is in control (RDS is delevitated)

Data position	Binary value	Indication
1	11110101	Levitated = false
2	00000000	Open
3	00000000	GUI control = false
4	10101011	PLC control = true
5	00000000	Open
6	10101011	Delevitated = true

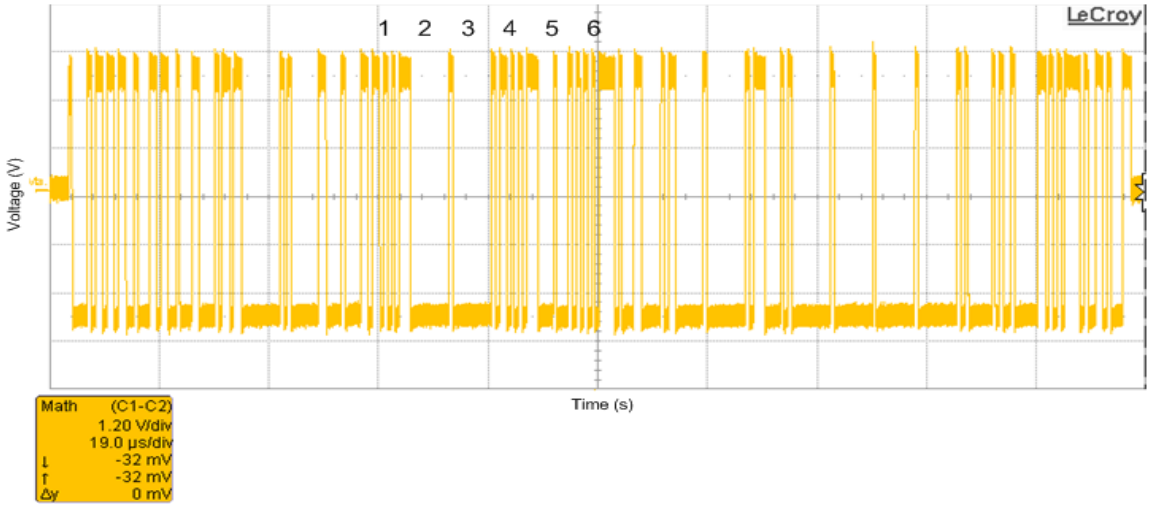


Figure 7.23 – Receiving data from PLC when GUI has operational control

Table 7.20 – Transmitting data to FPGA while GUI is in control

Data position	Binary value	Indication
1	10101011	not applicable
2	00000000	Open
3	00000000	GUI control = true
4	10101011	PLC control = false
5	00000000	Open
6	11110101	Not applicable

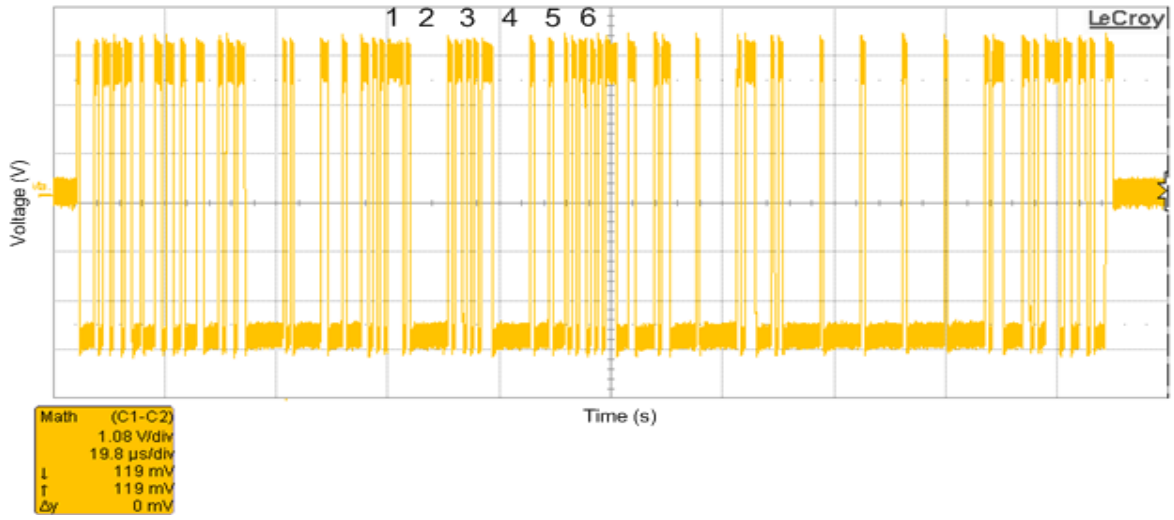


Figure 7.24 – Receiving data from PLC to levitate the RDS

Figure 7.24 shows the data received from the PLC which indicates that the PLC is no longer in control but the graphical user interface has operational control of the RDS. The data values in figure 7.24 that are received from the PLC are listed in tables 7.21.

Table 7.21 – Transmitting data to FPGA while PLC is in control (RDS is levitated)

Data position	Binary value	Indication
1	11110101	Levitated = true
2	00000000	Open
3	00000000	GUI control = false
4	10101011	PLC control = true
5	00000000	Open
6	10101011	Delevitated = false

7.6 Developed protocol vs. commercial PMC module

The verification and validation done in this chapter has shown that the PROFIBUS protocol that was developed and implemented on the Xilinx® FPGA conforms to the PROFIBUS DP standard. In order to determine the effectiveness of the developed PROFIBUS protocol; a comparison will be made with the PROFIBUS PMC model that was used in chapter 4 as the original PROFIBUS master device. This will help determine if the development of the PROFIBUS protocol was feasible and whether it could compete with the commercial PMC module.

To compare the commercial PROFIBUS card with the developed protocol, features that relate to both implementation methods is compared. The following characteristics are commonly used to compare different Fieldbus protocols and will be used for this comparison [27] [30]:

- Cost
- Throughput
- Data coding efficiency
- Medium access efficiency
- Responsiveness
- Cyclic times
- Bit error rate

Cost

Table 7.22 displays the cost involved with the two types of implementation for the PROFIBUS standard. Although the cost of the developed PROFIBUS core is much less than the commercial PMC module, the developed PROFIBUS protocol was implemented on the Xilinx® Virtex©-5 FPGA which was already available in the active magnetic drive electronic system (ADES). The cost comparison of table 7.22 therefore relates directly to the ADES.

Table 7.22 – Cost comparison

PMC 253	**Cost**
PMC253 Advanced PROFIBUS controller card	R 8500 + VAT
System configuration software for PROFIBUS networks for Windows 95/98/NT	R 6006 + VAT
Total	**R 14506 + VAT**
Developed PROFIBUS core	**Cost**
RS 485 differential drivers, circuit board, resistors, connectors	< R 200
Total	**± R 200**

Throughput

Throughput was defined earlier as the user data rate and is related to the networks nominal bit rate by the communication efficiency. Throughput can be mathematically represented by (7.10) [27].

$$U^{(\max)} = \eta_{dc}.\eta_{ma}.R \tag{7.10}$$

In (7.10), *R* is the nominal bit rate, η_{dc} is the efficiency of the data coding scheme, with η_{ma} representing the MAC efficiency.

Data coding efficiency η_{dc}

The quotient between the size of the user data that is transmitted in one transaction and the time needed to execute the service is called the data coding efficiency of a data link. This is represented in (7.11) [27].

$$\eta_{dc} = \frac{8d}{T_{ms}} \tag{7.11}$$

d represents the number of user data octets and T_{ms} is the message service time or message cycle time in *Tbits* (Tbits are discussed in chapter 5). T_{ms} is seen as the time needed for the completion of one read or write message. This time is composed of several transmission services when the number of data octets, *d,* is higher than the maximum number of data octets per frame that is allowed by the protocol [27].

Medium access efficiency η_{ma}

Medium access efficiency can be defined as the time that is not wasted by the stations in the management of the bus. This will differ for each protocol, based on their method of MAC. In the case of PROFIBUS where a token scheme is used, an amount of overhead exists and the medium access efficiency is represented by (7.12) [27].

$$\eta_{ma} = \frac{T_c - N_M T_{TC}}{T_c} \tag{7.12}$$

N_M is the number of master stations, with T_c equal to the mean cycle time of the network and T_{TC}, the token time [27].

Responsiveness

Responsiveness refers to the system response time T_R. Responsiveness is therefore the maximum time allowed between a high priority request being made and the time when the first bit of that frame is being transmitted. This function is also dependent on whether a protocol uses a token passing scheme or not [27].

Cyclic times

Cyclic times differ for each protocol implemented on a communication system. Basically the cycle time of a protocol is defined as the time it takes for one telegram cycle to complete. The cyclic times are calculated in appendix D by evaluating the different message waveforms. As an example of cyclic time calculation, consider the PROFIBUS DP standard in (7.13) [16].

$$T_{MC} = \left(\left(T_{S/R} + T_{SDR} + T_{A/B}\right) * T_{TD}\right) + T_{ID} \tag{7.13}$$

with:

T_{MC} - Time of one telegram cycle

$T_{S/R}$ - $a \times 11Tbit,$ where $a = \text{No. of frame characters in Send/Request frame},$

T_{SDR} - The reaction time of the slave to respond to a message

T_{TD} - Maximum time interval that occurs on the transmission medium between a transmitter and receiver when a frame is transmitted

T_{ID} - Amount of time the initiator must wait before the next telegram can be sent

$T_{A/B} = b \times 11Tbit,$ where $b = \text{No. of frame characters in Acknowledge/Response frame}$

Bit error rate

Bit error rate (*P*) is the number of erroneous bits in relation to the number of all the transmitted bits in a communication system. This simple equation is given by (7.14) [30].

$$P = \frac{\text{number of erronous bits}}{\text{number of all transmitted bits}} \tag{7.14}$$

The worst case scenario would have a bit error rate of 0.5, meaning every second bit is wrong. Practical implementation as low as $P = 10^{-4}$ is feasible. A twisted-pair telephone cable has a bit error probability of 10^{-5}. Error detection mechanisms are used to detect these errors, but sometimes not all errors are captured. The residual error rate (R_e) is defined in (7.15) [30].

$$R_e = \frac{\text{number of undetected erronous bits}}{\text{number of all transmitted bits}} \tag{7.15}$$

The integrity of an error detection method can be calculated as the relation between P and R_e. Integrity class 1 is used for cyclic data exchange; class 2 for event driven communication; and class 3 for remote control systems [30].

In figure 7.15, it was observed that no transmission errors were seen on the developed PROFIBUS channel. Equation (7.14) and (7.15) can therefore not be used for the BER evaluation. Statistical analysis can be used, but since the developed PROFIBUS protocol and the commercial PMC module has the same protocol properties, this would be beneficial. An alternative method will be used for this comparison. SNR and BER are closely related to each other, since a high SNR will ensure a low BER [46]. The SNR of the developed protocol and the PMC 253 module is therefore compared to analyze the stability of the protocol implementations. All of the above calculations are listed in appendix D, with the results displayed in table 7.23.

Table 7.23 – PROFIBUS comparison results

Comparison	**Developed PROFIBUS protocol**	**PROFIBUS PMC 253 module**
Cost	**R 200**	R 14506 + VAT
Throughput	**700.5 kbps**	694.5 kbps
Data coding efficiency	**0.467**	0.463
Medium access efficiency	1 (only one master present)	1 (only one master present)
Responsiveness	14.77 us	**7.32 us**
Cyclic times	381.17 µs	**375.47 µs**
Signal-to-noise rate	32.87 dB	**34.46 dB**

In table 7.23, the bold results indicate the implementation method that performed better with regards to the characteristic that was evaluated. It is seen that the developed PROFIBUS protocol not only competes well, but outperforms the PMC 253 module with regards to cost, throughput and data coding efficiency. Except for the cost, the other characteristics relates closely to each other because of the same PROFIBUS DP protocol that was implemented.

7.7 Conclusion

The aim of this chapter was to validate the design and implementation of the PROFIBUS protocol on a Xilinx® FPGA. The developed protocol was also compared with a commercial off-the-shelf PROFIBUS device to test the feasibility and quality of the developed protocol. Comprehensive tests were done to validate the implementation of the physical and data link layer layers in the real-time environment. Sufficient results were found and provided adequate evidence for the accurate design and implementation of the PROFIBUS protocol on the FPGA.

After the physical layer and data link layer were validated, the ADES was evaluated with the implemented VHDL-based PROFIBUS protocol. It was found that the developed protocol has the same capabilities as the commercial PMC 253 module that was previously used. This reduced the cost of the complete ADES that was originally designed for commercial development. The chapter concluded by comparing the developed PROFIBUS protocol against the commercial off-the-shelf PMC 253 PROFIBUS module. This was done to evaluate the feasibility of the developed protocol and determine if the developed protocol can compete with its commercial counterpart. It was found that the developed protocol has all of the capabilities of its commercial counterpart with the added benefit of a much lower implementation cost.

Chapter 8

Conclusions and recommendations

Chapter 8 starts off by briefly discussing the requirements for the VHDL-based PROFIBUS protocol for the ADES as well as the process followed to achieve these requirements. An analysis of the developed PROFIBUS protocol is presented with regards to the results found in the previous chapter. The chapter concludes by presenting possible future work and commercial application possibilities.

8.1 PROFIBUS protocol design for the ADES

The active magnetic bearing drive-electronic system (ADES) was developed to be a robust, cost effective industrial standard controller for active magnetic bearing systems. Communication between various systems and devices were developed and implemented based on the requirements of the ADES. To communicate between a Siemens® PLC and the main controller of the ADES, the Fieldbus profile, PROFIBUS DP, was chosen based on its reputation of being a rugged open bus system that can guarantee problem-free communication. This profile was implemented by means of a PROFIBUS PMC module and established a communication link between the PLC and the main controller to communicate control information between the two systems.

After the PROFIBUS network was established, the focus shifted towards implementing the PROFIBUS protocol on the Xilinx® FPGA. This was done to reduce the cost of the ADES while maintaining the PROFIBUS communication network. The implementation of the PROFIBUS protocol on the FPGA allowed for the removal of the PROFIBUS PMC 253 module and therefore the reduction in cost. In order to implement the PROFIBUS protocol on the FPGA, the PROFIBUS standard was analyzed, designed in VHDL and instantiated on the FPGA to execute in parallel with the various other control and communication systems that are instantiated on the FPGA. After the implementation, the PLC was directly connected to the Xilinx® FPGA and a new PROFIBUS network was established.

This dissertation started by defining the requirements for the ADES as well as the PROFIBUS network. A critical literature study was conducted to evaluate the applicable literature with special focus on the PROFIBUS DP standard. After the literature study, the ADES environment and the implementation of the PROFIBUS PMC module in the ADES were discussed to familiarize the reader with the implementation platform. Once these issues were discussed, the focus shifted towards the design and implementation of the PROFIBUS DP protocol on the Xilinx® FPGA. To evaluate the developed PROFIBUS protocol, intensive verification and validation tests were performed on the designed and implemented PROFIBUS DP protocol. The developed protocol was then evaluated against the commercial off-the-shelf PMC 253 PROFIBUS module which was originally used for PROFIBUS communication. This comparison showed that the developed protocol not only competes, but also outperforms the commercial PMC 253 module in various aspects. The difference in cost between the developed PROFIBUS protocol and the commercial PMC 253 module reinforced the initial requirement to reduce the cost of the system.

8.2 Analysis of the developed PROFIBUS DP protocol

In order to establish a PROFIBUS network between the PLC and the developed protocol instantiated on the FPGA, the developed protocol needed to meet all the mandatory specifications of the PROFIBUS DP standard defined in the IEC 61158 and IEC 61784. The EM 277 on the PLC is a standardized PROFIBUS slave device. In order to communicate with this device, the PROFIBUS DP protocol had to be designed and implemented exactly as specified.

The evaluation performed with regards to the physical layer was done to ensure that the communication channel suited the PROFIBUS protocol, as well as to determine probable interferences during communication. The physical layer of the developed PROFIBUS protocol was analyzed with the construction of eye diagrams and determining the properties of the serial communication signal. Low noise and jitter were detected while the ADES was operational, which suggest that bit errors will not be caused as a result of this. The implemented physical layer complies not only with regards to sufficient data transfer, but fully meets the requirements of the PROFIBUS DP standard.

The PROFIBUS type A cable was designed by PROFIBUS Nutzerorganization to ensure a rugged bus for PROFIBUS networks. The calculated signal-to-noise ratio (SNR) provided the evidence to support this statement. A SNR of 32.87 dB was measured and calculated while the ADES was fully

operational. This suggests that the cable may easily be extended to the specified 100 m (without repeaters) and still ensure error free communication.

The next step in evaluation was the verification of the data link layer of both the designed and implemented PROFIBUS DP protocol. Every aspect of the data link layer that was implemented was individually evaluated. It was found that the developed PROFIBUS protocol met all the requirements of the PROFIBUS standard. Results from simulations and implementation measurements provided the evidence for the security standard and error free communication ensured by PROFIBUS. In simulation, the developed protocol's response to data errors was shown. This could not be shown during implementation as a result of the non-existence of errors occurring during transmission. This is not a strange observation, since both devices that communicate on the PROFIBUS network conforms to the PROFIBUS DP standard.

Once the data link layer was evaluated, the implementation on the Xilinx® FPGA was evaluated together with the functionality of the developed PROFIBUS network. The developed PROFIBUS protocol met all the requirements of the ADES and allowed the same functionality as the original PMC 253 PROFIBUS module that was used. Control and measurement data is communicated between various devices of the ADES and the Siemens® PLC, using the instantiated PROFIBUS protocol on the FPGA.

To complete the study, the developed PROFIBUS protocol was compared with the commercial off-the-shelf PROFIBUS PMC 253 module. The comparison was based on common measurements used to compare Fieldbus devices. This included cost, throughput, data coding efficiency, medium access efficiency, responsiveness, cyclic times and bit error rates. It was found that the developed PROFIBUS protocol not only compares well, but outperforms the commercial PMC 253 module in various aspects.

The largest difference in comparison is the cost related to both devices. The cost of implementing the PROFIBUS protocol on an FPGA is approximately 1% of the commercial PROFIBUS PMC 253 module. The reduction in cost was one of the main requirements for the PROFIBUS implementation on the FPGA for the ADES. The comparison done at the end of chapter 7 showed that the developed protocol and the commercial PMC 253 protocol closely relate to each other with regards to throughput, data coding efficiency, responsiveness, cyclic times and signal-to-noise ratio. The developed VHDL-based protocol achieved a better throughput and data coding efficiency than the

commercial PMC module, while the commercial PMC module achieved a better cyclic time and SNR. These values do however relate so closely as a result of the same protocol implementation.

The higher SNR of the commercial PMC module is due to the fact that the PROFIBUS channel is better isolated with this implementation. The VHDL-based PROFIBUS protocol has more noise interference because of the external RS 485 driver circuit that was implemented. Since this driver is not protected by the cable shielding of the PROFIBUS Type A cable, the noise is higher in this implementation.

8.3 Recommendations for future work

In conclusion, recommendations for future work to be done with regards to the PROFIBUS standard as well as the ADES implementation are discussed in this section. The ADES improvements discuss how the ADES can be improved with the expansion of the PROFIBUS network. The final section focuses on the commercial opportunities of the developed PROFIBUS protocol.

8.3.1 ADES improvements

As discussed, the ADES can be controlled with either the PLC or graphical user interface of the single-board computer. Currently the PROFIBUS network is only used to communicate control data between the FPGA and PLC. The PROFIBUS standard allows for up to 244 bytes of user data to be transmitted in the data exchange state. It can therefore be said that the PROFIBUS network is not nearly used to its full capacity. This can be improved by transmitting the power amplifier currents and position sensor readings over the PROFIBUS network to the PLC. The PLC user interface can then be expanded to create the same graphical user interface that is connected to the single-board computer of the ADES. This will allow operators of the ADES to have a more visual interpretation of the state of the system.

The next step in improving the ADES would be the complete Fieldbus integration of the RDS devices. The motor drive used to control the operational speed of the rotor delevitating system (RDS) is currently operated separately, and requires that more than one person is needed to operate the RDS and ADES. The motor drive can be set up as a PROFIBUS slave device, therefore allowing the PROFIBUS protocol implementation on the FPGA to control the motor drive over the PROFIBUS network. This will require the addition of a token frame in the PROFIBUS network to exercise correct medium access control.

Another method for improving the ADES should include the implementation of the popular Ethernet communication network on the system. This will not cause removal of the PROFIBUS network, but will allow the implementation of full remote operation or maintenance of the ADES. Should the ADES together with the RDS be implemented in the industry, the developers of the ADES can gain access to the ADES unit if maintenance or support is needed.

8.3.2 PROFIBUS certification and commercial applications

The PROFIBUS protocol implementation on the Xilinx® FPGA was developed to reduce the cost of the ADES and was not intended for commercial use. The evaluation of the developed PROFIBUS protocol and the comparison against the commercial off-the-shelf PMC 253 PROFIBUS module produced evidence for the commercial feasibility of this developed protocol for FPGA implementation. In order to commercialize the VHDL-based PROFIBUS protocol, the developed PROFIBUS protocol needs to be certified by the PROFIBUS Nutzerorganization.

In order to certify the VHDL-based PROFIBUS protocol, a few features need to be added to the existing protocol implementation. In the ADES only one master and slave were used on the PROFIBUS network and there was no need for the token-based scheme of PROFIBUS. This function will have to be added to the current design. The token based scheme is merely an implementation of an extra PROFIBUS telegram frame in the current state machine. This token is used to grant access to the PROFIBUS network to all the masters connected on the bus. The current protocol implementation can be slightly altered to meet this specification.

Once the token-based scheme is implemented in the VHDL-based protocol, the design can be certified for commercial use. The certification by the PROFIBUS Nutzerorginization has a high initial cost, but once the design is certified the product can be sold commercially with the PROFIBUS stamp of approval. This product will produce a VHDL-based PROFIBUS protocol which can be implemented on any Xilinx® FPGA. FPGA users will greatly benefit from this, since nearly no additional cost will be needed to implement a PROFIBUS network for their specific implementation. This will also give developers the possibility to alter the design according to their specific needs without acquiring the additional software needed for this in PMC module implementations.

8.4 Closure

The aim of this project was to design and implement the PROFIBUS DP protocol on a Xilinx® FPGA to establish a PROFIBUS network between a FPGA and a PLC device, while reducing the cost of the ADES. The protocol needed to match all the specifications of the PROFIBUS DP standard as well as the requirements of the ADES. In addition, the feasibility and functionality of the developed PROFIBUS protocol was to be compared with the commercial off-the-shelf PROFIBUS PMC module.

From the results it is shown that the developed PROFIBUS protocol met all the specifications and requirements of the PROFIBUS DP standard specified in the IEC 61158 and IEC 61784. The Fieldbus network requirements for the ADES were also met and the communication network was established between the PLC and the FPGA for various control communications. Furthermore the developed PROFIBUS protocol not only competed well with the commercial PROFIBUS PMC module but outperformed the device with regards to throughput and data coding efficiency. The cost comparison showed that the VHDL-based PROFIBUS protocol has an implementation cost of approximately 1% to that of the commercial PMC module. This project laid the foundations for a PROFIBUS implementation on the FPGA and can easily be extended for research or commercial use.

REFERENCES

[1] C. R. Knospe, "Active magnetic bearings for machining applications," *Control Engineering Practice*, vol. 15, no. 3, pp. 307-313, Mar. 2007.

[2] G. Schwietzer, H. Bleuler, and A. Traxler, *Active Magnetic Bearings: Basics, Properties and Applications of Active Magnetic Bearings*. Zurich, Switzerland: Authors Working Group, 2003.

[3] S. Ion, D. Nicholls, R. Matzie, and D. Matzner, "Pebble Bed Modular Reactor - The First Generation IV Reactor To Be Constructed," in *World Nuclear Assosiation*, London, 2003, pp. 1-14.

[4] International Engineering Consortium. (2009, Aug.) Compact PCI. [Online]. http://www.iec.org

[5] A. DeHon, "The Density Advantage of Configurable Computing," in *IEEE Computer Society*, Pasadena, 2000, pp. 41-49.

[6] V. George, "Low Energy Field-Programmable Gate Array," UNIVERSITY OF CALIFORNIA, Dissertation, 2000.

[7] J. Irvine and D. Harle, *Data Communications and Networks: An Engineering Approach*, 1st ed. Chichester, England: Jonh Wiley & Sons, Ltd, 2002.

[8] D. E. Nordell, "Communication Systems for Distribution Automation," in *IEEE/PES Transmission and Distribution Conference and Exposition*, Chicago, 2008, pp. 1-14.

[9] P. Neumann, "Communication in industrial automation—What is going on?," *Control Engineering Practice*, vol. 15, p. 1332–1347, Dec. 2006.

[10] PROFIBUS Nutzerorganization, "PROFIBUS - Technology and Application," System Description 4.002, Oct. 2002.

[11] R. Patzke, "Fieldbus basics," *Computer Standards & Interfaces*, vol. 19, no. 5-6, pp. 275-293, Oct. 1998.

[12] J. Thomesse, "Fieldbus technology in industrial Automation," *Proceedings of the IEEE* , vol. 93, no. 6, pp. 1073-11101, Jun. 2005.

[13] J. Thomesse, "A Review of the Fieldbuses," *Annual reviews in Control*, vol. 22, pp. 35-45, Nov. 1998.

[14] C. J. HOSSACK and C. G. GUY, "The Route From VHDL to FPGA Using Synthesis," *IEE Colloquium on Software Support and CAD Techniques for FPGAs*, pp. 811-814, Apr. 1994.

[15] V. A. Pedroni, *Circuit Design with VHDL*. Massachusetts, USA: MIT Press, 2004.

[16] Acromac Incorporated, "Introduction to PROFIBUS DP," Acromac Incorporated Technical Reference 8500-698-A02M000, 2002.

[17] T. H. D. L. Ulf Nordqvist, "CRC Generation for Protocol Processing," pp. 1-6.

[18] L. Y.-b. CHEN Shi-yi, "Error Correcting Cyclic Redundancy Checks based on Confidence Declaration," in *6th International Conference on ITS Telecommunications Proceedings*, Chengdu, 2006, pp. 511-514.

[19] International Electrotechnical Commission, "Overview and guidance for the IEC 61158 series," IEC, Ceneva, Technical report 2-8318-6966-8, 2003.

[20] R. Murugesan, "EVOLUTION OF FIELDBUS TECHNOLOGY," *PEJ Online Article*, pp. 2-3, Nov. 2008.

[21] M. Felser and T. Sauter, "The fieldbus War: History or Short Break Between Battles?," *IEEE International Workshop on Factory Communication Systems*, no. 4, pp. 73-80, May 2005.

[22] S. G. Park, "Fieldbus in IEC 61158 Standard," in *15th CISL Winter Workshop*, Kushu, Japan, 2002, pp. 1-3.

[23] G. Cena, C. Demartini, and A. Valenzano, "ON THE PERFORMANCES OF TWO POPULAR FIELDBUSES," in *IEEE International Workshop on Factory Communication Systems, 1997. Proceedings*, Barcelona, 1997, pp. 177-188.

[24] D. M. Elshafei. (2007, Oct.) Dr. Moustafa Elshafei homepage. [Online]. http://www.ccse.kfupm.edu.sa/~elshafei/elshafei_ch6_v4.pdf

[25] P. Pinceti, "Fieldbus: More Than a Communication Link," *IEEE Instrumentation & Measurement Magazine*, vol. 7, no. 1, pp. 17-23, Mar. 2004.

[26] PROFIBUS Nutzerorganization, "PROFIBUS Specification: Normative Parts of PROFIBUS - FMS, -DP, -PA according to the European Standard," Specification EN 50170 Volume 2, 1998.

[27] M. D. Rubio Benito, J. M. Fuertes, K. E., and N. Perez Arzoz, "PERFORMANCE EVALUATION OF FOUR FIELDBUSES," in *7th IEEE International Conference on Emerging Technologies and Factory Automation*, Barcelona, 1999, pp. 881-890.

[28] Synergetic Micro Systems. (2001, Sep.) Fieldbus comparison chart. [Online]. http://www.itk.ntnu.no/fag/TTK4545/TTK2/PDF/Message_10.pdf

[29] M. M. Dennis Derickson, *Digital Communications Test and Measurement*, 1st ed., M. M. Dennis Derickson, Ed. Boston, USA: Prentice Hall, 2008.

[30] M. Felser, "Quality of Profibus Installations," *IEEE International Workshop on Factory Communication Systems*, pp. 113-118, Sep. 2006.

[31] Siemens IK PI. (2000, Oct.) PROFIBUS acc. to IEC 61158/EN 50170. Document.

[32] PROFIBUS User Organization. (2010, May) PROFIBUS Overview. [Online]. http://www.profibus.com/technology/profibus/overview/

[33] J. Weigmann and G. Kilian, *Decentralization with PROFIBUS DP/DPV1*, 2nd ed. John Wiley & Sons, 2003.

[34] J. J. v. Rensburg, "Controller design concept for the ADES," North-West University Specifications, 2009.

[35] Concurrent Technologies Inc, "CompactPCI," Specification 1502/0906, 2006.

[36] TEWS TECHNOLOGIES, "PMC Modules," Datasheet, 2008.

[37] A. C. Niemann, "Functional diagram of inductive sensor (ADES)," North-West University Internal document ADES-01-12400-116-02, 2009.

[38] Siemens AG, *SIMATIC S7-200 Programmable Controller System Manual*, 9th ed. Nuernberg, Germany: Siemens AG, 2007.

[39] Kontron Embedded Computers, "PMC253 Advanced PROFIBUS Controller for Fieldbus Applications," Datasheet 24084 DS-02/2002, 2002.

[40] R. A. Mueller, "Firmware Engineering: The interaction of Microprogramming and Software Technology," *IEEE Software*, vol. 3, no. 4, pp. 4-5, Jul. 1986.

[41] S. Davidson and B. D. Shriver, "An Overview of Fimware Engineering," *IEEE Computer Society*, vol. 11, no. 5, pp. 21-33, May 1987.

[42] J. Norhuzaimin and H. M. Maimum, "The Design of High Speed UART," in *Asia-Pacific conference on applied electromagnetics proceedings*, Malaysia, 2005, pp. 306-310.

[43] M. Y. I. Idris and M. Yaacob, "A VHDL Implementation of BIST Technique in UART Design," in *Conference on Convergent Technologies for Asia-Pacific Region*, 2003, pp. 1450-1454.

[44] D. R. Walace and R. U. Fujii, "Software verification and validation: An overview," pp. 264-268, May 1989.

[45] P. Pierce, "Software verification and validation," in *IEEE Technical Applications Conference*, Seattle, Nov. 1996, pp. 265-268.

[46] F. Yongquan and Z. Zilic, "BER Testing of Communication Interfaces," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 5, pp. 897-906, May 2008.

[47] Profibus. (2002, Oct.) Profibus Technology and Application. [Online]. http://www.profibus.com/pall/meta/downloads/

[48] Softing. (2007, Nov.) FPGA-based Hardware Integration for Field Devices – Chip Solution. [Online]. http://www.profibus.com/member/softing_north_america/products/article/01299/

[49] Siemens AG, "PROFIBUS Technical," Siemens Automation Brochure E86060-A4678-A171-A5-7600, 2008.

[50] PROFIBUS Nutzerorganization, "GSD-Specification for PROFIBUS-DP," Specification 2.122, 1998.

[51] Siemens IK PI. (2000, Oct.) PROFIBUS acc. to IEC 61158/EN 50170. [Online]. www.sea.siemens.com/autogen/docs/net/pfb/lit/Profibus%20from%20IKPI%202000.pdf

[52] J. A. Carvalho, A. S. Carvalho, and P. Portugal, "Assessment of Profibus networks using a fault injection framework," in *10th IEEE Conference on Emerging Technologies and Factory Automation*, Catania, 2005, p. 415423.

[53] M. A. Dominguez, P. Marino, J. B. Nogueira, C. A. Siguenza, and F. Poza, "The PROFIBUS formal specification: a comparison between two FDTs," in *Computer Networks*, J. Quemada, Ed. Vigo, Spain: Elsevier, Apr. 2001, ch. 18, pp. 345-362.

[54] M. Felser, "The Fieldbus Standards: History and Structures," pp. 1-5, 2002.

Appendix A – Photos of ADES and RDS system

Figure A.1 – Rotor delevitating system (RDS)

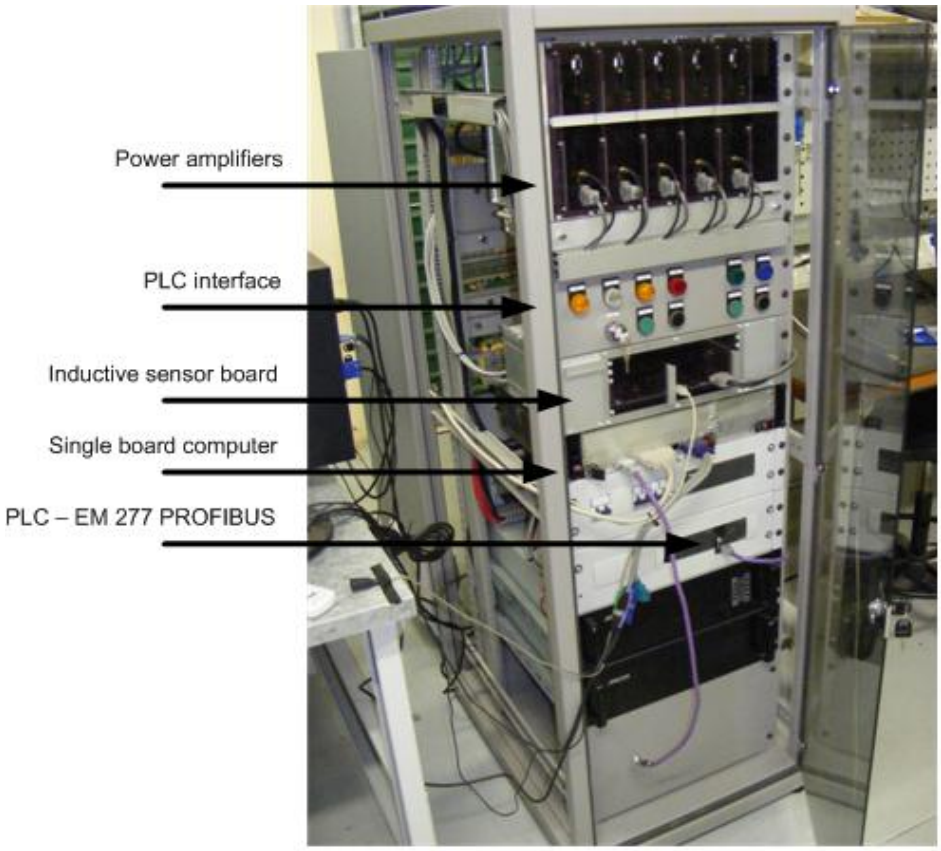


Figure A.2 - ADES

Figure A.3 – RDS motor drive

Figure A.4 – ADES connections

Figure A.5 – PLC interface

Figure A.6 – PLC PROFIBUS interface

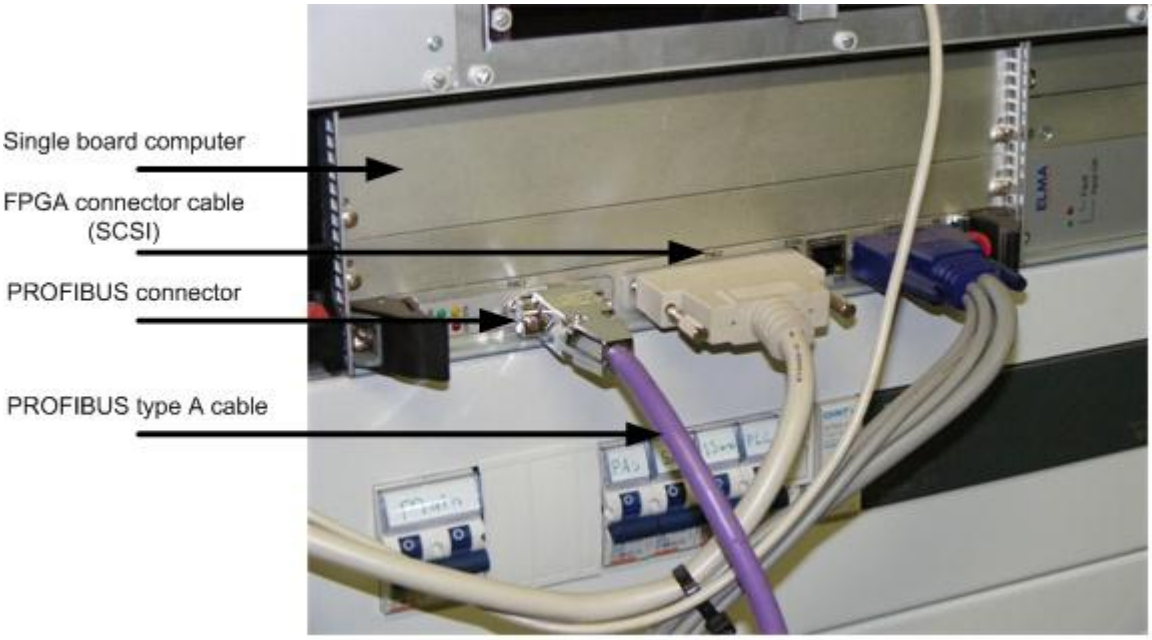


Figure A.7 – Single-board computer

Appendix B – Data CD

Appendix B.1: ADES requirements specifications

System requirements specifications for the active magnetic bearing drive electronic system

Appendix B.2: VHDL code

VHDL code

Appendix B.3: MATLAB® Code

MATLAB® code

Appendix B.4: Hardware specifications

Applicable hardware datasheets

Appendix B.5: IEC 61158 standard

IEC 61158 standard (1998)

Appendix B.6: Photos

ADES and RDS photos

Appendix B.7: Documentation

Digital version of this dissertation

Appendix C – PLC slave

Appendix C.1 – EM 277 GSD file

A GSD file is a PROFIBUS standard used to define the slave devices according to the possible parameter settings. This file indicates the parameter and configuration options available for the specific slave device. The following GSD file specifies the EM 277 PROFIBUS module that is used in this design as the slave device.

```
;=====================================================
; GSD File for the EM 277 PROFIBUS-DP with a DPC31
; MLFB  : 6ES7 277-0AA2.-0XA0
; DATE  : 19-May-2003
;=====================================================
#Profibus_DP
;General parameters
GSD_Revision          = 1
Vendor_Name           = "Siemens"
Model_Name            = "EM 277 PROFIBUS-DP"
Revision              = "V1.02"
Ident_Number          = 0x089D
Protocol_Ident        = 0
Station_Type          = 0
FMS_supp              = 0
Hardware_Release      = "1.00"
Software_Release      = "1.02"
9.6_supp              = 1
19.2_supp             = 1
45.45_supp          = 1
93.75_supp          = 1
187.5_supp          = 1
500_supp            = 1
1.5M_supp           = 1
3M_supp             = 1
6M_supp             = 1
12M_supp            = 1
MaxTsdr_9.6         = 60
MaxTsdr_19.2        = 60
MaxTsdr_45.45       = 250
MaxTsdr_93.75       = 60
MaxTsdr_187.5       = 60
MaxTsdr_500         = 100
MaxTsdr_1.5M        = 150
MaxTsdr_3M          = 250
MaxTsdr_6M          = 450
MaxTsdr_12M         = 800
Redundancy          = 0
Repeater_Ctrl_Sig   = 2
```

```
24V_Pins            = 2
Bitmap_Device       = "EM_277_N"
Bitmap_SF           = "EM_277_S"
; Slave-Specification:
OrderNumber="6ES7 277-0AA2.-0XA0"
Periphery="SIMATIC S5"
Slave_Family=10@TdF@SIMATIC

Freeze_Mode_supp    = 1
Sync_Mode_supp      = 1
Set_Slave_Add_Supp  = 0
Auto_Baud_supp      = 1
Min_Slave_Intervall = 1
Fail_Safe           = 0
Max_Diag_Data_Len   = 6
Modul_Offset        = 0
Modular_Station     = 1
Max_Module          = 1
Max_Input_len       = 128
Max_Output_len      = 128
Max_Data_len        = 256


; UserPrmData-Definition
ExtUserPrmData=1 "I/O Offset in the V-memory"
Unsigned16 0 0-10239
EndExtUserPrmData
; UserPrmData: Length and Preset:
User_Prm_Data_Len=3
User_Prm_Data= 0,0,0
Max_User_Prm_Data_Len=3
Ext_User_Prm_Data_Const(0)=0x00,0x00,0x00
Ext_User_Prm_Data_Ref(1)=1

; Module Definition List
Module = "2 Bytes Out/ 2 Bytes In     -" 0x31
EndModule
Module = "8 Bytes Out/ 8 Bytes In     -" 0x37
EndModule
Module = "32 Bytes Out/ 32 Bytes In   -" 0xC0,0x1F,0x1F
EndModule
Module = "64 Bytes Out/ 64 Bytes In   -" 0xC0,0x3F,0x3F
EndModule
Module = "1 Word Out/ 1 Word In       -" 0x70
EndModule
Module = "2 Word Out/ 2 Word In       -" 0x71
EndModule
Module = "4 Word Out/ 4 Word In       -" 0x73
EndModule
Module = "8 Word Out/ 8 Word In       -" 0x77
EndModule
Module = "16 Word Out/ 16 Word In     -" 0x7F
EndModule
Module = "32 Word Out/ 32 Word In     -" 0xC0,0x5F,0x5F
```

```
EndModule
Module = "2 Word Out/ 8 Word In        -" 0xC0,0x41,0x47
EndModule
Module = "4 Word Out/ 16 Word In       -" 0xC0,0x43,0x4F
EndModule
Module = "8 Word Out/ 32 Word In       -" 0xC0,0x47,0x5F
EndModule
Module = "8 Word Out/ 2 Word In        -" 0xC0,0x47,0x41
EndModule
Module = "16 Word Out/ 4 Word In       -" 0xC0,0x4F,0x43
EndModule
Module = "32 Word Out/ 8 Word In       -" 0xC0,0x5F,0x47
EndModule
Module = "4 Byte buffer I/O            -" 0xB3
EndModule
Module = "8 Byte buffer I/O            -" 0xB7
EndModule
Module = "12 Byte buffer I/O           -" 0xBB
EndModule
Module = "16 Byte buffer I/O           -" 0xBF
EndModule
```

Appendix C.2 – EM 277 programming

The Siemens® PLC is programmed with STEP 7© software. Visual methods are used to design how the input and output device connected to PLC should operate. Memory can be allocated together with logical functions to design the various operating procedures of the PLC. This section briefly present the designs of the local key, levitate and delevitate button which were used in this project.

Local key STEP 7© design

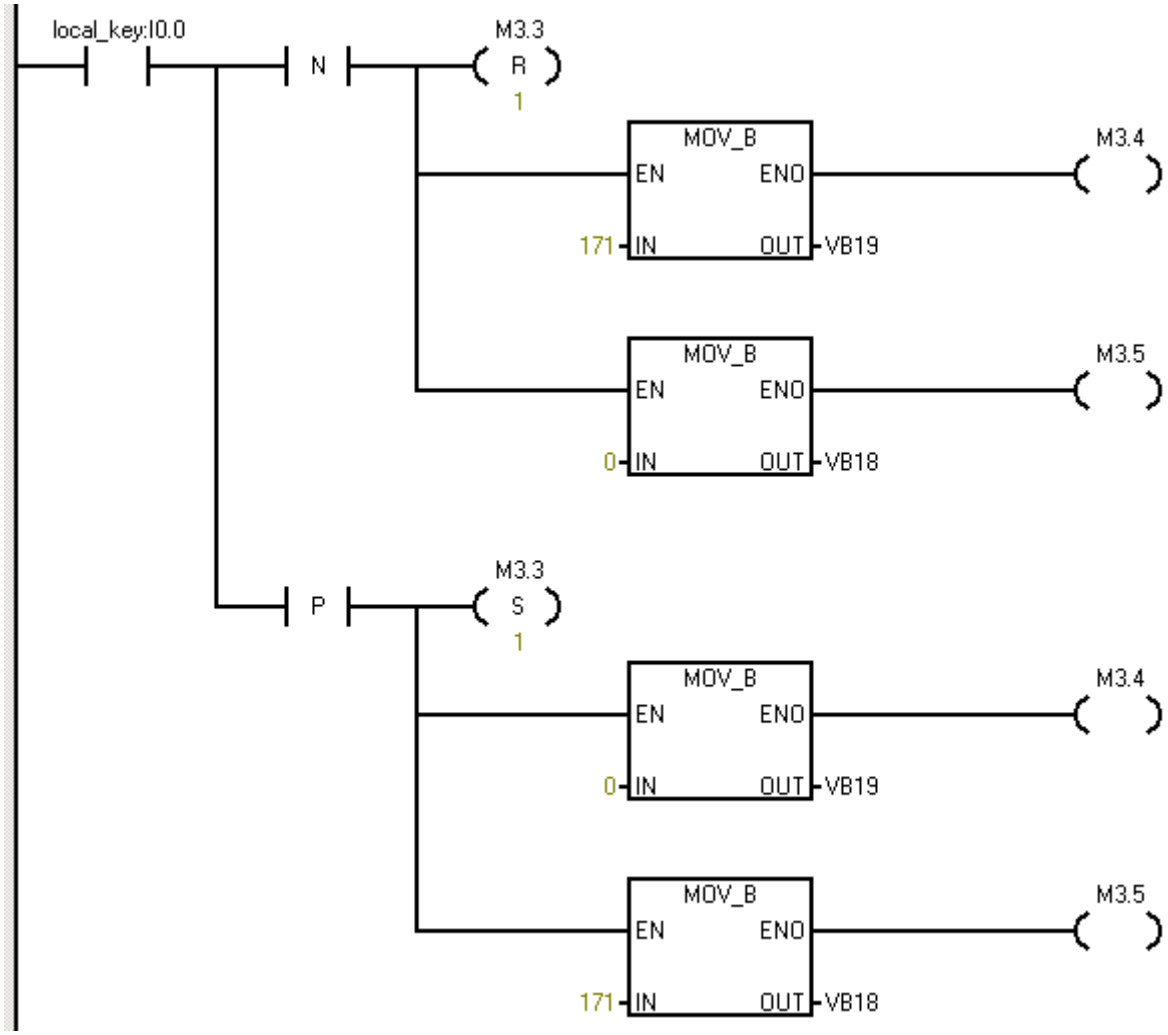


Figure C.1 – Local key STEP 7 design

Delevitate button STEP 7© design

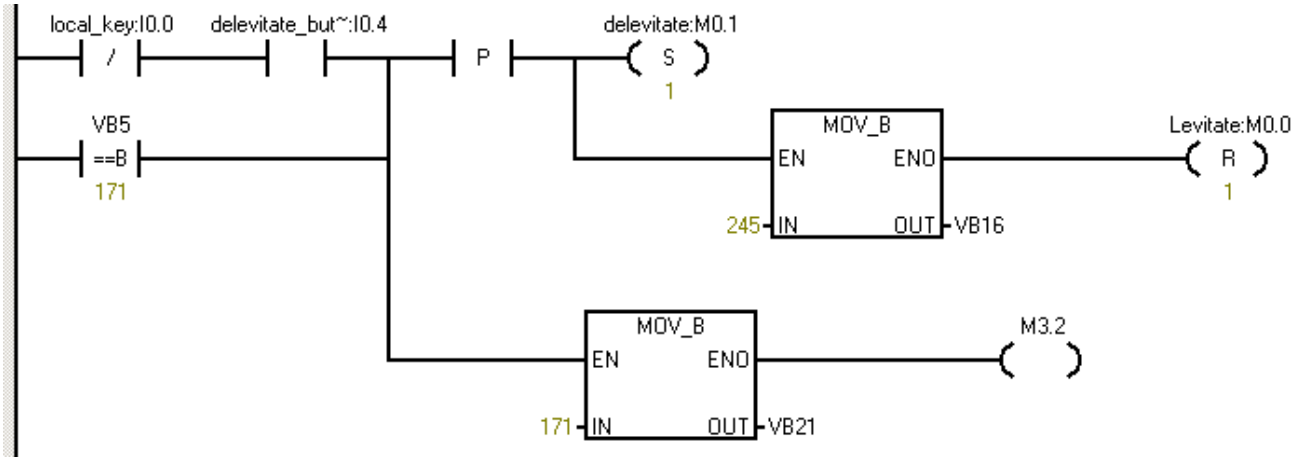


Figure C.2 – Delevitate STEP 7© design A

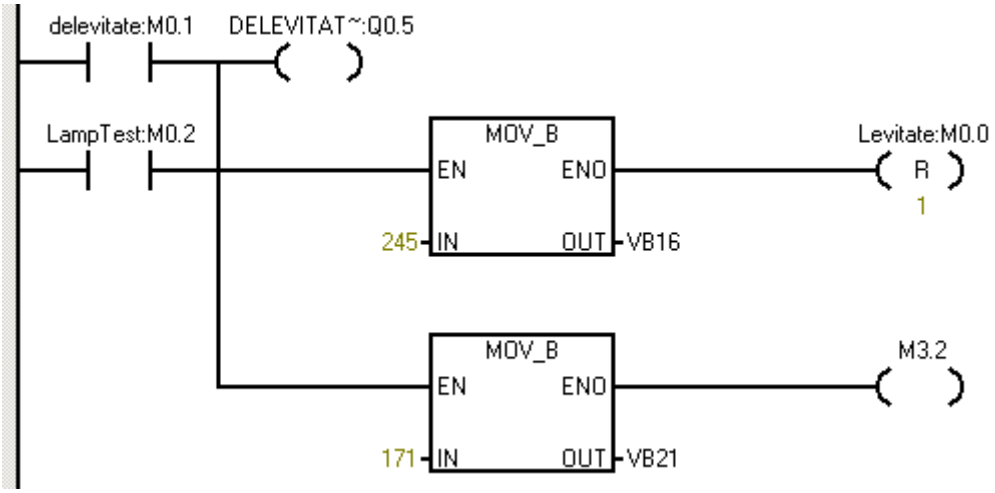


Figure C.3 – Delevitate button STEP 7© design B

Levitate button STEP 7© design

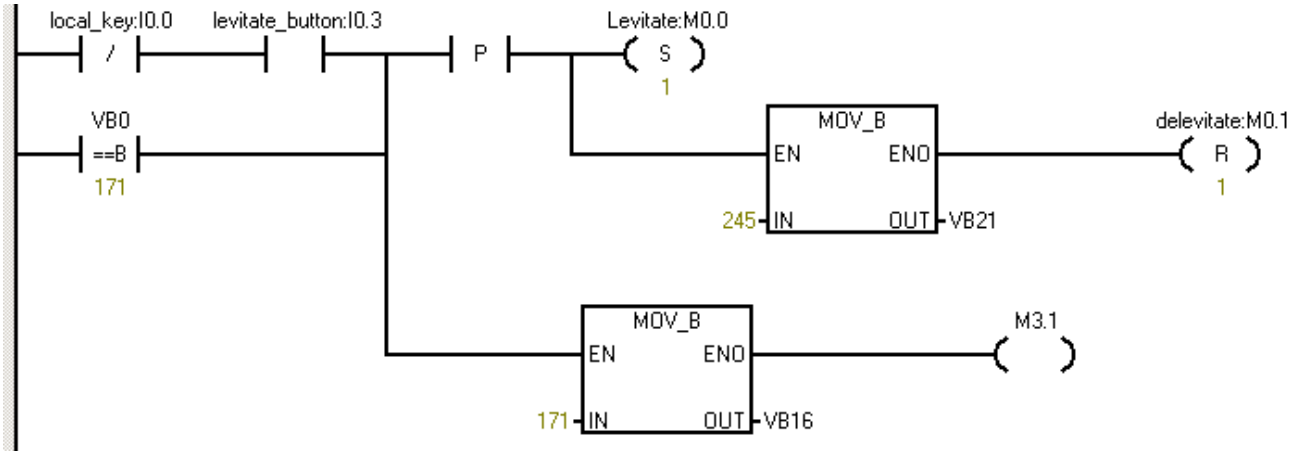


Figure C.4 – Levitate STEP 7© design A

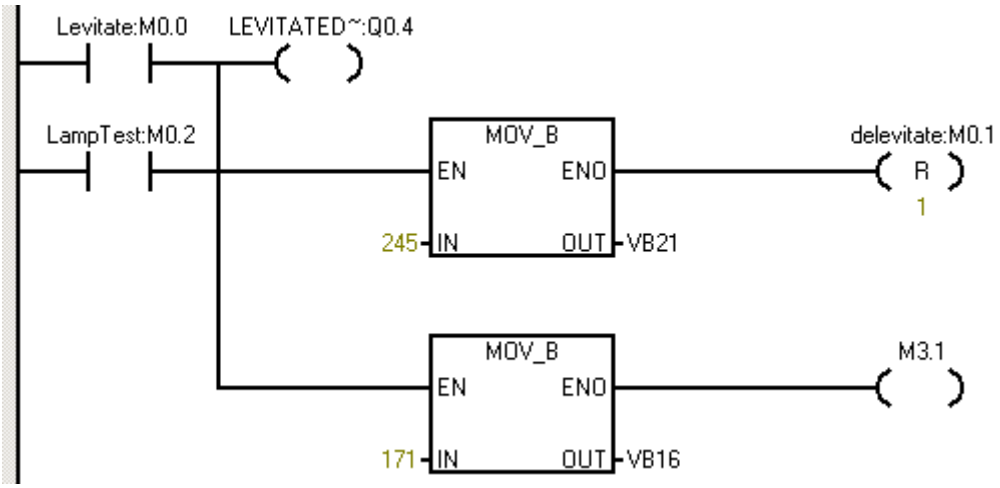


Figure C.5 – Delevitate STEP 7© design B

Appendix D – Mathematical calculations

Appendix D.1: Developed protocol vs. commercial PMC module calculations

This section presents the mathematical calculations used to compare the developed protocol against the commercial PMC module. The calculations is only presented in this section, but thoroughly discussed in chapter 7.6.

Table D.1 – Message service time calculation

Developed PROFIBUS DP protocol	**Commercial PMC PROFIBUS module**
$T_{ms} = \frac{T_{End} - T_{Begin}}{Tbit}$ $T_{ms} = \frac{0.000834974 - 0.000652623}{666\ ns}$ $T_{ms} = 273.8$	$T_{ms} = \frac{T_{End} - T_{Begin}}{Tbit}$ $T_{ms} = \frac{0.00022286 - 0.0000387517}{tBIT}$ $T_{ms} = 276.239$

Table D.2 – Data coding efficiency calculation

Developed PROFIBUS DP protocol	**Commercial PMC PROFIBUS module**
$\eta_{dc} = \frac{8d}{T_{ms}}$ $= \frac{8 \times 16}{273.8}$ $= 0.467$	$\eta_{dc} = \frac{8d}{T_{ms}}$ $= \frac{8 \times 16}{276.439}$ $= 0.463$

Table D.3 – Throughput calculation

Developed PROFIBUS DP protocol	**Commercial PMC PROFIBUS module**
$U^{(\max)} = \eta_{dc}.\eta_{ma}.R$ $= 0.467 \times 1 \times 1.5\ Mbps$ $= 700.5\ kbps$	$U^{(\max)} = \eta_{dc}.\eta_{ma}.R$ $0.463 \times 1 \times 1.5\ Mbps$ $= 694.5\ kbps$

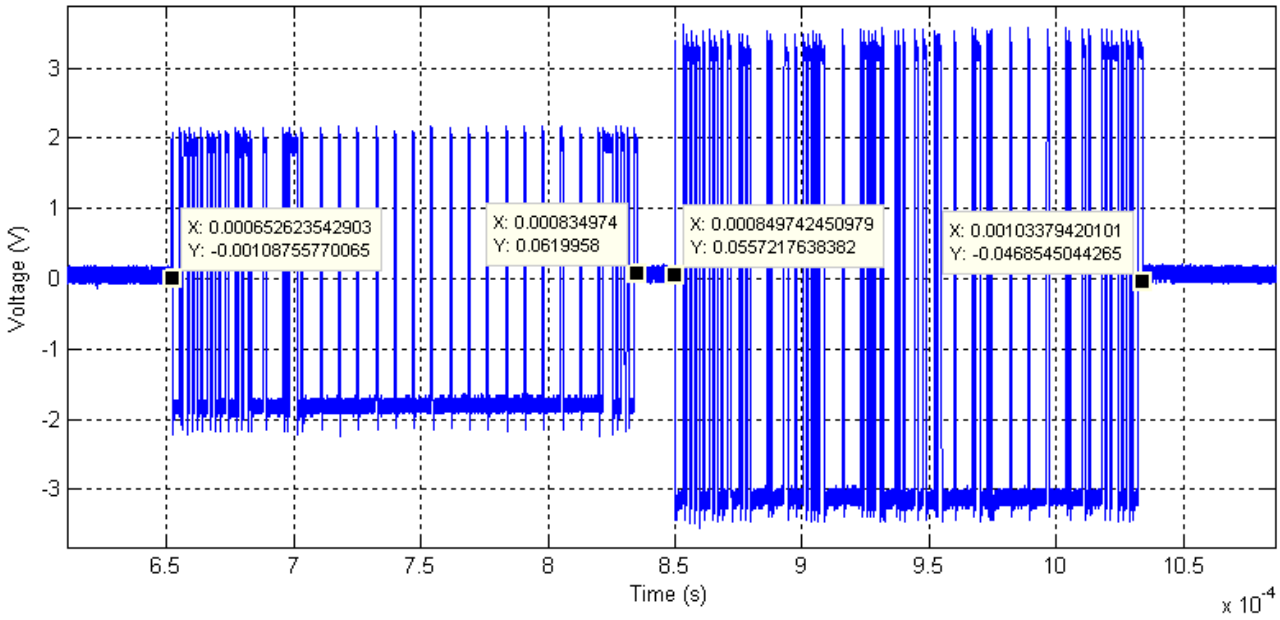


Figure D.1 – Developed PROFIBUS DP protocol

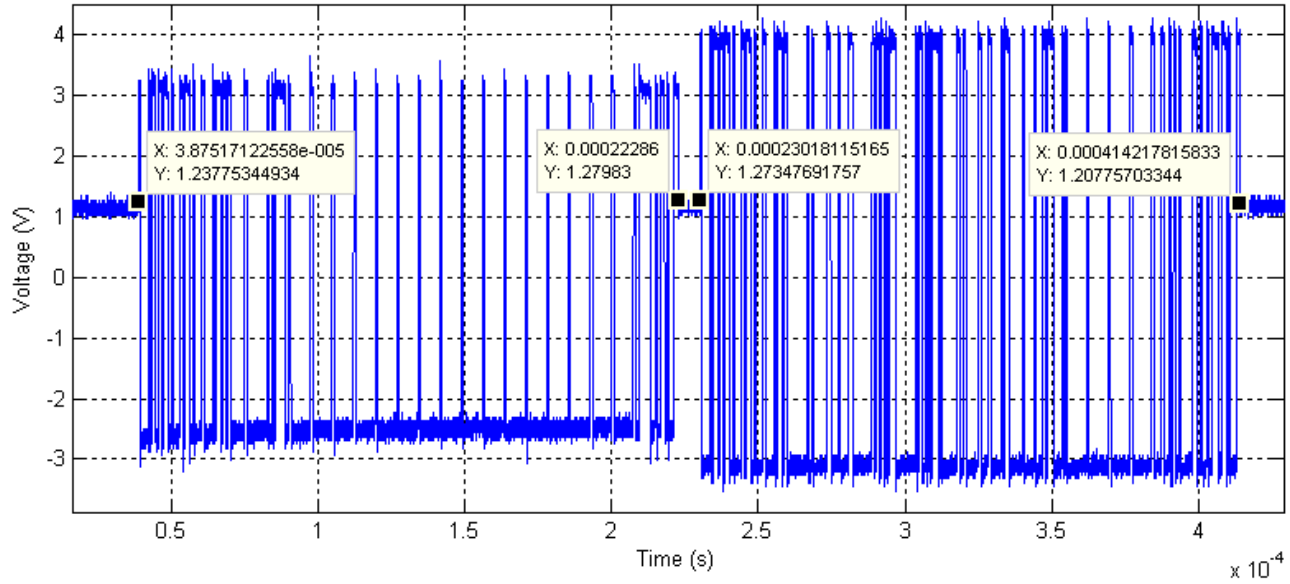


Figure D.1 – PMC 253 PROFIBUS DP protocol

Table D.4 – Signal-to-noise ratio

Developed PROFIBUS DP protocol	**Commercial PMC PROFIBUS module**
$Signal\ to\ noise\ Ratio = \frac{(\mu_{PTop} - \mu_{PBase})}{(\sigma_{PTop} + \sigma_{PBase})}$ $Signal\ to\ noise\ Ratio = \frac{(1.92 - (-1.81)}{(0.0463 + 0.0385)}$ $= 43.99\ V$ $20 \log_{10} SNR$ $20\log_{10} 43.99 = 32.87\ dB$	$Signal\ to\ noise\ Ratio = \frac{(\mu_{PTop} - \mu_{PBase})}{(\sigma_{PTop} + \sigma_{PBase})}$ $Signal\ to\ noise\ Ratio = \frac{(3.11 - (-2.63)}{(0.05341 + 0.05526)}$ $= 52.82\ V$ $20 \log_{10} SNR$ $20\log_{10} 52.82 = 34.46\ dB$

Appendix E – Fieldbus profiles of the IEC 61158

In 1999 the IEC Committee of Action together with representatives of the main Fieldbus contenders put aside their differences and developed the IEC 61158 standard that was released on the 31st of December 2000. The IEC 61158 standard was however useless for implementation and a manual was needed for practical use, showing which parts can be compiled into a functioning system and how this can be accomplished. This brought about the guidelines for the so-called 'profiles' of the Fieldbus systems and was defined in the IEC 61784. Table E.1 highlights the profiles and protocols according to the IEC 61158 and IEC 61784. The main profiles of table E.2 and some other popular profiles are discussed and compared in the next section.

Table E.1 – Fieldbus profiles and protocols acc. to the IEC 61158 and IEC 61784

IEC 61784 Profile	**IEC 61158 Protocols**			**GENELEC**	**Brand names**
	Physical	**Data link**	**Application**		
CPF-1/1	Type 1	Type 1	Type 9	EN 50170-A1(Apr. 200)	Foundation Fieldbus (H1)
CPF-1/2	Ethernet	TCP/UDP/IP	Type 5	-	Foundation Fieldbus (HSE)
CPF-1/3	Type 1	Type 1	Type 9	EN 50170-A1(Apr. 2000)	Foundation Fieldbus (H2)
CPF-2/1	Type 2	Type 2	Type 2	EN 50170-A3(Aug. 2000)	ControlNet
CPF-2/2	Ethernet	TCP/UDP/IP	Type 2	-	EtherNet/IP
CPF-3/1	Type 3	Type 3	Type 3	EN 50254-4(Oct. 1998)	PROFIBUS-DP
CPF-3/2	Type 1	Type 3	Type 3	EN 50170-A2(Oct. 1998)	PROFIBUS-PA
CPF-3/3	Ethernet	TCP/UDP/IP	Type 10	-	PROFInet
CPF-4/1	Type 4	Type 4	Type 4	EN 50170-1(Jul. 1996)	P-Net RS-485
CPF-4/2	Type 4	Type 4	Type 4	EN 50170-1(Jul. 1996)	P-Net RS-232
CPF-5/1	Type 1	Type 7	Type 7	EN 50170-3(Jul. 1996)	WorldFIP (MPS,MCS)
CPF-5/2	Type 1	Type 7	Type 7	EN 50170-3(Jul. 1996)	WorldFIP (MPS, MCS, SubMMS)
CPF-5/3	Type 1	Type 7	Type 7	EN 50170-3(Jul. 1996)	WorldFIP (MPS)

CPF-6/1	Type 8	Type 8	Type 8	EN 50254-2(Oct. 1998)	INTERBUS
CPF-6/2	Type 8	Type 8	Type 8	EN 50254-2(Under vote)	INTERBUS TCP/IP
CPF-6/3	Type 8	Type 8	Type 8	EN 50254-2(Under vote)	INTERBUS Subnet
CPF-7/1	Type 6	Type 6	-	-	Swiftnet transport
CPF-7/2	Type 6	Type 6	Type 6	-	Swiftnet full stack

Table E.2 – Technical characteristics and application domain of the Fieldbus profiles

Profile	**Name**	**Industry**	**Special features**	**Processing**	**Bus access**	**Nodes per segment**
CPF-1/1	Foundation Fieldbus (H1)	Process	Function blocks for decentralized control	Central Decentral	producer-consumer with distributor	Max. 32
CPF-1/2	Foundation Fieldbus (HSE)	Factory Process		Decentral	CSMA/CD	Max. 30
CPF-1/3	Foundation Fieldbus (H2)	Factory		Central Decentral	producer-consumer with distributor	Max. 32
CPF-2/1	ControlNet	Factory	Optimized for factory applications	Central	producer-consumer	Max. 99
CPF-2/2	EtherNet/IP	Factory				Max. 30
CPF-3/1	PROFIBUS-DP	Factory	Optimized for Re-mote I/O	Central	master-slave with token-passing	Max. 126
CPF-3/2	PROFIBUS-PA	Process	Optimized for Process Control	Central		Max. 32
CPF-3/3	PROFInet	Factory	Distributed Automation Objects	Decentral	producer-consumer	Max. 30

CPF-4/1	P-Net RS-485	Factory Shipbuilding	Multi-net capability	Central	master-slave with token-passing	Max. 32
CPF-4/2	P-Net RS-232					
CPF-5/1	WorldFIP (MPS,MCS)	Factory	Distributed realtime database	Central Decentral	producer-consumer with distributor	Max. 256
CPF-5/2	WorldFIP (MPS, MCS, SubMMS)					
CPF-5/3	WorldFIP (MPS)					
CPF-6/1	INTERBUS	Factory	Optimised for remote I/O	Central	single master with synchronised shift register	max. 256
CPF-6/2	INTERBUS TCP/IP					
CPF-6/3	INTERBUS Subnet					
CPF-7/1	Swiftnet transport	Aircraft	Optimised for aircraft	Central	producer-consumer with distributor	max. 1024
CPF-7/2	Swiftnet full stack					

Foundation Fieldbus

Foundation Fieldbus (FF) was established in 1994 by the merging of WorldFIP North America and the Interoperable Systems Project (ISP). FF is considered to be the leading organization dedicated to a single international, interoperable Fieldbus standard [22].

FF is an all-digital, serial, 2-way communication system that interconnects sensors, actuators and controllers. FF serves as a local-area network for process control and manufacturing automation in automation instruments, at base level. The communication method is a client/server, publisher/subscriber model that has the built-in capability to distribute the control application across the network. FF consists of the two Fieldbus protocols FF H1 and FF H2 (HSE) [22].

ControlNet

ControlNet was formed in 1997 by ControlNet International, an independent organization for users and vendors of ControlNet products. This 5 Mbps communications protocol enables users to predict data transmission and guarantee its arrival. This feature allows deterministic and repeatable performance for time-sensitive industries. System devices can include simple I/O or complex controllable devices [22].

P-Net

P-net was part of the European Standard EN 50170 since July 1996 and became the type 4 profile in the IEC 61158. This profile was designed to connect distributed process components such as process computers, I/O modules, intelligent sensors, actuators, field and central controllers, PLC's, etc [22].

Swiftnet

Swiftnet was developed by SHIPSTAR, which was founded in 1985. Swiftnet is a modern Fieldbus that addresses the need in Boeing' commercial airplanes, for a truly synchronous very high speed flight data bus. This protocol also produces unique synchronous global, group and individual triggers to stimulate device actions. This ensures that data transfer rates on the bus are not reduced, since these triggers use each the local clock of each device. This feature is a result of the use of a structured, synchronous time division multiple access (TDMA) bus protocol. Applications include flight test and simulation, direct numerical control, oil fields, wells and mining [22].

WorldFIP

WorldFIP is considered as an accepted technology for international contracts and therefore guarantees stability and openness, by offering the most practical and proven solution for Fieldbus systems. The bus provides a single communication technology for time-critical data and unscheduled messages. It is mostly used in PLCs and distributed control systems down to low-end non-intelligent sensors and actuators. WorldFIP also uses the produces or consumer model with a centralized bus scheduler [22].

Interbus

Interbus was also created by SHIPSTAR and developed by Phoenix Contract to be an open systems approach for a high performance, ring-based, distributed device network for manufacturing and process control. The ring topology used by this profile actively connects devices via a closed

loop path and therefore requires a 5-wire cable between devices to carry the logical ground as well. An integrated repeater function allows the profile to extend up to 13km [22].

CAN

The Controller Area Network (CAN) protocol is based on the carrier sense multiple access scheme with collision avoidance (CSMA/CD) and is enhanced with a deterministic collision based on the priority of the exchanged objects. To avoid collision of two stations transmitting at the same time, the contention on this bus is resolved by means of an arbitration phase, which will stop all the stations transmitting, except the one transmitting data with the highest priority. This ensures that the network never becomes congested by ensuring that no data or time is lost. By using this medium access control (MAC), all the stations in a CAN network can gain bus mastership and transmit. This allows for the implementation of a simple producer-consumer model which is ideally suited for asynchronous data exchange. CAN is highly suited to deal with periodic traffic and can substantially reduce the responsiveness of some nodes in the event of a heavy loaded event-driven system [23].

DeviceNet

DeviceNet was originally developed by Allen-Bradley and is managed by the independent supplier organization Open DeviceNet Vendors Association (ODVA). This low-level network is designed to connect industrial devices (sensors, actuators) to higher-level devices (controllers). The focus of the profile lies on the interchangeability of very low-cost, simple devices that is often used in manufacturing applications, such as limit switches, photo-electric sensors, motor starters, bar code readers, variable frequency drives, motor starters, etc. DeviceNet builds upon the CAN profile, where CAN only specifies sections of the physical an data link layer, DeviceNet adds the remainder of these layers and adds a media and application layer. DeviceNet can be defined as a digital, multi-drop network that connects and serves as a communication network between industrial controllers and I/O devices. Systems that implement this profile can be configured to operate in a master-slave model, or a distributed control architecture using peer-to-peer communication. A unique feature of DeviceNet is its ability of having power on the network that allows devices with limited power requirements to be powered directly from the network and thus reduces connection points and physical size [24].

HART communication protocol

Highway addressable remote transducer (HART) was developed in the mid 1980’s by Rosemount Inc. The profile is widely accepted in the industry, for digitally enhanced 4-20 mA communications

with smart field devices. HART was an acceptable profile for intelligent measurement and control instruments which traditionally used 4-20 mA signals for communications, by preserving the 4-20 mA signals with two-way digital communications, which occur without disturbance of the analog signals. The process variable is permitted to continue to be transmitted by the 4-20 mA signals, together with additional information pertaining to other variables, parameters, device configuration, calibration and device diagnostics [24].

MODBUS

MODBUS is a messaging structure, independent of the underlying physical layer, which is widely used to establish master-slave communication between intelligent devices. The MODBUS protocol either uses ASCII transmission, where each eight-bit byte is sent as 2 ASCII characters or remote terminal unit (RTU) transmission, where each eight-bit byte is sent as two four-bit hexadecimal characters. The RTU method achieves higher throughput, while the ASCII mode allows time intervals up to 1 second to occur between characters [24].

Appendix F – GUI memory access functions

Table F.1 – Read operation

Driver function – PCIVFX_WriteDWORD(handle, PCI_BAR2, 8060, Address1)	The value of "*Address1*" is written into the DDR SDRAM address register at address *PCIBAR2* + *8060*. "*Address1*" is the register that the user wants to read from the allocated DDR memory space in the table below. The *handle* is initiated when the FPGA is configured.
Driver function – PCIVFX_WriteDWORD(handle, PCI_BAR2, 805C, Data1)	By reading a value 02H from *Data1* into the control register at address *PCIBAR2* + *805C*, the control register is set to perform a read operation at the specified address of function1.
Driver function – PCIVFX_ReadDWORD(handle, PCI_BAR2, 8064, &ReadValue)	After function 2, the value of the memory address specified is read DDR SRAM address register at address *PCIBAR2* + *8064*. By performing function 3, the value that was read is moved to the variable *ReadValue.*

Table F.2 – Write operation

Driver function – PCIVFX_WriteDWORD(handle, PCI_BAR2, 8060, Address2)	The value of "*Address2*" is written into the DDR SDRAM address register at address *PCIBAR2* + *8060*. "*Address2*" is the register that the user wants to write to in the allocated DDR memory space in the table below. The *handle* is initiated when the FPGA is configured.
Driver function – PCIVFX_WriteDWORD(handle, PCI_BAR2, 806C, DataMaskValue)	By reading a value 00H from DataMaskValue into the DDR SRAM register at address PCIBAR2 + 806C, the mask register is set to 8 bit values.
Driver function – PCIVFX_ReadDWORD(handle, PCI_BAR2, 8068, WriteData)	The value of WriteData is written into the DDR SRAM address register at address PCI_BAR2 + 8068. When the control register is changed to perform a write operation, this value is written to

	the specified address.
Driver function – PCIVFX_WriteDWORD(handle, PCI_BAR2, 805C, Data2)	Reading a value of 01H from Data2 into the control register at address PCI_BAR2 + 805C, a write operation is performed that reads the value of function 3 into the address register specified in function 1.

By combining these read and write operations with the different GUI procedures, the GUI operates in real-time by changing states, sending PID parameters and displaying specific values of the AMB system. A large amount of data is continuously read and written through the GUI program. This consumes a large amount of processing power on the single-board computer. Table F.3 – displays the specific memory segments used together with their specific use.

Table F.3 – DDR Memory Map

Address	**Value**	**Use**
0x0FFFFFFC – 0x0FFFFFFF	Open	
0x0FFFFFF8 – 0x0FFFFFFC	Interrupt flag for state-machine	This memory space is continuously monitored on the single-board computer to detect a state change
0x0FFFFFF4 – 0x0FFFFFF8	Interrupt flag for PID parameters	This memory space is continuously monitored on the single-board computer to detect a change in the PID parameters
0x0FFFFFF0 – 0x0FFFFFF4	Power Amplifier Current 10	Memory space used to store the Power Amplifier current value
0x0FFFFFEC – 0x0FFFFFF0	Power Amplifier Current 9	Memory space used to store the Power Amplifier current value
0x0FFFFFE8 – 0x0FFFFFEC	Power Amplifier Current 8	Memory space used to store the Power Amplifier current value
0x0FFFFFE4 – 0x0FFFFFE8	Power Amplifier Current 7	Memory space used to store the Power Amplifier current value
0x0FFFFFE0 – 0x0FFFFFE4	Power Amplifier Current 6	Memory space used to store the Power Amplifier current value
0x0FFFFFDC – 0x0FFFFFE0	Power Amplifier	Memory space used to store the Power

	Current 5	Amplifier current value
0x0FFFFFD8 – 0x0FFFFFDC	Power Amplifier Current 4	Memory space used to store the Power Amplifier current value
0x0FFFFFD4 – 0x0FFFFFD8	Power Amplifier Current 3	Memory space used to store the Power Amplifier current value
0x0FFFFFD0 – 0x0FFFFFD4	Power Amplifier Current 2	Memory space used to store the Power Amplifier current value
0x0FFFFFCC – 0x0FFFFFD0	Power Amplifier Current 1	Memory space used to store the Power Amplifier current value
0x0FFFFFC8 – 0x0FFFFFCC	State Check	Used as redundancy for state change
0x0FFFFFC4 – 0x0FFFFFC8	State	Used by the FPGA to know the state of the system when a interrupt flag is triggered
0x0FFFFFC0 – 0x0FFFFFC4	Axial D-Parameter	Axial AMB D-Parameter for PID control
0x0FFFFFBC – 0x0FFFFFC0	Axial I-Parameter	Axial AMB I-Parameter for PID control
0x0FFFFFB8 – 0x0FFFFFBC	Axial P-Parameter	Axial AMB P-Parameter for PID control
0x0FFFFFB4 – 0x0FFFFFB8	Radial D-Parameter	Radial AMB D-Parameter for PID control
0x0FFFFFB0 – 0x0FFFFFB4	Radial I-Parameter	Radial AMB I-Parameter for PID control
0x0FFFFFAC – 0x0FFFFFB0	Radial P-Parameter	Radial AMB P-Parameter for PID control
0x0FFFFFA8 – 0x0FFFFFAC	AMB 2 – Z	AMB 2 – Z position
0x0FFFFFA4 – 0x0FFFFFA8	AMB 2 – Y	AMB 2 – Y position
0x0FFFFFA0 – 0x0FFFFFA4	AMB 2 – X	AMB 2 – X position
0x0FFFFF9C – 0x0FFFFFA0	AMB 1 – Z	AMB 1 – Z position
0x0FFFFF98 – 0x0FFFFF9C	AMB 1 – Y	AMB 1 – Y position
0x0FFFFF94 – 0x0FFFFF98	AMB 1 – X	AMB 1 – X position

Appendix G – PROFIBUS state machine details

Appendix G.1: Request diagnostics received bytes

Table G.1 to G.6 displays the diagnostic information data characters received from the slave.

Table G.1 – Request diagnostic data byte 1 [16] [26]

Bit	Diagnostic
0	**Diag.Station_Non_Existent:** Set to 1 by the master if slave cannot be reached over the line. Slave sets this bit to 0.
1	**Diag.Station_Not_Ready:** Set by slave if slave is not ready for data transfer.
2	**Diag.Cfg_Fault:** Set by slave if it detects a mismatch in configuration data.
3	**Diag.Ext_Diag:** Set by slave to indicate a diagnostic entry is in the slave-specific diagnostic area (see below).
4	**Diag.Not_Supported**: Set by slave if requested function/service is not supported.
5	**Diag.Invalid_Slave_Response:** Slave sets this bit to 0. Set to 1 by the master if it receives an implausible response from the slave.
6	**Diag.Prm_Fault:** Set by slave if last parameter frame was faulty (wrong parameterization, bad length, bad ident_number, etc.).
7	**Diag.Master_Lock:** Set by a class 1 master to indicate slave has been parameterized by another master (if address in DU byte 4 is not 255 and differs from its own address). Set to 0 by slave.

Table G.2 – Request diagnostic data byte 2 [16] [26]

Bit	Diagnostic
0	**Diag.Prm_Req**: Set by a slave if it needs to be parameterized and cleared once parameterization is complete.
1	**Diag.Stat_Diag:** Static diagnostics. Slave sets this bit to cause the master to retrieve diagnostic information until this bit is cleared (the slave sets it if it's not able to provide user data).
2	*Slave sets this bit to 1.*
3	**Diag.WD_ON:** Set by slave to indicate Watchdog is active.
4	**Diag.Freeze_Mode:** Set by slave after it has received the Freeze control command.
5	**Diag.Sync_Mode:** Set by slave after it has received a Sync command.
6	*Reserved.*
7	**Diag.Deactivated:** Set by the master if slave has been marked inactive within the slave parameter set and is removed from cyclic processing. Slave sets this bit to 0.

Table G.3 – Request diagnostic data byte 3 [16] [26]

Bit	Diagnostic
0-6	*Reserved.*
7	**Diag.Ext_Diag_Overflow:** Set if there is more diagnostic information than specified in Ext_Diag_Data. For example, slave sets if slave has more diagnostics than it can enter into its send buffer. Set by master if slave sends more diagnostic information than it can enter into its diagnostic buffer.

Table G.4 – Request diagnostic data byte 4 [16] [26]

Bit	Diagnostic
0-7	**Diag.Master_Add:** The master's address that parameterized this slave is entered here. If no master has parameterized this slave, then the DP Slave inserts 255 here (FF without parameterization).

Table G.5 – Request diagnostic data byte 5 [16] [26]

Bit	Diagnostic
0-7	Manufacturer Identification Number High byte for ID & verification

Table G.6 – Request diagnostic data byte 6 [16] [26]

Bit	Diagnostic
0-7	Manufacturer Identification Number Low byte for ID & verification.

Appendix G.2: Parameterization transmitted bytes

Table G.7 to G.11 lists the data units with their specific setting for this design.

Table G.7 – Parameterization telegram data unit byte 1 [16] [26]

Bit	7	6	5	4	3	2	1	0
Parameter	Lock_Req	Unlock_Req	Sync_Req	Freeze_Req	WD_On	*Reserved –Set to 0*		
Value	1	0	0	0	1	0	0	0

WD_On: By setting this bit to 1, the watchdog control is activated.

Freeze_Req: By not setting this bit to 1, the freeze mode is not activated for the slave

Sync_Req: By not setting this bit, the slave will not operate in the synchronization mode

Unlock_Req: Not set

Lock_Req: The slave is locked for other masters

Table G.8 – Parameterization telegram data unit byte 2

Bit	7	6	5	4	3	2	1	0
Value	0	0	1	0	1	0	0	0

Table G.9 – Parameterization telegram data unit byte 3

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	0	1

The watchdog control is enabled in data unit 2. Bytes 2 and 3 are used as factors to set the watchdog time. The watchdog control will cause the slave to go to an idle state if the master fails to communicate with the slave before the time expires. The watchdog time is calculated with (F.1).

$$T_{WD} = 10\,ms \times WD_{factor1} \times WD_{factor2} \quad \text{(F.1)}$$

$$T_{WD} = 10\,ms \times 40 \times 1 = 400\,ms$$

Table G.10 – Parameterization telegram data unit byte 4

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	0	0	0	0

Byte 4 of the data units sets the minimum station delay response time of the slave. The slave must wait this amount of time before responding to a request from the master. By setting this value to 0 as shown in table H.10, the default value of the slave is selected.

Table G.11 – Parameterization telegram data unit byte 5

Bit	7	6	5	4	3	2	1	0
Value	0	0	0	0	1	0	0	0

Table G.12 – Parameterization telegram data unit byte 6

Bit	7	6	5	4	3	2	1	0
Value	1	0	0	1	1	1	0	1

Data units 5 and 6 are used for security purposes and represent the slave's identity number. Byte 5 represents the high byte and byte 6 the low byte of the identity number. For the specific slave in this design the value of the two bytes – 00001000 10011101 – represents the identity number '89D' in

hexadecimal which represents the slave's identity number. The slave will verify these bytes against its own identity number and only transmit data to the master if the identity number is correct. Data unit 7 is used for the global control state and is not used in this design and therefore set to 0.

Appendix G.3: Configuration transmitted byte

Table G.13 – Configuration telegram data unit byte

Bit	7	6	5	4	3	2	1	0
Value	0	1	1	1	0	1	1	1